UNIVERSIDADE DE LISBOA
Faculdade de Ciências
Departamento de Informática

DETACH: DESIGN TOOL FOR SMARTPHONE APPLICATION COMPOSITION

Filipe Miguel da Costa Fernandes

DISSERTAÇÃO

MESTRADO EM ENGENHARIA INFORMÁTICA
Sistemas de Informação

2013

UNIVERSIDADE DE LISBOA
Faculdade de Ciências
Departamento de Informática

DETACH: DESIGN TOOL FOR SMARTPHONE APPLICATION COMPOSITION

Filipe Miguel da Costa Fernandes

DISSERTAÇÃO

Trabalho orientado pelo Prof. Doutor Luís Manuel Pinto da Rocha Afonso Carriço

e co-orientado por Luís Miguel Santos Duarte

MESTRADO EM ENGENHARIA INFORMÁTICA
Sistemas de Informação

2013

Resumo

A Terapia Comportamental Cognitiva (TCC) é uma forma de tratamento que se foca nas relações entre pensamentos, emoções e comportamentos. A TCC pode assim mudar a forma como pensamos (cognição) e como reagimos (comportamento), de maneira a que nos possamos sentir melhor. Para ter sucesso, esta intervenção psicológica usa processos sistemáticos, que são normalmente compostos pelo preenchimento de formulários em papel, com um determinado objectivo específico. Este objectivo poderá ser o tratamento de várias condições: humor, ansiedade, personalidade, obesidade, fobias, depressões, controle de dor, etc.

Actualmente, esta área da terapia é normalmente dividida em dois tipos de sessões distintas com os pacientes. Se por um lado existe a sessão no consultório com o terapeuta, por outro, é muitas vezes pedido ao paciente que complete algumas tarefas no exterior, sem a presença do mesmo. Estas tarefas pretendem que este pratique as diversas situações analisadas com o terapeuta. Para isso são usados questionários e formulários em papel que permitem que o paciente registe os seus pensamentos e experiências. No entanto, a incapacidade de adaptação dos mesmos ao paciente e às diversas situações em que são na verdade utilizados resulta na falta de motivação, paciência e sentimento de acompanhamento pessoal para o paciente. Além disso, o tipo de suporte em que estes questionários e formulários assentam acabam por fazer com que o paciente muitas das vezes se esqueça deles em casa ou tenha vergonha de os transportar e preencher em público.

Assim, este trabalho vem propor, através das várias tecnologias presentes nos smartphones de hoje em dia, um melhoramento à vertente no exterior do processo terapêutico, que muitas vezes é desprezada pelo paciente ao fim de pouco tempo pondo em causa o sucesso do tratamento. Com isto, pretende-se que o paciente se sinta acompanhado, a todo o momento, por um terapeuta, através de uma aplicação presente no seu próprio smartphone. Idealmente esta aplicação seria criada especificamente para a pessoa em questão e conseguiria adaptar-se às várias situações em que iria ser usada, ajudando eficazmente o paciente.

A criação destas aplicações requer conhecimentos técnicos que não estão presentes em todas as pessoas. Se por um lado os especialistas da área de programação são capazes de tal criação, por outro são os especialistas da área da saúde, e neste caso em específico os terapeutas, que são capazes da sua idealização, consoante as necessidades dos seus pacientes. Surge então a necessidade da criação de aplicações móveis, adaptáveis ao

contexto do utilizador, por pessoas que não sejam especialistas na área da programação. Assim, este projecto assenta sob três entidades distintas: a) Utilizadores não programadores, que utilizarão uma ferramenta de criação de aplicações móveis; b) Utilizadores das aplicações móveis, que utilizaram as aplicações criadas pelos anteriores; c) Utilizadores programadores, que irão desenvolver novas componentes para a ferramenta de autoria.

Nas fases iniciais do desenvolvimento tentámos perceber através de sessões de desenho participativo, como é que os utilizadores não programadores interagiam com um conjunto de elementos presentes num protótipo de baixa fidelidade. Usando algum material de desenho como post-its, folhas de papel, lápis e borracha, propusemos aos participantes que representassem uma determinada aplicação passível de ser utilizada num dispositivo móvel. Os resultados que obtivemos mostraram que a presença destas mesmas representações que foram fornecidas aos participantes permitiram que os mesmos criassem aplicações muito mais complexas e potentes. Adicionalmente estas sessões permitiram também que percebêssemos que a generalidade dos participantes preferiu organizar os elementos da aplicação sobre uma área de trabalho "infinita" e não apenas limitada a uma sequência linear. O método preferencial de relação entre estes mesmos elementos foi a utilização de setas que indicavam quando seria executada a transição entre eles.

Com os resultados anteriores construímos o primeiro protótipo funcional de DETACH (DEsign Tool for smarphone Application Composition - Ferramenta de Desenho para a Composição de Aplicações para Smartphones). De acordo com os resultados obtidos nas sessões anteriores com utilizadores, este protótipo continha uma área com alguns ecrãs padrão passíveis de serem usados num ambiente de trabalho "infinito". Com este protótipo pedimos a alguns utilizadores não programadores que tentassem criar uma aplicação móvel adequada à sua área do conhecimento. Com este primeiro protótipo verificámos que nenhum dos participantes foi capaz de completar todos os passos com sucesso. O aspecto mais problemático que verificámos foi na maneira como os mesmos ligavam os vários ecrãs da aplicação. O facto de o protótipo disponibilizar num dado ecrã a possibilidade de criar dois tipos de ligações, uma que considerasse o mesmo como destino e outra como origem, acabou por confundir os participantes. Apesar de tudo verificámos que a maior parte intencionava usar ambos as ligações considerando o ecrã selecionado como origem.

Tais resultados levaram-nos à criação de um novo protótipo com as preferências verificadas pelos utilizadores. Após apresentarmos este último a um novo grupo de utilizadores, para a criação da mesma aplicação, verificámos que mais de metade deles concluiu a mesma com sucesso e em cerca de metade do tempo verificado no protótipo

anterior. Tais resultados indicavam que claramente ainda havia espaço para melhoramentos ao protótipo. A visibilidade de algumas das funcionalidades, que ainda não estava clara para alguns dos utilizadores, foi um dos factores negativos apontados.

A utilização destes dois protótipos iniciais de DETACH permitiu-nos perceber alguns padrões usados pelos nossos participantes aquando da ligação entre ecrãs, a funcionalidade verificada mais crítica. Enquanto a maior parte dos utilizadores preferiu criar ligações considerando o ecrã selecionado como origem, verificamos também a utilização de outros padrões que se assemelhavam aos usados em ferramentas complexas de programação. Para o produto final decidimos seguir assim a abordagem mais utilizada pelos utilizadores, melhorando o último protótipo.

Através dos requisitos reunidos nestas sessões com utilizadores, no trabalho relacionado e em algumas reuniões que tivemos com terapeutas, definimos algumas métricas que a ferramenta DETACH seguiria. De maneira a ser de fácil utilização por qualquer utilizador, proporcionando também o seu alcance através de qualquer plataforma, desktop ou móvel (em tablets por exemplo), seria criada uma ferramenta web. Esta recorreria a um servidor para guardar os projectos criados, para que os mesmos pudessem ser carregados remotamente para os dispositivos móveis destino. De maneira a criar uma ferramenta robusta que possa ser melhorada no futuro com novos tipos de ecrãs ou variáveis de contexto utilizáveis, era necessário também desenvolver a mesma de uma maneira altamente modular para ser então continuada por programadores profissionais.

A versão final da ferramenta DETACH nasceu de todo o conjunto de requisitos que verificámos anteriormente. Através de melhoramentos à última versão do protótipo de alta-fidelidade usado chegámos a uma ferramenta que pode até ser usada para além da área da terapia, como para a área de jogos ou do ensino. Para uma correcta avaliação da ferramenta, recorremos a utilizadores distintos daqueles que tinham testado os protótipos iniciais de alta-fidelidade. Os resultados mostraram que todos os participantes conseguiram criar, testar e atribuir a aplicação proposta aos utilizadores destino.

Para além da avaliação realizada com os utilizadores finais da ferramenta, pedimos também a alguns programadores para tentarem estender a mesma adicionando um novo ecrã. Os resultados mostraram que também todos os programadores conseguiram realizar a tarefa com sucesso numa média de cerca de 35 minutos.

O processo de desenvolvimento deste projecto contribuiu com a publicação de três artigos para conferências na área da saúde e da interacção.

Palavras-chave: Ferramentas de autoria, Desenho participativo, Programação visual, Web, Smartphone, Terapia.

Abstract

This thesis focuses on the Cognitive Behavioral Therapy (CBT) area. This type of therapy is normally subdivided in two kinds of sessions: the ones where the therapist and the patient are both inside an office, and the ones where the patient is outside the therapist office and has to follow some homework tasks, alone. These tasks intend patients to practice the situations analyzed in the sessions with therapists and are normally supported with simple paper forms. The inexistent ability for these homework tasks to adapt themselves to the patient or different use contexts compromises the success of the treatment. It is hence important to find a way where the patient, while outside the office, doesn't feel that difference, because he carries a virtual therapist inside his smartphone.

The usage of modern mobile phones can address the previous problem. Existing solutions encompass replacing traditional treatment methods with a mobile application that is provided to the patients. However, the application content is the same for every patient, regarding the age or treatment focus. Therapists lack the knowledge to create their own mobile applications and information technologies professionals lack the ability to personalize their contents properly.

This work aims at circumventing this situation with the introduction of DETACH (DEsign Tool for smartphone Application Composition), a system that comprises: a) a flexible enough platform that allow developers to easily add new components and enables non-programmer users to create powerful mobile applications; b) a framework that runs previously created mobile applications. Particularly important was the user-centered development process of this system.

We conducted a series of participatory design and thinking aloud trials with non-programmer users aiming to understand how they conceptualized programming. The results of interacting with low and high fidelity prototypes provided us with a set of interaction patterns and behaviors which we capitalized on in order to design the final DETACH product.

Afterwards DETACH was submitted to some tool evaluation tests, by asking non-programmer users to create a mobile application and developers to create a new component for the authoring tool. The results proved the tool success as every participant was able to complete the requested tasks.

Keywords: Authoring tools, Participatory design, Visual programming, Web, Smartphone, Therapy.

Content

Figures List

Tables List

Chapter 1

Introduction

Cognitive Behavioral Therapy (CBT) is a type of treatment that addresses the relations between thoughts, emotions and behaviors. Patients engage in a series of sessions with therapists that are supported with homework tasks. These tasks intend patients to practice the situations analyzed in the sessions with therapists and are normally supported with simple paper forms. The inexistent ability for these homework tasks to adapt themselves to the different use contexts compromises patients commitment to the treatment.

Mobile phones can address the previous problem. These devices have been evolving quickly in the last few years, replacing others such as digital cameras, dedicated GPS devices, gaming consoles and music players. The aggregating of all these features with such a computational power rivals low-spec personal computers.

While not everyone has one, the number of people adopting these devices has been rising and, in 2011, already one third of American adults owned a smartphone [1]. Two years later there are reports of countries with a smartphone coverage of 3 owners for each 4 individuals[1]. With such a massive adoption, the potential of these devices as a support tool in everyday tasks and as an object which may improve our quality of life also scales.

Existing solutions encompass replacing traditional treatment methods with a mobile application that is provided to the patients. However, the application content is the same for every patient, regarding the age or treatment focus. Therapists lack the knowledge to create their own mobile applications and information technologies professionals lack the ability to personalize their contents properly.

This work aims at circumventing this situation, by allowing health professionals to create personalized mobile applications that can adapt themselves to different use contexts.

[1] Our Mobile Planet: http://www.thinkwithgoogle.com/mobileplanet/en/ (Accessed 09 September 2013)

1.1 Motivation

This work's main application domain focuses on the Cognitive Behavioral Therapy (CBT) domain. CBT can change the way we think (cognition) and the way we act (behavior) in order to make us feel better. The goals are the treatment of a variety of conditions, including mood, anxiety, personality, eating habits, fears, depressions, pain control, etc. [2] [3] [4] [5]. For the success of this type of treatment, a number of goal-oriented, explicit systematic procedures are followed.

The range of methods utilized in this type of treatment is wide and based on sessions performed with therapists. These sessions may be supplemented with homework tasks. Assignments such as readings, behavior monitoring, and training of different ways to understand situations and their responses should be given to the patient to practice and use outside sessions. Homework assignments facilitate patient skill acquisition, treatment compliance, and symptom reduction by integrating the concepts learned in sessions into daily life. Homework is a key mechanism for facilitating between-session work and progress [2] [4] [5].

These homework activities strongly rely on the utilization of paper artefacts, such as questionnaires and forms, in order to allow patients to register their thoughts and experiences. Research in the area suggests this type of artefacts has a very low adherence rate [5][8]. Researchers justify it with patients often forgetting to carry the artefacts with them. Additionally, when they do bring them, they often wait until the therapy session day to perform the tasks or fill-in the questionnaire. This is problematic because certain assignments, such as monitoring automatic thoughts, are most effective and most accurate when completed at specific moments (i.e. typically when the patient is confronted with a situation related to his / her pathology) [4]. Identified limitations include: unreliable retrospective completion of diaries [8] and time intensive data entry [8]. In addition, this method provides little privacy or security to participants and may not be available when it is needed. These paper-based procedures also lack on the communication and customization, making them too generic and low in motivation, since they are not adapted to each patient's needs. Lastly, these artefacts are unable to adapt their content in critical scenarios where the patients' activities may not face the predicted conditions.

From the therapist point of view, the use of these paper-based artefacts also causes delay when gathering and analyzing the data collected. Storing and searching these artefacts is also a critical task.

The described problems can be solved by using the patient's smartphone, that he / she carries all the time, to replace these generic forms with ones that can adapt themselves to the patient's needs. With rich, mobile applications, it is possible to increase patients'

endeavor and therefore make treatments more successful by improving their state [6] [7] [8] [9] [10] [11].

Therapists, on the other hand, could recur to sophisticated tool suites which empower them to carefully manage patient records and track their evolution as the interventions progress.

Success cases for pathologies and therapy procedures on the benefits of technology are as diverse as autism [12], fear therapy [13], aphasia [11] or obsessive-compulsive disorder [14]. However, a significant number of these applications fall short to success for longer necessary periods [15]. Several factors can account for this outcome, among which the inability to personalize and adapt content [16]. For instance, an application's presentation is typically the same for all users who download it. Yet, the expectations of a potential 8 year old user are quite different from those of a 45 year old patient [13]. Also, the evolution of the patient's health status often requires adjustments that applications are not ready to accompany. For example, monitoring thresholds vary, support messaging and data collection should be adapted to new clinical assessments [9] [16].

The origin of this application stiffness builds on many factors. The complexity of the technology and of the application domain is certainly one of those reasons. In fact we believe that it is one of the most important factors: the dichotomy and complexity of knowledge involved. Information technology engineers and researchers understand technology and are able to handle its complexity. Clinicians on the other end comprehend patients and the protocols they must put forward to provide them a better quality of life. Combining the two knowledge sources is no easy task. It gets worst because both knowledge domains evolve rapidly as well as the ultimate target, the patient wellbeing.

The solution is not to give IT professionals the knowledge to help patients under a specific treatment, nor to make therapist learn how to code. It is in fact the therapist, the person who knows the patients more than anyone that should create and adapt the content of these mobile applications. While one could ask an IT professional to code the applications, the feasibility of this solution would be questionable as he / she would have to:

a) Wait for an application to be created, already considering that everything was developed as requested;
b) Pay for the request to the IT professional;
c) Iterate (a) and (b) for every patient of him undertaking the same type of therapy, but with other symptoms;
d) Iterate (a) and (b) for every change asked in a specific application, due to a progress of one of his patients.

Therefore, a proper solution must be developed to give non-expert programmers the power to create mobile applications.

1.2 Background

This work is part of the InSiThe (In-Situ Therapy Support) research project. InSiThe aims at providing an Information and Communication (IC) platform to therapeutic settings, coping with a diversity of scenarios which are known to be featured in such treatments. Among these, remote (i.e. therapists and patients are geographically and / or time distributed) and sessions taking place in outdoor settings emerge as some of the most challenging scenarios. The project contributes to the definition of a set of models, a specification language and an IC platform that fully covers a specific area of therapy (Cognitive Behavioral Therapy) in a relevant subset of scenarios.

The project also focuses on the development of a set of tools specifically targeting a variety of scenarios, such as outdoor settings. The toolset is intended for therapists and patients, having in mind the software's usability, communication and resource requirements.

1.3 Goals

To solve the identified problems, this project will follow a user-centered design methodology in order to:

- **Find how non-expert programmers deal with programming elements.** This includes the identification of organizational and interaction patterns, while at the same time we strive to analyze which programming concepts do non-expert programmers have.
- **Develop a tool where the therapist specifies the behavior of the patient´s artefacts** (e.g. when to provide help, when to reassure) mimicking its own behavior on a classic session. In this application we aim at allowing users with little to no programming skills to create mobile applications that can adapt themselves to the context in which they are being used. The usage of authoring tools emerged as a viable and possible solution. Past works proved to be valuable in tackling similar situations [12]. These tools aim at joining two knowledge sources to a middle ground. They provide domain experts the mechanisms to build, customize and adapt applications, refining them as new requirements emerge. For that, the tools must hide the technological complexity under a well-defined set of components, developed by IT staff. In the end they offer means for end-user programming. Finding that adequate middle-ground can be complicated. It is not just about usability. It is also about

programming and domain concepts, and ultimately the domain experts' perception of its combination. From the developers' point of view, the authoring tool has to allow an IT professional to easily create new components that can be added to the tool in order to provide new functionalities in the future. With this point we therefore want to provide a tool that improves alongside with new technologies so that people do not stop using it.

- **Evaluate the developed tool.** Previous created tool must target their stakeholders properly. This includes being able to create personalized context-aware mobile applications by non-programmers, as the creation of new tool components by IT professionals.
- **Develop a framework that interprets mobile applications**. Rule-specifications for non-programming-experts are in it a major challenge and overall these tools raise expectation for prominent Human-Computer Interaction and Computer Supported Cooperative Work contributions. Therefore we also need to concretize a framework that will interpret and run the created applications in a mobile environment. This framework has to offer the means to easily load these applications in order to be used in people's mobile devices. This framework must also be sensible to the context in which it can be used in. This will allow the mobile applications created to be able to react themselves when facing different contextual information. Also, this framework should be able to run high-quality applications that make use of audio and animations.

1.4 Contributions

This work's main contribution is DETACH (DEsign Tool for smartphone Application Composition) – an authoring tool which aims at allowing individuals, with and without programming skills, to create mobile Android applications.

The development process of this tool also allowed us to identify how non-expert programmers cope with design elements when designing an application that can have different flows according to conditional transitions and environment variables. More specifically we identified patterns users like to follow when representing, organizing and connecting these application elements.

The following publications emerged from this work:

- Filipe Fernandes, Luís Duarte, Luís Carriço (2013). *Flow Specification Patterns of End-User Programmers: Lessons Learnt from a Health Mobile Application Authoring Environment Design*. Human-Computer Interaction - INTERACT 2013, 14th IFIP TC13 International Conference, Cape Town,

South Africa, September 2- 6, Lecture Notes in Computer Science. Springer: Berlin/Heidelberg, 2013. CORE rank A;

- Filipe Fernandes, Luís Duarte, Luís Carriço (2013). *DETACH: Authoring Digital Therapeutic Artefacts*. 7th International Conference on Pervasive Computing Technologies for Healthcare (Pervasive Health '13), Venice, Italy, April 2- May 8, 2 pages. IEEE: New York, NY, USA, 2013;
- Filipe Fernandes, Luís Duarte, Luis Carriço (2013). *DETACH, Criação de Aplicações Móveis para Todos*. 5ª Conferência Nacional em Interacção Pessoa-Máquina (Interacção 2013), Trás-os-Montes e Alto Douro, Portugal, November 07- 08. GPCG, 2013.

1.5 Planning

The development of this thesis work was divided in the following phases:

- **Phase 1 - September 2012 / October 2012: Related work and technologies study**

  We started by analyzing previous work in the area of application creation for computers and mobile devices. The purpose of this investigation was to figure out the existing possibilities and adapt them, or create new ones, for our final mobile applications authoring tool. Additionally it was also studied current mobile applications, not only in the therapy area but in the health domain in general, in order to identify main requirements and key functionalities our tool should offer.

  This initial study also focused on the possible technologies the mobile applications created could be based on, with adequate tests in order to verify the possibilities of each of them.

- **Phase 2 - November 2012: Participatory Design Sessions**

  During this second working phase several participatory design sessions were made, with sketches and low-fidelity prototypes, in order to identify the requirements and basic elements the final authoring tool should offer. These sessions were made with non-programmer users, more specifically, users from the health area.

  At the end of these sessions, an initial report was also written with the research made to this working point and about how would the work evolve till the end.

- **Phase 3 - December 2012: Prototype creation**
  This third phase consisted on the creation of some interface sketches and low-fidelity prototypes of the authoring tool to be developed. Additionally a simple static mobile application was also created, related with a specific therapy procedure, in order for us to emulate some of the functionalities our system should offer.

- **Phase 4 - January 2013: First functional prototype creation**
  After the user evaluations made, and according to the results obtained, the first functional prototype of our authoring tool was created. This prototype offered functionalities such as connecting and configuring tool elements.

- **Phase 5 - February 2013: First functional prototype tests**
  Like the initial participatory design sessions made, this prototype was also presented to some health professionals in order for them to test it with the creation of a specific mobile application and we could identify the possible problems that could come from that task.

- **Phase 6 - March / April / May 2013: Transition from prototype to final product**
  The previous results were analyzed and according to them we refined the initial created prototype. All the functionalities not implemented so far became functional.

- **Phase 7 - June 2013: Final application tests**
  During this month the mobile application to install in the users smartphone, in order to run the created applications, was improved and finished.
  Finally a final set of tests with the resulting authoring tool and non-programmer users was also made, in order to prove the success of the developed work.

- **Phase 8 - July / August 2013: Final thesis**
  The last phase of this work was the writing of this document with all the developed work and results achieved.

In the end all the initial work was executed, even though a slight delay was verified in the last half of phases. This delay was mainly caused in phase 6, during the improvement of the initial prototype, and was due to the concern of the correction and implementation of,

not only the main tool functionalities, but also all the smallest details not required to the project but felt needed to be present in order for the final tool to be successful.

1.6 Document organization

This document is organized as follows:

- **Chapter 2 – Related work**
  In the next chapter we will focus in this project domain, available applications and tools for end-user programming.
- **Chapter 3 – Understanding Users' Programming Concepts**
  In this chapter we present the results of a series of participatory design sessions with non-expert programmers in order to understand how would they cope with programming elements.
- **Chapter 4 – Understanding Users' Interactions Patterns**
  This chapter presents the results of some trials made with initial DETACH prototypes in order to identify user organization and connection strategies when using this type of authoring tools.
- **Chapter 5 – DETACH System Overview**
  This chapter will convert related work, our meetings with therapists and our user studies into system requirements. This include functional and non-functional requirements our tool should verify by the end of this project.
- **Chapter 6 – DETACH for Non-Expert Programmers**
  In this chapter we describe all the details regarding our final DETACH product from the non-expert programmers' point of view.
- **Chapter 7 – DETACH for Developers**
  In this chapter we describe how a developer can interact with DETACH by adding new components to the tool.
- **Chapter 8 – DETACH Mobile & Emulator**
  This chapter describes the framework that is going to run the mobile applications created with DETACH.
- **Chapter 9 – Evaluation**
  In this chapter we present the results of some trials with both end-users and developers, in order to prove the success of our tool.

- **Chapter 10 – Conclusions & Future Work**
  The final chapter presents the conclusions that we took from all our work. As there will be still some space for improvements, we are also going to point DETACH possible future work.

Chapter 2

Related work

This chapter presents aspects related to this project domain, namely Cognitive Behavioral Therapy, and existing applications. Even though we do not want to restrict the final products to be used only in this domain, we found it necessary to focus our development process in a specific case. Besides, it also analyzes the tools and systems that hide their complexity in order to enable non-expert user programming. We therefore will analyze the ones that make this possible, more specifically visual programming authoring tools.

2.1 Cognitive behavioral therapy

Almost all human emotions and behaviors are the result of what people think, assume or believe about situations they face [5]. CBT is a form of psychotherapy that emphasizes the important role of those thoughts about how we feel and what we do. In this way, CBT helps a person to check out the reality of their beliefs, with the help of sophisticated techniques to achieve this empirical aim.

To improve their health states, patients engage in a series of sessions with therapists [17]. What enables CBT to be briefer than other types of therapy is its highly instructive nature and the fact that it makes use of homework assignments.

CBT therapists believe that the patients change because they learn how to think differently and they act on that learning. Therefore, CBT therapists focus on teaching rational self-counseling skills. The therapist's role is to listen, teach, and encourage, while the patient's roles is to express concerns, learn, and implement that learning.

Cognitive-behavioral therapists have a strict agenda for each session where specific techniques / concepts are taught during each session. The goal of therapy is to help patients unlearn their unwanted reactions and to learn a new way of reacting. Goal achievement (if obtained) could take a very long time if a person were only to think about the techniques and topics taught for one hour per week, during the session with the therapist. That's why CBT therapists use homework tasks, with reading assignments and questionnaires, encouraging their patients to practice the techniques learned.

2.1.1 Actors

This type of psychotherapy comprises therapist(s) and patient(s).

Therapists diagnose and gather the appropriate information about patients. This includes the analysis of paper forms and questionnaires filled by the latter. They participate in the sessions with patients by taking notes and discussing about their attitudes or reactions. Finally, they are also responsible for gathering and studying all this information in order to propose a treatment plan to the patient.

Patients, on the other hand, are the ones that are being treated by the therapists. This includes the filling of paper forms and questionnaires about their thoughts or attitudes. These individuals participate in the sessions with therapists by exposing their feelings to critical situations and practice new skills with homework tasks.

2.1.2 Process

The first step of a CBT intervention [5] is to build a relationship with the patient helping him to identify the problematic beliefs. This stage is important for learning how thoughts, feelings, and situations can contribute to maladaptive behaviors. The process can be difficult, especially for patients who struggle with introspection, but it can ultimately lead to self-discovery and insights that are an essential part of the treatment process.

Most of the sessions will occur afterwards using activities like:

- Analyzing specific episodes where the target problems occur, ascertaining the beliefs involved, changing them, and developing relevant homework (known as ‘thought recording’ or ‘rational analysis’);
- Developing behavioral assignments to reduce patient symptoms or modify ways of behaving;
- Supplementary strategies & techniques as appropriate, e.g. relaxation training, interpersonal skills training, etc.

Toward the end of the intervention it will be important to check whether improvements are due to significant changes in the patient’s thinking, or simply to a fortuitous improvement in their external circumstances.

It is usually very important to prepare the patient to cope with setbacks. Many people, after a period of wellness, think they are ‘cured’ for life. Then, when they slip back and discover their old problems are still present to some degree, they tend to despair and are tempted to give up self-help work altogether. In this last phase therapists warn patients that relapse is likely for many mental health problems and ensure they knows what to do when their symptoms return.

2.1.3 Techniques

There are no techniques that are essential to CBT – one uses whatever works, assuming that the strategy is compatible with CBT theory [5]. However, the following are examples of procedures in common use.

Cognitive techniques

- Rational analysis: analyses of specific episodes to teach patient how to uncover and dispute irrational beliefs. These are usually done in-session at first – as the patient gets the idea, they can be done as homework.
- Reframing: another strategy for getting bad events into perspective is to re-evaluate them as 'disappointing', 'concerning', or 'uncomfortable' rather than as 'awful' or 'unbearable'. A variation of reframing consists in helping the patient see that even negative events almost always have a positive side to them, listing all the positives the patient can think of.

Behavioral techniques

One of the best ways to check out and modify a belief is to act. Patients can be encouraged to check out the evidence for their fears and to act in ways that disprove them.

- Exposure: possibly the most common behavioral strategy used in CBT involves patients entering feared situations they would normally avoid. Such 'exposure' is deliberate, planned and carried out using cognitive and other coping skills. The purposes are to (1) test the validity of one's fears (e.g. that rejection could not be survived); (2) make them less awful (by seeing that catastrophe does not ensue); (3) develop confidence in one's ability to cope (by successfully managing one's reactions); and (4) increase tolerance for discomfort (by progressively discovering that it is bearable).
- Hypothesis testing: with this variation of exposure, the patient (1) writes down what they fear will happen, including the negative consequences they anticipate, then (2) for homework, carries out assignments where they act in the ways they fear will lead to these consequences (to see whether they do in fact occur).
- Risk-taking: the purpose is to challenge beliefs that certain behaviors are too dangerous to risk, when reason says that while the outcome is not guaranteed they are worth the chance. For example, if the patient has trouble with perfectionism or fear of failure, they might start tasks where there is a chance of failing or not matching their expectations. Or a patient who fears rejection might talk to an attractive person at a party or ask someone for a date.

- Paradoxical behavior: when a patient wishes to change a dysfunctional tendency, it is important to encourage them to deliberately behave in a way contradictory to the tendency. Emphasize the importance of not waiting until they 'feel like' doing it: practicing the new behavior – even though it is not spontaneous – will gradually internalize the new habit.

Other strategies

Other CBT strategies are:

- Skills training, e.g. relaxation, social skills;
- Reading (self re-education);
- Tape recording of interviews for the patient to replay at home.

Probably the most important CBT strategy is homework. This includes reading, self-help exercises such as thought recording, and experiential activities. Therapy sessions can be seen as 'training sessions', between which the patient tries out and uses what they have learned.

2.1.4 Application scenario

CBT addresses a diversity of disorders such as depression, anxiety, panic, social phobia, bulimia, obsessive compulsive disorder, schizophrenia or post-traumatic stress disorder.

As an example [18] we can describe a scenario where a patient, consider David, feels distressed near hospitals. The patient is at an early stage of the therapy process and assigned to complete an exposure process.

At the end of a session the therapist explains David for him to start going to work gradually closer to the local hospital. David is asked to be attentive to his behavior and heartbeat and remember that step he goes near is a victory. Finally he is asked to fill in a paper form as soon as he leaves the premises. The therapist passes five copies of the form and shortly after the session ends.

Multiple issues emerge from this scenario. From the therapists' point of view, he / she will never be able to confirm how committed the patient is with the treatment by truly filling the paper forms in the desired place and time. Besides, the physical nature of these forms imply storage and searching difficulties when grouped with ones from other assignments/patients. From the patients' point of view, he / she can forget to carry the paper forms or be afraid of filling them publicly. The patient can also miss the therapist support with the positive thoughts that should be present at the exposure time.

2.1.5 Requirements

CBT application domain, as seen above, implies multiple requirements. However, we can summarize them according to their two actors. From the therapists' point of view it is important to track their patients providing meaningful help when it's needed. From the patients' point of view it is important to have appropriate therapist support at the exposure situations.

2.2 Technology in Cognitive Behavioral Therapy

We should begin by mentioning the importance in the use of technology as a way to replace the old paper registry methods. As shown in [11], when compared, the adherence to express the patients' feelings is much higher when they use their mobile phones over the traditional papers. This is due to many factors such as not having to carry additional material through the day, the privacy issues that would expose them when they would write in a paper and the sheer experience of using modern devices.

In order to cope with CBT treatment procedures, the use of technology in homework tasks should offer the options for the person to do some reading, to receive brief examples related to the current assignment that should have been interpreted in another way or to record some data.

For the person who is treating, he / she should also be able to do some teaching about how to dispute and change the person beliefs and to show him / her ways to get started with the homework task. In perfect conditions, this should be customized according to user needs. For example, the use of metaphorical depictions of some concepts is really important, at least when it is designed for young people [13]. Patients respond better to them, when compared to the traditional text based forms, making it even suitable for children with few reading capabilities. However, it is important to notice how, even between children, it is important to distinguish the types of images used, since older children prefer more mature ones.

Therapy applications also have the possibility to be used in order to calm a stressed person [19]. Simple positive thinking and visualizations are compared to three advanced approaches: haptic feedback, games and social networks. In the end, it was noticed that there was no significant stress reduction from using the first and the second methods.

In that sense, the technology use in this type of treatment is actually a positive point as, even the technology phobic patients, after the therapist explains to them how the goal application works, feel more comfortable interacting with it than, for example, talking directly to the therapist [20]. Expressing their feeling to a machine has more accurate results since the patients won't have anyone to judge them (at least at that moment). The

use of the technology also makes it possible to remind them about the therapy assignment, which wouldn't be possible to do with traditional paper forms.

Medical help is also available at home through internet with the e-health applications. The large spectrum of this type of applications made them also possible to target Cognitive Behavioral Therapy cases. While with this there is an advantage to access medical treatment at home, that same advantage may become a liability as people, knowing that there's such possibility, will not leave the house to see real professional medical help and auto-diagnoses themselves [21].

There are also many other technological desktop approaches to cognitive behavioral therapy, such as the Cool Teens CD-ROM, a self-help treatment tool for young people with anxiety problems, where it is provided a CD- ROM with multimedia therapeutic content for teenagers to use at home [22] [23] [24]. Disadvantages of this program are pointed out as being that this program possibly increased adolescents' awareness of their anxieties and the lack of the time by the patients to complete the modules. One of the reasons of this work to be developed is the context awareness of the application. That being said, if the patient is feeling well it makes no sense in reminding him of his fears as pointed by being one of the negative points of Cool Teens. Being a home-based program can also prove to be hard either for the teenagers to find the necessary time needed as having the disadvantage that they aren't expressing their feelings in real time and, therefore, the results won't be the best.

All the presented solutions have one aspect in common: they offer the same help to all their users. They are not created to help a specific person nor are they created with the ability to understand, and therefore adapt themselves to the environment the user is in. Hence none of them verifies previous requirements for our project.

2.2.1 Context-aware systems

Context awareness is paramount in our research, requiring us to review existing literature in the area as well. Context-awareness is a capability which enables systems with the ability to discover and take advantage of contextual information in order to increase the richness of communication in human-computer interaction and make it possible to produce more useful computational services [25] [26]. The word "context" is defined as "the interrelated conditions in which something exists or occurs" in Merriam-Webster's Collegiate Dictionary[2]. Therefore the use of the word "context" tends to be vague because everything in the world happens in a certain context [26].

[2] Merriam-webster: http://www.merriam-webster.com/ (Accessed 10 August 2013)

In one of the first works that introduces the term 'context-aware,' Schilit and Theimer [27] refer to context as location, identities of nearby people and objects, and changes to those objects. In a similar definition, Brown et al. [28] define context as location, identities of the people around the user, the time of day, season, temperature, etc. Ryan et al. [29] define context as the user's location, environment, identity and time. Dey [30] enumerates context as the user's emotional state, focus of attention, location and orientation, date and time, objects, and people in the user's environment. From these definitions one can conclude that context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves [31].

Context information in health applications can be used to improve the quality of healthcare delivery, use the appropriate resources, and to better match the healthcare services to the current medical conditions and needs of the patients. This chapter presents why context awareness is desirable for healthcare through examples of real systems.

The importance of context information when using this type of applications, more specifically the user location, is shown in [32] [33], as the patients liked to create geo-referenced data and the therapists liked to see where the data was captured.

Cisco Context-Aware Healthcare solution enables hospitals to integrate contextual information such as location, temperature and presence information into the clinical workflow to increase staff efficiency, streamline inventory management and improve patient care. Mobile assets and staff can quickly be located for more efficient use of resources and storage conditions can be monitored to reduce waste and improve patient satisfaction[3].

As part of the InSiThe project, this work inherits some of Geo Ties's concepts. This system was made to intensify people connections by allowing them, among other things, to specify areas on a map where they want to monitor for the presence of one of their friends. With these areas they can also specify a series of screen sequences to be displayed in their mobile phones when that friend reaches that area [18] [34].

This system, even though allowing the specification of a series of simple screens, doesn't allow the user to specify different workflows according to the screen answers or context information such as the location of the mobile users. Apart from that it lacks in the aspect of the different mobile screens personalization and creation.

[3] Cisco Context-Aware Healthcare Solution: http://www.cisco.com/web/strategy/healthcare/Context-Aware_for_Healthcare.html (Accessed 10 September 2013)

In the end we can see these systems use different context information to act upon. This access to context information makes a system more intelligent by being able to respond to different situations. In that way, the capability of a system to adapt itself according to such information can provide important help in critical scenarios like the ones in CBT.

With previous findings we conclude that the use of technology in CBT should:

- Allow the creation of personalized mobile applications;
- Use context-based information;
- Target non-programmer users.

Such possibilities should be gathered in a powerful end-programming tool that targets the creation of mobile applications and hides its complexity from non-expert programmers.

2.3 Tools and systems for end-user programming

Solutions for end-programming are designed for a wide range of users that can range from the ones that barely use technology to the ones that do programming their daily lives.

Non-expert programmers find visual languages really intuitive especially because they use an autonomous dataflow representation that can be understood regardless of the implementation language [35].

In this sense, Microsoft PowerPoint, a visual programming presentation tool created in 1990, is able to successfully target every end-user. PowerPoint makes use of a set of components that the user can insert into its working area in order to create content to present to an audience. The slide-based visual presentation allows users to easily create presentation that are shown sequentially by default. Expert users also have the ability to link two non-sequential slides to jump in the presentation. The support provided by this tool is prevalent when looking at talks in research, industry, education, government, and many other areas. Nevertheless, this format has been criticized repeatedly for the limitations it imposes on authors and presenters [36] [37].

Side by side with PowerPoint release, tools like SUEDE [38] and SILK [39] also allowed users to create interfaces without writing a single line of code through visual elements, such as buttons, images and text areas. While the first one was intended to create speech interfaces, the second one was intended to create interface prototypes. In addition, both of them allowed the user to add simple behavior by linking some of the represented elements together. DENIM [40] [41] took sketch-based prototypes a step further by allowing conditional linking between representations, based on some of the

presented elements state. However, this possibility was too limited as only one element state could be verified at the same time, and all the navigational behavior was based on hyperlinks on top of the prototype. UISKEI [42] [43] was then developed to solve this by assigning different possible events to each element.

More recent web based authoring tools[4,5,6,7,8,9] made prototyping more simple by allowing the user to choose from a set of pre-defined components the one they wanted to use in the creation of the prototypes. Some of the tools actually allow the creation of adaptive prototypes that can be tested in a multiplicity of devices sporting, for instance, different screen sizes. Other types of prototyping tools [44] allow the creation of low-fidelity and hi-fidelity prototypes. These may be augmented with advanced features such as time-based rules or events based on delimited interaction areas on a specific prototype.

However, the authoring tools domain does not rely only in the prototyping cases. Ranging more expert users, tools like Adobe Photoshop, Adobe Dreamweaver or Microsoft Visio are an example of that.

Scratch [45] [46] [47] and Unreal Kismet[10] rely on a different approach when it comes to authoring, by having simple script codes in containers that can be dragged and linked. Scratch uses a set of containers in the shape of puzzle pieces, where each one can be linked to the ones that fit. Unreal Kismet uses box containers that have their own output(s) that can be linked to the next container input(s) through a representative arrow. Even though still able to be used by non-programmers, are probably the tools that most stand back from them. Actually, the second tool even allows the user, if he / she is a programmer, to write his / her own piece of programming code in order to use it as well.

Nevertheless, none of the presented tools is related to the desired domain: mobile application creation. In fact, we can say that for this domain we have the designer's view of tools like Microsoft Visual Studio and Eclipse. Since these available views rely on element drag and dropping into a working area, they can actually be considered authoring tools. However, these views are not intended to be used by non-programmers. They still maintain their target programmer users, although simplifying their work, by allowing a visual organization of the screen elements. All the remaining application behavior and element linking (that defines when should a specific screen appear) still has to be defined

[4] Balsamiq Mockups: http://balsamiq.com/products/mockups/ (Accessed 10 August 2013)
[5] Mockingbird: https://gomockingbird.com/ (Accessed 10 August 2013)
[6] MockFlow: http://www.mockflow.com/ (Accessed 10 August 2013)
[7] HotGloo: http://www.hotgloo.com/ (Accessed 10 August 2013)
[8] Invision: http://www.invisionapp.com/ (Accessed 10 August 2013)
[9] Proto.io: http://proto.io/ (Accessed 10 August 2013)
[10] Unreal Kismet: http://www.unrealengine.com/features/kismet/ (Accessed 10 August 2013)

with programming code. Also, these tools typically involve the installation of many components and specification of multiple configuration aspects that are not simple to the common user, such as the definition of a keystore, an application package, etc.

Some recent web tools, which stand close to the goals of this project, are Codiqa and AppArchitect. Microsoft had also released their own authoring tool, entitled Windows Phone App Studio. All these tools rely in the same principle when creating mobile applications: element linking and drag and drop. Nevertheless, after trying these platforms we realized that they are aimed at static applications that do not change their content based on the user or context information. They are therefore intended to create informative applications.

Improving these last applications, the available beta tool of the MIT App Inventor for Android, released in late 2010, also claims to be usable by non-programmers in order to create mobile applications for the Android platform with access to contextual information. The platform also works by having a set of screen elements, such as buttons and text, which the user can drag and drop in the screen itself in order to create it. The specification of the application behavior (when to link screens and based on what elements), that requires the user to install a specific software plugin, works similar to the Scratch language by having small actions in linkable boxes. Nevertheless, the appropriation of the tool to our project falls apart as the user, who was intended to not have programming skills, also has to define application packages, activities, activities results and classes, typical concepts related to specific Android programming. Apart from these required fields, the type of language used when referring to elements actions is also still too programming oriented and not easily understandable by a non-skilled user. For users that are not too comfortable dealing with computers, this interaction level is still too detailed to make the tool usable by everyone and therefore it is necessary to make it even simpler, where almost no specification is required.

As seen, none of the above available tools can aggregate all the functionalities that our final tool should offer:

Tool	Mobile Application Creation	Context aware behavior	Non-Programmers
PowerPoint	No	No	Yes
SUEDE	No	No	Yes
SILK	No	No	Yes
DENIM	No	No	Yes
UISKEI	No	No	Yes
Balsamiq Mockups	No	No	Yes
Mockingbird	No	No	Yes
MockFlow	No	No	Yes
HotGloo	No	No	Yes
Invision	No	No	Yes
Proto.io	No	No	Yes
Mixed-Fidelity	No	No	Yes
Adobe Photoshop	No	No	Yes
Adobe Dreamweaver	No	Yes	No
Microsoft Visio	No	No	Yes
Scratch	No	No	Yes
Kismet	No	No	Yes
Eclipse Design View	Yes	Yes	No
Visual Studio Design View	Yes	Yes	No
Windows Phone App Studio	Yes	No	Yes
Codiqa	Yes	No	Yes
AppArchitect	Yes	No	Yes
MIT App Inventor for Android	Yes	Yes	No

Table 1 - Presented visual programming authoring tools summary

2.4 Summary

In this chapter we have presented some of the solutions that use the components our final product will require.

We started by explaining what Cognitive Behavioral Therapy is and how it is used with nowadays technology. Afterwards we have explored some systems that handle context-based information in order to perceive their uses. This included both the systems that are designed to be used at buildings as the ones that are designed to mobile users. Finally we have presented some of the authoring tools that are available in the market. Even though the domain these tools focus on is quite wide, we realized that none possessed the requirements we are looking for in this project.

Nevertheless, the studied systems allowed us to perceive some of our project requirements that should be verified.

Chapter 3

Understanding Users' Programming Concepts

In this research, we aimed at understanding how these users cope with specific design elements for a mobile application. We particularly emphasize the way they represented the flow of an application, taking into account potential conditional transitions and environment variables which may influence the behavior of the application. With the outcome of a series of participatory design sessions we understood how non-expert programmers deal with programming elements. Based on these results we wanted to build DETACH first prototype.

3.1 Experimental Trials

In order to identify the main design elements of an authoring tool and, especially, how users would interact with such application, we envisaged a two stage participatory design process (particularly suited for scenarios like this one [48] [49]) in which non-programmer users would be able to create an application based on specific scenarios using low-fidelity components (e.g. post-its, paper and pencil based drawings). We were especially interested in two aspects:

- Which programming elements are crucial for the creation of a mobile application according to non-expert programmers;
- How non-expert programmers organize the different components of prototype to express the flow of the application, having in mind that conditional transitions and environment variables may influence its behavior.

Our participatory design sessions took place on the participants' working places, in order for them to feel comfortable to express their ideas in a more natural way. It comprised professionals from different domains, ranging from psychotherapy to general clinic to the teaching and arts area.

Every session was carried out with one and only one person at a time, in order to avoid any bias from multiple subjects sharing ideas simultaneously and was supervised by an expert UI researcher.

3.1.1 Participants

In these sessions we tried to recruit a moderately large number of health professionals as well as people with no programming background, so that the results could be as helpful as possible.

In the end we had 28 users (12 male and 16 female), with an average age of 43 years old, being 20 of them from the health domain. A detailed description of the participants is annexed to this document.

3.1.2 Tools & Equipment

The following common material was given to the users:

- White sheets of paper (representing the authoring tool environment);
- A pencil;
- An eraser;
- A pen;
- Some empty post-its.

In order for us to study how participants would deal with the provided material, some of the sessions had available an additional guidance text of a specific application for them to represent (annexed to this document) and/or some template mockups (Figure 1):

- 4 distinct post-its representing potential mobile screens:
  - One displaying a list of emotions the user would need to select from;
  - One presenting a simple message;
  - One displaying a question with two possible answers;
  - One presenting an animation.
- Distinct accessible environment variables such as:
  - User location;
  - Time;
  - Health monitors.

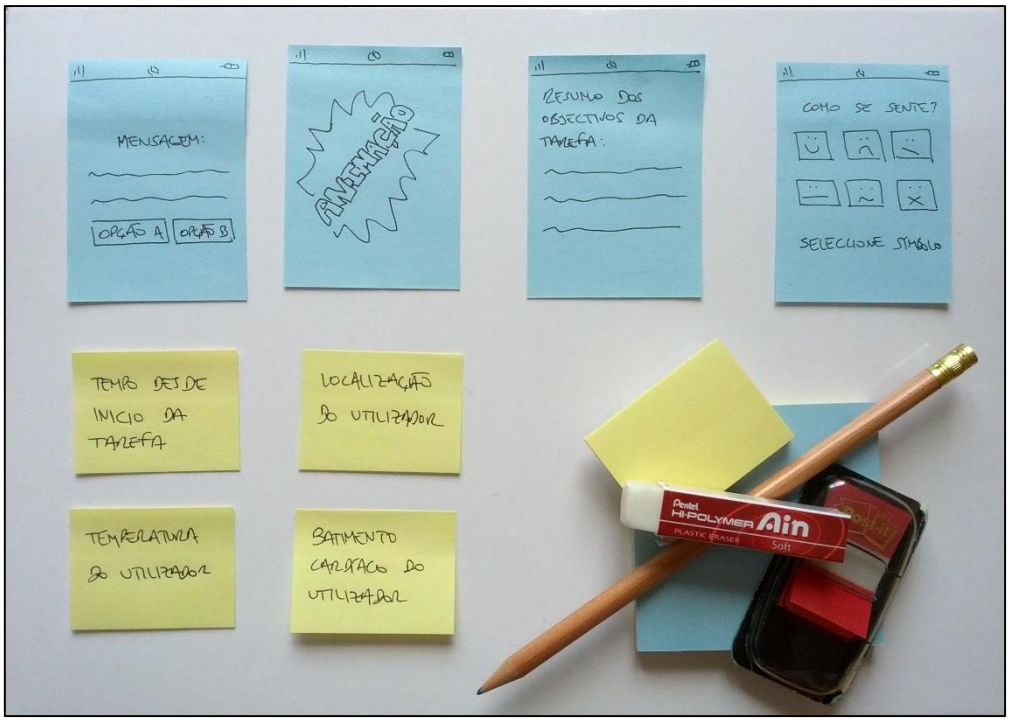

Figure 1 - Participatory Design Tools & Equipment

3.1.3 Procedure

In these sessions some instructions were given to the users. They were asked to role-play as therapists that were configuring an application that was running on the patients' smartphone, when he / she was outside the session doing a homework assignment. The white paper represented the therapist screen, and they could draw/write whatever they wanted in order for them to better represent an application.

For the participants outside the health area, we provided them some screen mockups and a specific story text with a screen sequence they were asked to create. This story intended the participant to use all the provided screen mockups and connect them based on the user answers and provided environment variables.

For the remaining participants we completely omitted the guiding exercise, giving them freedom to arrange the screens to form an application that made sense in their specific clinical case. The idea was the users to think of a real therapy assignment, where the order of the things was not implicit. Therefore, these sessions made sense to be done with health professionals only, since they could think of a real clinical case. They were asked to use as many screens, environment variables and patient screen choices they could, where in some of the sessions no mockups were provided.

No clues were given to the users about which methods they could use or how to describe the application behavior (e.g. no indication was given whether they could opt to create visual depictions or textual description about the application's flow). Every detail

regarding each material the users had available to compose the application during the exercise was explained. Subjects were also explained about the possible environment variables they could use in the application.

3.1.4 Results

The first noticeable behavior was that some clinicians, being inspired by the mock-ups we handed them to create their own versions of each screen (Figure 2 - middle). They evoked reasons pertaining to their own clinical cases and personal experience to justify the changes performed on the UI.

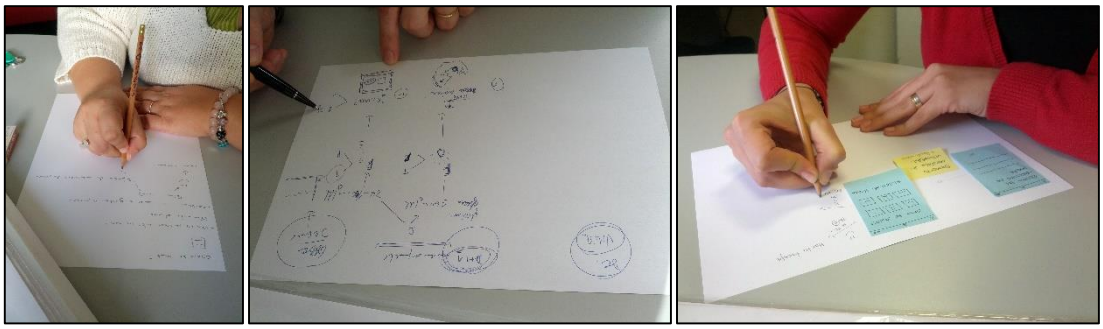

Figure 2 - Participatory Design Example Results

As far as screen organization is concerned, 7 users adopted a textual description strategy (Figure 2 - left), while 4 of the users approached the flow organization sequentially, by aligning screens side by side (Figure 2 - right). All the remaining users used a variety of techniques revolving around the concept of arrows connecting each screen with footnotes or post-its indicating the conditions upon which the user transits to another screen.

We also verified that most of the users replaced the environment variables post-its with logical symbols representing each one (a heart, a clock…), perhaps to same space in the drawing or to be more explicit.

Concerning these complementary post-its representing external variables, most users actually capitalized on them to represent continuous heartbeat rate retrieval – they added a footnote on the canvas containing these post-its and what they represented. 3 users adopted a different approach and actually created a few more post-its representing the same variables handed to them and placed them attached to each screen in which they considered relevant to retrieve these variables, creating richer and even more complex applications.

The most surprising new element introduced here was the addition of a loop programming element (represented by a circular arrow between the affected screens).

Despite having no programming background, two of our subjects decided to include a loop function between two screens in order for a potential user to indicate his / her emotional state and then be confronted with a question related to his / her pathology until a positive emotion was indicated. This programming element emerged as a surprise to us, effectively emphasizing not only the complexity users can come up with during design sessions but also the rich requirements these clinical apps have.

Another behavior verified with some of the users was the initial specification of different intensity approaches the assignment could have. The idea was that the application would chose the appropriate task according to user answers or times he / she took to execute specific tasks. If the user was spending more than the predicted time doing a specific task, an easier one would come in place.

One of the health professionals really liked the fact that the application that was being created would be able to react differently to answers or context information, suggesting that the application could take the appropriate measures when it was needed (like calling the emergency services by itself).

Some participants also designed their applications in order that, besides the normal execution, simple messages were displayed once and a while, just to reduce the loneliness feeling a patient could have during his / her normal life.

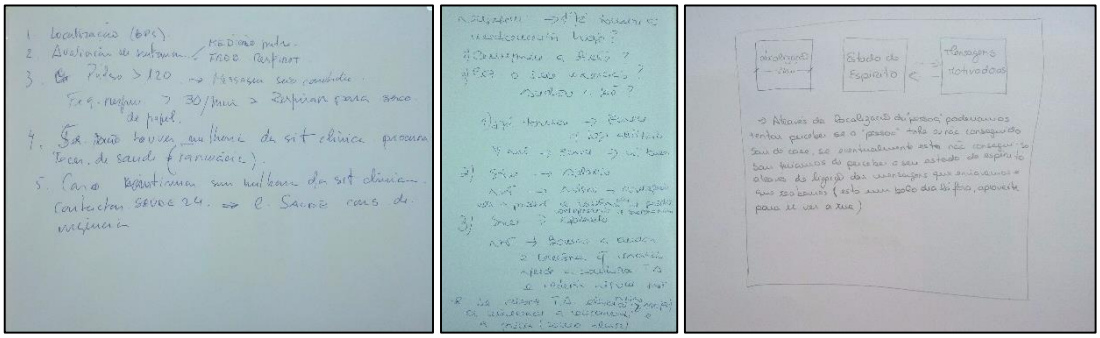

Figure 3 - Participatory Design Example Results

A remarkable result in these sessions was the diminished sense of creativity we sensed from the participants that were merely translated the guiding script into the interface and from those that had no additional mockups available. The last ones represented their applications in bulleted natural language text (Figure 3). Besides, a lot of help was needed in order for users to understand, even with the initial explanation, what they could do after each screen.

We also verified that some of the participants decided to draw the actual authoring tool interface, where we realized that they reserved areas, represented by separated boxes/clouds/balloons where they could get pre-defined elements from to use in the

specified application (Figure 3 - right). This in fact makes the use of mockups even more important.

These sessions took on average 30 minutes per subject.

3.2 Conclusions

In total we ended up with 28 different representations. In general, these experiments allowed us to identify a few key aspects in the way non-expert programmers deal with authoring tools, with the following common results:

- Most subjects prefer a free canvas in which they can position the application's elements;
- Even with no previous programming experience, subjects were not afraid to add complex routines such as AND/OR statements or loops, even though no reference to them was made in the initial explanation;
- Subjects adopted a top-down perspective, beginning by defining the building blocks of the application, and only after pursued the detailed behavior associated with each element;
- The majority of the users just preferred to draw the connections between screens after all of those screens were already place in the working space;
- The similar representation of screen mockups and environment variables confused the users in a way that the last ones were also used as screens mockups, with no distinction in the use;
- New screens were suggested. Interestingly some of them addressing physiological data retrieval. More importantly a new type of transition and condition was requested by some (more active) participants, featuring an interruption of the normal flow.

3.2.1 Connection Strategies

People use different techniques to represent the logic behind the mobile application flow. In fact, the amount of used techniques makes this a critical functionality. Most of the people tend to add arrows, often complemented with notes stating transition rules, to represent the navigational flow. Nevertheless, some participants preferred to organize screens on a sequential fashion, touching each other to pinpoint the transitions. Different workflows were addressed by separating one of the screens slightly from the main group and annotating the rules associated with that transition. The remaining participants used a mix of the previous techniques, combining bulleted text or aligned post-its with some arrows indicating jump transitions (Figure 4).

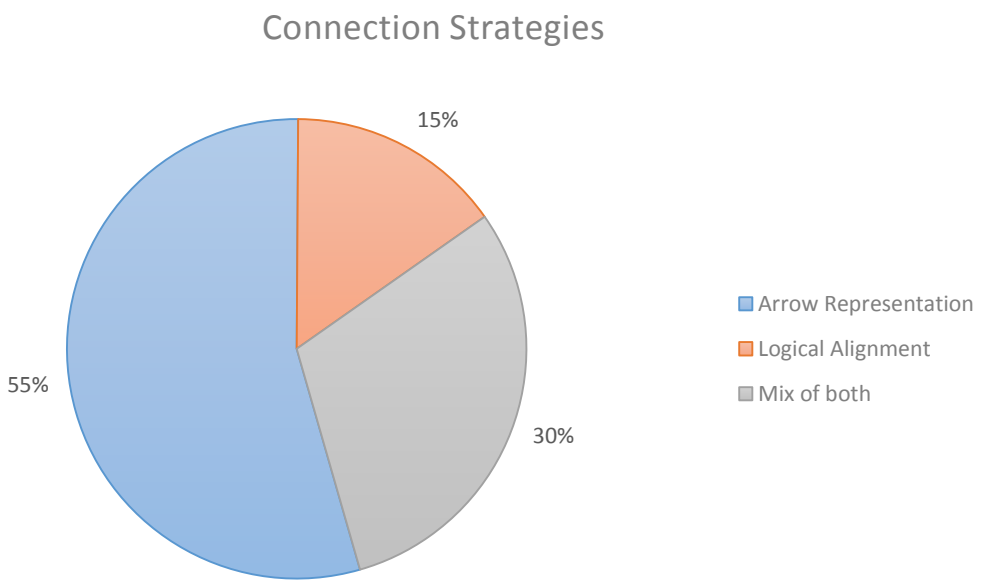

Figure 4 - Users connection strategies

3.3 Summary

This chapter described a series of design exercises with non-expert programmers in an attempt to gather empirical data about how they design applications.

There is a major lesson we gathered from this experimental period: UI experts should give more power to the users. From the observations of these experiments, creativity appeared to be hindered when users had fewer tools (e.g. screen post-its, external variables, etc.) at their disposal. Even with a thorough briefing regarding device capabilities and which environment variables a designer can capitalize on, users failed to explore the full capabilities of a potential target device. By presenting them with physical representations of potential environment variables, features and data streams that may be used with an application, users promptly came up with more complex prototypes, aiding us in better understanding the needs of expert clinicians and, in particular, psychotherapists in their ventures.

For our future authoring application, we then identified a selection of key elements which will definitely be included in the final design:

- A toolbox containing default designs for typical screens used in psychotherapy;
- Restrict the nature of editing in each screen, thus focusing on the main functionalities provided by each one;
- Sequential or behavioral navigation through screens.

Chapter 4

Understanding Users' Interactions Patterns

Based on stakeholders feedback, we designed a high-fidelity prototype for the DETACH tool. We conducted a series of thinking aloud and usability tests [81] [82] during the design process in order to reach a final successful product. This trial was performed either by some of the participants who had been present in the previous experimental period as well as new non-programmer users that had volunteered themselves to participate. When running the trial we felt the need to improve the used prototype after we observe the first results. In the end, and after using both prototypes, we could perceive some user patterns worth mentioning when people are handling this type of tool.

4.1 Hi-Fi Prototypes

The first DETACH hi-fi prototype was achieved through a set of design revisions that had in common key elements we found necessary from previous participatory design sessions. As the presence of the screens mockups in previous sessions was critical, all of the revisions contained a set of template screens the person could use. These template screens were built based in our previous trials, where we felt the need to have:

a) A screen that would only show some content to the user;
b) A screen that would ask a question to the user;
c) A screen that would allow a user to set his emotional feelings with representative images;
d) A screen that would allow a user to provide text based answers, where images, sound and video could also be referenced;
e) A screen that presented an animation.

In order to maintain the simplicity and similarity as people were dealing with real post-its, these screens just needed to be dragged to the working area (replacing the previously used white sheets of paper).

Between the different revisions we've tried to concretize distinct interaction possibilities in order to reach a final one that could be tested by real users.

4.1.1 Initial Design

The first design created worked similarly to Microsoft PowerPoint tool by offering three distinct areas: a set of template screens available to use on the top left; a rightmost empty panel; a bottom timeline with a sequence the used template screens (Figure 5).

In order to add a new screen to the application, the user just had to drag one of the top left screen templates and drop it over the rightmost panel. Each dropped screen would appear at the end of the bottom timeline. This timeline allowed the re-organization of the used template screens in order to create a logical sequence to be executed after each screen button press.

This initial design was too limited as the user could just define basic screen sequences, which would always be presented by the same order. Therefore, it was not possible to represent different workflows during the application, a requirement that was verified with the previous trials.

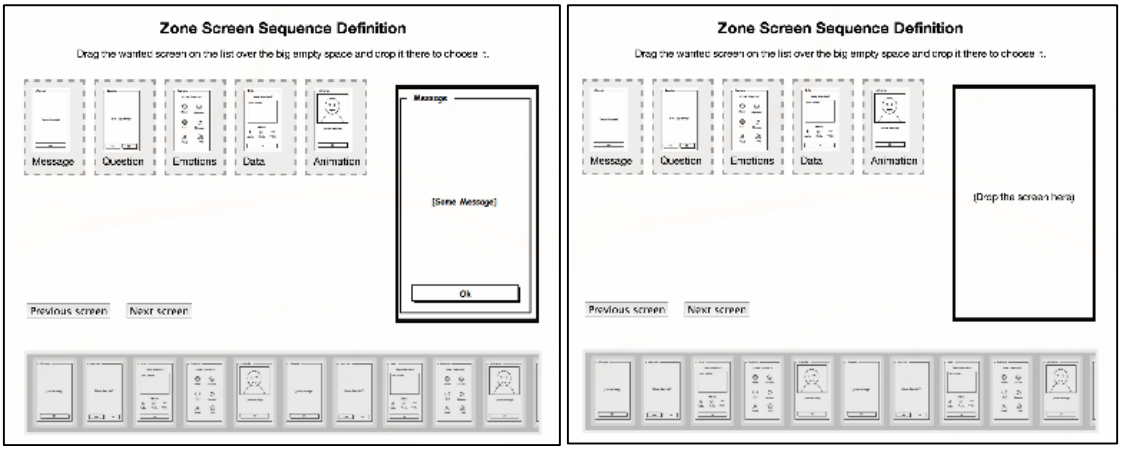

Figure 5 - DETACH initial design

4.1.2 Simple and Expert Mode Design Revision

The previous design had the simplicity we wanted to be verified in our tool. It allowed users to easily add and configure each screen they wanted to display. However, it was hard to create different screen flows with the presented timeline. Therefore a new design emerged that improved this interface and divided it in two views: the simple one, where the user could add and personalize each screen (Figure 6 - left); the expert one, where the user could have an overview of the created screens in order to organize them and define how would they transit to each other (Figure 6 - right). In this view we also changed the way users could organize screens, giving them the freedom to place them in an infinite canvas. Screen transitions would depend on previous screen answers or data that could be monitored constantly, such as user location and time.

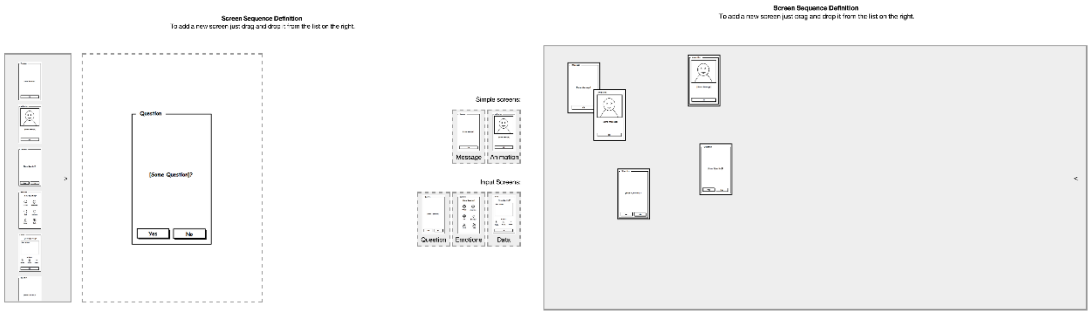

Figure 6 - Simple (left) and expert view (right) of DETACH initial prototype, second design revision

4.1.3 Final Prototype

In the previous design we presented two distinct views for people to use. In the simple one people could add new screens, organize and define their contents. In the overview one people could define how would a screen connect to others. Nevertheless, no connections could be defined in the simple view, nor a screen could be added or filled in the global view. This need for a user to switch between views to reach functionalities made us create a prototype that united both in a way all the functionalities were present in one only view (Figure 7).

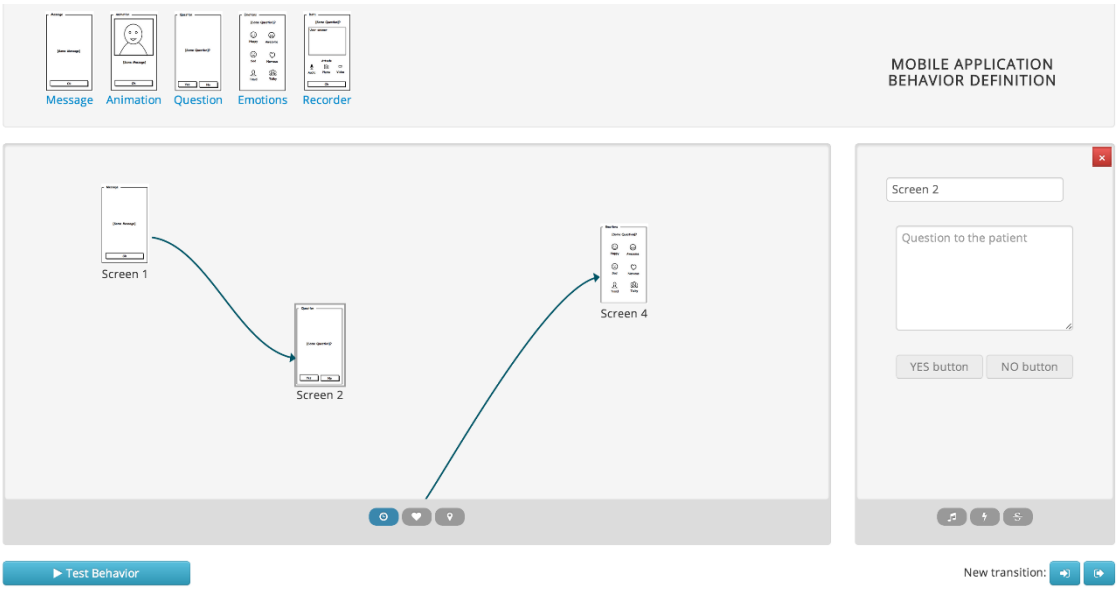

Figure 7 - Presented prototype representing connections based on screens and connections based on environment variables

The result is a web application that offers a workspace on which users may populate screens based on different templates and connect them according to a set of rules whose complexity may vary (e.g. from basic screen sequence to transitions based on previous patient answers). The top section displays the available screen templates (e.g. multiple choice answer, animation display, etc.). Users can drag a template into the center canvas

and configure each screen's particular elements on the rightmost panel. This could include screen content, title, animation type, etc.

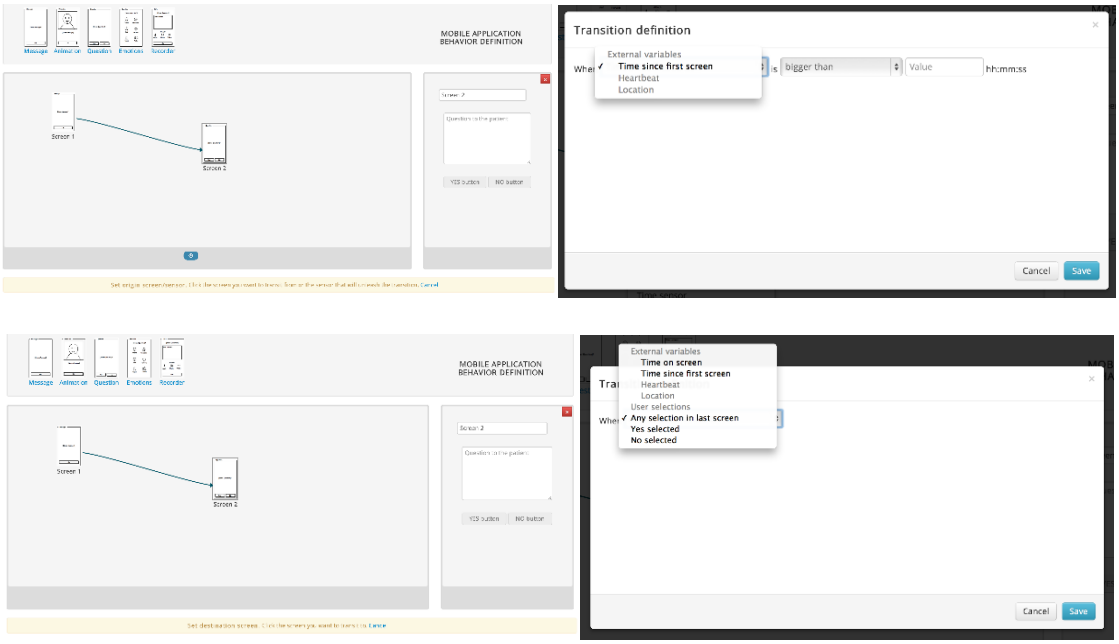

Figure 8 - Top: Presented prototype left connection button and environment variable click result; Bottom: presented prototype right connection button and screen click result

Transitions between screens (represented by arrows) can depend on:

a) Screen outputs, where each screen provides its own triggers. In this prototype, all the created triggers depended on specific elements that were present on each screen. For example, the message screen, that faced an OK button at the end, would allow to create transitions based on that OK button click. On the other hand, the question screen, that allows a user to answer clicking the Yes or No button, allowed a user to create transitions based on each of this button clicks;
b) Environment variables, where we included some data values that are not associated to a specific screen but can still be monitored to create transitions, such as time and GPS values.

These transitions are defined by selecting a screen, hitting one of the lower right buttons, selecting another screen and finally stipulating the rules associated (Figure 8). These buttons actually differed in their functionalities: the left one considered the selected screen as the destination, and presented a tooltip asking to select the source, where the source could be a screen or an environment variable (in case the user previously activated it in the application), if a user wanted to monitor those values permanently, independently of the screens; the right one considered the selected screen as source and presented another tooltip asking to select the destination one, that could only be a screen. The resulting arrow source would therefore be the selected screen or environment variable

representation (Figure 7). In this prototype we also wanted to make the interactions easier by automatically create these arrows (transitions) between two consecutively created screens. The conditions associated with these last ones would be the source screen first button press (OK in case of a message screen or YES in case of a question one). Nevertheless all the conditions associated with an arrow could be changed by clicking on it.

As this was not the final DETACH result, but instead a prototype for user tests only, some functionalities were left unaddressed, such as user management, project saving and application styling and testing. However, a user could already add new screens to an application and configure their text contents as well as define the condition that would make the screen to be displayed. Based on these functionalities we built this prototype to be able to run locally on top of HTML5, Javascript and CSS3 programming languages.

4.2 Experimental Trials

With the DETACH initial functional prototype we've conducted a series of thinking aloud trials with non-programmer users. These trials occurred roughly 3 months after the low-fi ones.

4.2.1 Goals

Our set of thinking aloud sessions were intended to:

a) Identify non-expert programmers' interaction patterns with this type of tool, comparing them to those verified in the low-fidelity prototypes;
b) Test DETACH prototype usability, verifying if the chosen design and functionalities were easily perceived by our participants.

4.2.2 Participants

11 non-expert programmers (4 male and 7 female), with an average of 34 years old have participated in this sessions. 4 of them had already participated in the previous participatory design sessions and 4 of them belonged to the health domain. All of the participants were comfortable with both Portuguese and English languages and therefore able to correctly use our prototype. A detailed description of the participants is annexed to this document.

4.2.3 Tools & Material

These trials were carried out with the previously described prototype presented to the participants in a laptop (HP Pavilion dv9890ep model) with an external mouse attached.

4.2.4 Procedure

Akin to the previous participatory design sessions, participants were asked to represent a specific application intended to help a patient in a therapeutic procedure. In order to do so, we presented them a fictitious scenario with the story they were going to be involved in (annexed to this document). Participants were asked to complete 11 sub-tasks where in each one they had to focus themselves in different application functionalities.

The presented script began by describing a scenario of a 26 years old Portuguese girl who wanted to work in the USA but was afraid to travel by plane. As her therapist, each user had to specify a series of actions to be executed by a mobile application. This application would be carried in the girl's smartphone behaving itself like a virtual therapist.

Each session took place with a user at a time, in order to avoid any idea sharing between them. To provide the most pleasant and relaxed experience to our subjects, testing occurred in the participants' working places.

4.2.5 Results

No user was able to complete all the tasks without any help in at least two of them (see annexed task completion times).

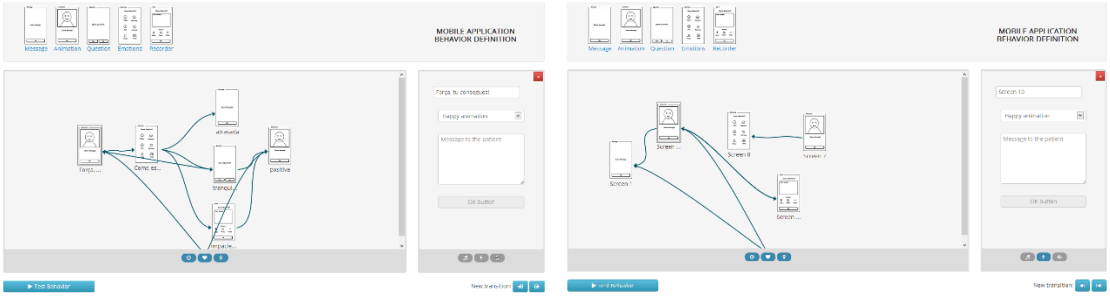

Figure 9 - Thinking aloud example results

The main problem verified was related to the creation of connections between two screens or a monitored data value and a screen. Just 2 of the users completed this task correctly. The intended action could be accomplished by clicking the bottom right buttons, where the words "New transition" can be seen. These buttons, although designed to create different types of connections, proved to not be understood. As a consequence of their location and similarity, participants:

- Did not notice the buttons;
- Did not perceive the buttons were intended to create connections and therefore did not even try to click them;

- Clicked the buttons to iterate through the various screens they had already defined. As such, the left button would show the previous defined screen while the right one would show the next one;
- Clicked the buttons to create new screens, to the left or to the right of the current one;
- Clicked the buttons randomly to try to connect the screens.
- Used them spatially, clicking in the right one if they wanted to connect the current screen to some screen in the right or the left one if they wanted to connect that screen to some in the left of the current screen (when to achieve this the right button should have been used both times);

We also verified that the users did not even read the help tooltips that appeared covering the buttons, when they were clicked. The resulting situations were diverse:

- Not being able to complete the current task as the clicked button did not provide the desired functionality (as the left button was able to create connections from the environment variables to the current screen and the right one was not);
- Being able to create the connection but the reverse way (when the wrong button was used to connect screens).

After the failed cases, the two buttons functions were explained in order for the users to try to complete the next steps. However, most of them continued to make no distinction on their use and when they were asked to describe their differences the users replied that they still did not know, even after reading the tooltips.

Even after the previous explanation, we verified that most of the times participants were expecting that both connection buttons should be used with the origin screen selected so they could define the destination one. This was particularly noticed as participants would prefer to go back to the origin screen in order to set the destination one. The left connection button, even after some explanation, proved to be useless and confusing, as people were in fact using it to also create the same type of connections as the right one. Some users suggested having descriptions in each connection buttons itself.

Another confusing aspect, also relative to the connections, was the need to previously activate possible monitored external data before its use. Participants complained that, being possible to use it in the connections, it made no sense in having to activate it first somewhere else.

After the connections were created, and during the task where they had to redefine the conditions associated with a specific one, we also observed that users did not perceive

that connections were clickable, instead, they tried to define a new one with different conditions, to override that one.

In the end, users spent an average of 28 minutes and 21 seconds trying to complete the whole script.

4.2.6 Improved Prototype

After the observed results with the previous prototype, we felt the need to reflect on people's interactions with the tool in order to continue our evaluations.

The critical functionality that was not perceived in the previous prototype was the connections between application elements. Since this is one of the main tool functionalities, in this phase we tried to make it more intuitive.

According to previous findings, the new prototype was improved with the following results:

- Since no distinction was made by participants when using the transition buttons, they were merged into one, made more visible and with an actual description. This button would in fact behave like the previous right connection button, where users from the source screen would choose the destination one;
- As users did not perceive that the connections between the screens were actually clickable and editable, their representation was changed in order to appear to have some content;
- Users did not perceive which screen was selected at each phase, so this selection was reinforced to what they are used to when selecting actual files in their computers - having a blue layer over the selection;
- The environment variables activation is now present when defining a new rule, since in the initial applications users were confused by having to use some connections in a way (the ones based on screen variables), and the ones based on environment variables in another, having to previously activate them in the application;
- Help tooltips were also included after observing that some users were afraid to click things they didn't know their purpose;
- Users confused screen names with notes, not knowing if that name was going to be visible in the end, so we felt the need to include a new screen field containing information only visible to the person who is creating the application (Screen Notes);
- The helping messages were also moved to the top of the screen since most of the users did not even perceived they were there;

- The automatic connection representation between consequent screens was also removed since it was confusing users, making them erase it in most of the cases.

A new type of interaction was also introduced, as some of the users liked to define transitions that would be applied to all screens as a new transitions created manually from each screen. Therefore, the previous option to define that was removed and replaced by the possibility to define a new connection, the normal way they are used to in a single screen, but having that option when multiple ones are selected. This type of connection would be represented with a softer arrow in a way it interfered less with the remaining application representation. For this functionality to be possible, we also provided selection checkboxes in each screen as well as a button to select all the available screens (Figure 10).

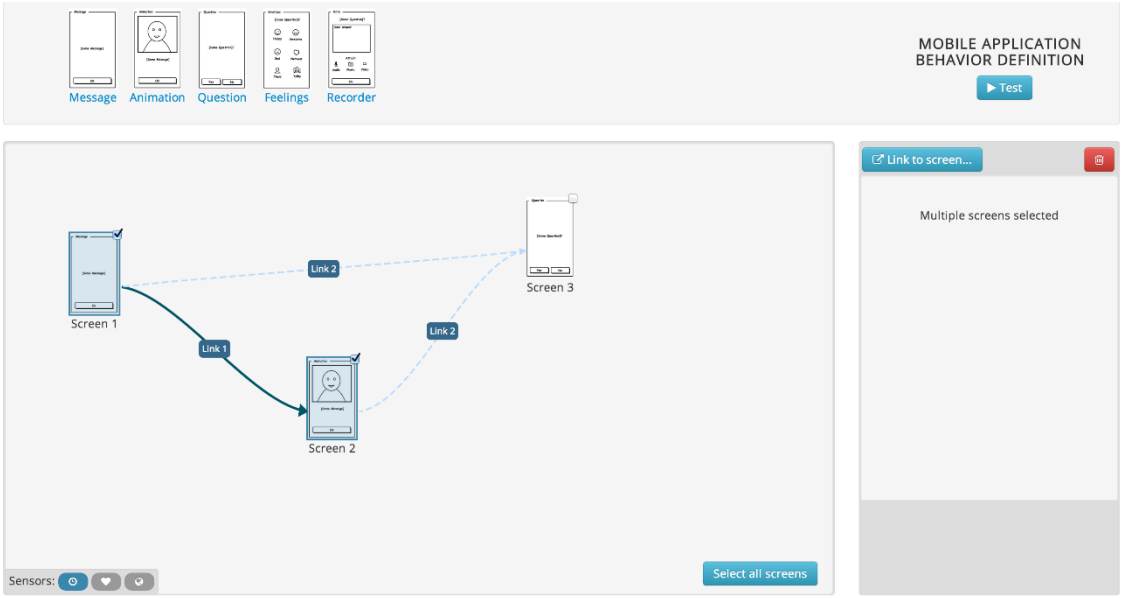

Figure 10 - Improved prototype representing connections based on screens (Link 1) and connections based on environment variables (Link 2)

4.2.7 Experience

7 non-programmer users (3 male and 4 female), with an average of 26 years old tested this new prototype. One of them had already participated in the initial participatory design sessions. A detailed description of the participants is annexed to this document.

The same script was given to users. All of them were comfortable with both Portuguese and English languages and therefore able to correctly use our prototype.

4.2.8 Final Results

With the new prototype we could observe several improvements. 4 out of the 7 users (~57%) could complete all the script steps without any help (where so far no user could

do it). If we look at the time spent to complete the whole script we can also see that they reduced about half the time (from 28 minutes and 21 seconds to 16 minutes) when comparing to the previous prototype (see annexed task times).

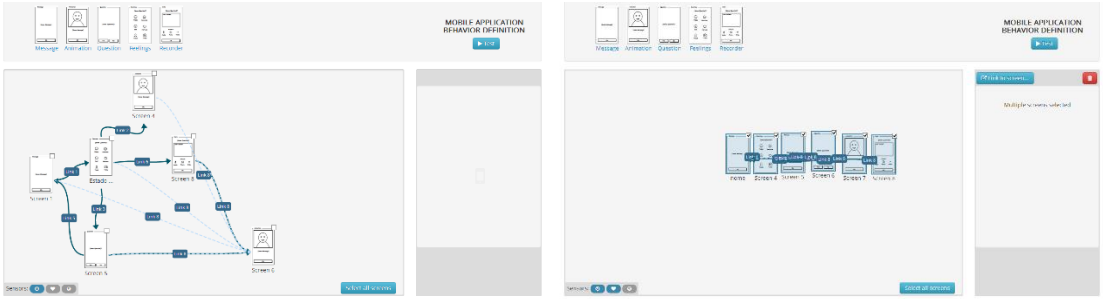

Figure 11 - Thinking aloud with the improved prototype example results

Nevertheless, we still saw some space for improvements between this prototype and the final product as:

- Users found it weird to have the arrows that linked specific and multiple screens overlapped;
- Some users would have preferred the difference between the two types of connections to be reduced.
- Users suggested that the connection button name and placement location could be more evident.
- When asked to create a connection that was based on environment variables, the fact that there was an actual representation of them in the application confused users when we asked to connect screens based on them.

4.3 Conclusions

In the end our hi-fidelity DETACH prototypes were presented to 18 participants. This allowed us to identify some key user patterns as well as how our final DETACH product should evolve from the used prototypes.

4.3.1 User patterns

During the sessions with our functional prototypes we registered the patterns displayed by users when organizing elements in a free canvas as well as the approaches taken to connect them.

Organization strategies

One of our concerns when testing these prototypes was how users would prefer to organize elements in a free canvas working space.

After the results of our 18 trials we could verify that subjects adhered to three main strategies (Figure 12):

a) Akin to the low-fi trial, 3 users positioned the screens sequentially, typically reflecting creation order (Figure 11 – right);

b) 5 users delineated imaginary columns upon which screens were positioned (Figure 9 - left). If a screen generates two navigational branches (i.e. users can transition towards two different screens from the same origin) then all destination screens would fit inside the same virtual column in the canvas, with the branches collapsing into another screen if that was the case.

c) The remaining 10 participants did not show any particular spatial strategy here, placing their screens feely in the canvas (Figure 11 - left).

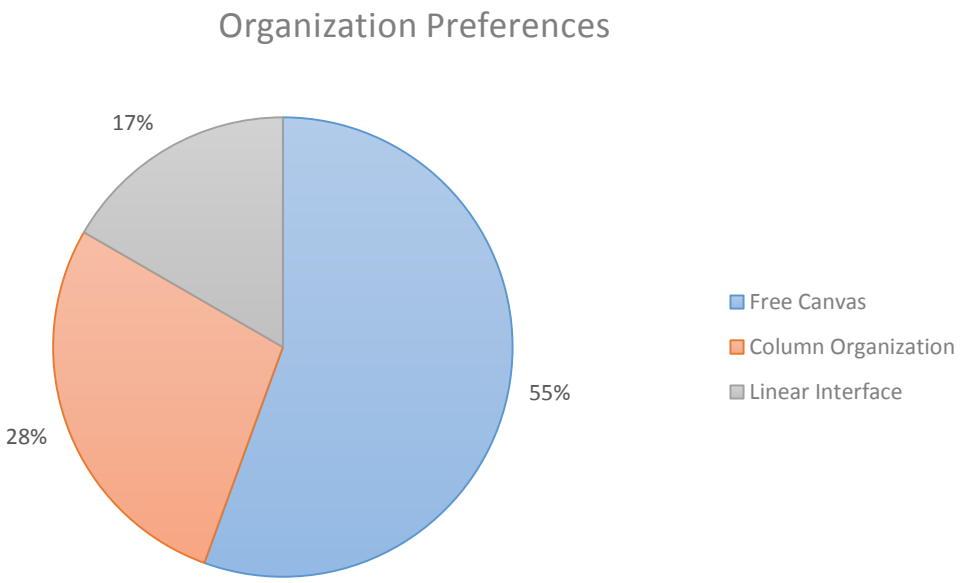

Figure 12 - User organization preferences

When comparing to the initial simple prototype (Figure 6) and with the Microsoft PowerPoint slide organization, we did not feel that the use of a free canvas increases the tool's complexity. In fact, this actually provides multiple possibilities as some users, although finishing the application with the screens spatially organized in the canvas, had organized them side by side till reaching a screen that would create more than one branch. People are perfectly comfortable with this type of element organization as they can opt for the organization they most preferred.

Connection strategies

Despite the existence of several endeavors to create authoring tools for a variety of purposes, some design aspects were left unaddressed by researchers and IT experts alike. Our research also focuses on one of these issues: while sequential transitions between screens of a mobile application may be sufficient for some domains of intervention (e.g. mobile app prototyping [44] [38] [39]), other domains possess a more critical nature which requires richer and more complex connections. Throughout our session periods with a total of 18 users with no training in end-user programming we strived to analyze and identify the strategies adopted to intertwine screens in a mobile application authoring environment.

These strategies were identified as the ones participants tried to use to connect the screens, and not necessarily the strategies our authoring application was offering. As result, we could verify that many of our participants tried to use more than one strategy to reach the connection functionality. For the participants that started by using the strategy actually offered by our tool, that strategy was accounted as the user preferred one, since no other was tried before.

Touch to Connect: One of the approaches observed rooted itself in the low-fi version of the prototype. 4 participants aggregated screens which had a sequential transition nature together in the canvas. A particular behavior was noted: akin to the low-fidelity prototype, where subjects often organized the screen post-it in a way resembling a deck of cards, with each screen slightly touching each other it had a connection with, on the hi-fi prototype, users attempted to link screens using a different strategy: they dragged one screen towards another, "touching" it. The expectation was that a new transition was established between the "touched" screen and the dragged one. This strategy clearly shows a sequence oriented thinking towards building mobile applications. When confronted with the possibility of adding additional rules for these transitions (e.g. based on patient inserted content / answers) they argued that this still felt like the most natural way to interact with the elements in the canvas.

Origin-Destination Paradigm: The most popular strategy adopted by users was inspired by the way they typically fill-in a postcard, an e-mail or a letter: they define the origin of the connection, the destination and then any related content with them. 13 of our subjects adopted this approach arguing "this is the way I naturally write" and "the way I did on the paper version", for the ones who had already participated in the first sessions. One may ask if using some of the same subjects and the previous experience with the low-fi prototype could influence this result: in part, we agree, but we must also note that a substantial number of participants did not follow the same connection strategies; also

the timespan between both trials dissipates some of the "training" acquired in the first trial.

A minority of the participants (3) approached this paradigm by completely switching the connection's order definition: they started by selecting the destination screen and then they picked the screens which would transit to the former. When asked to verbalize why they adopted this strategy, they argued "it made sense, considering a patient can reach the same screen from different branches", so "defining the destination first felt straightforward". Here, we must state such decision may have been influenced in part by the mobile application they were asked to create, since it featured a screen which could be reached from multiple navigational branches.

Connection from Screen Elements: When selecting and configuring each screen, 11 participants attempted to generate connections from the screen's components themselves (e.g. each answer, a button, etc.), justifying their behavior stating "the patient will transition to another screen if he / she presses this button". Even though no participant had previous programming experience, this is an approach reminiscent of existing Integrated Development Environments (IDE) such as Microsoft Visual Studio or Eclipse. In these tools, users may click a component, such as a button, to configure the application's behavior when the button is pressed. It is interesting that despite the absence of experience, some participants actually prefer this strategy.

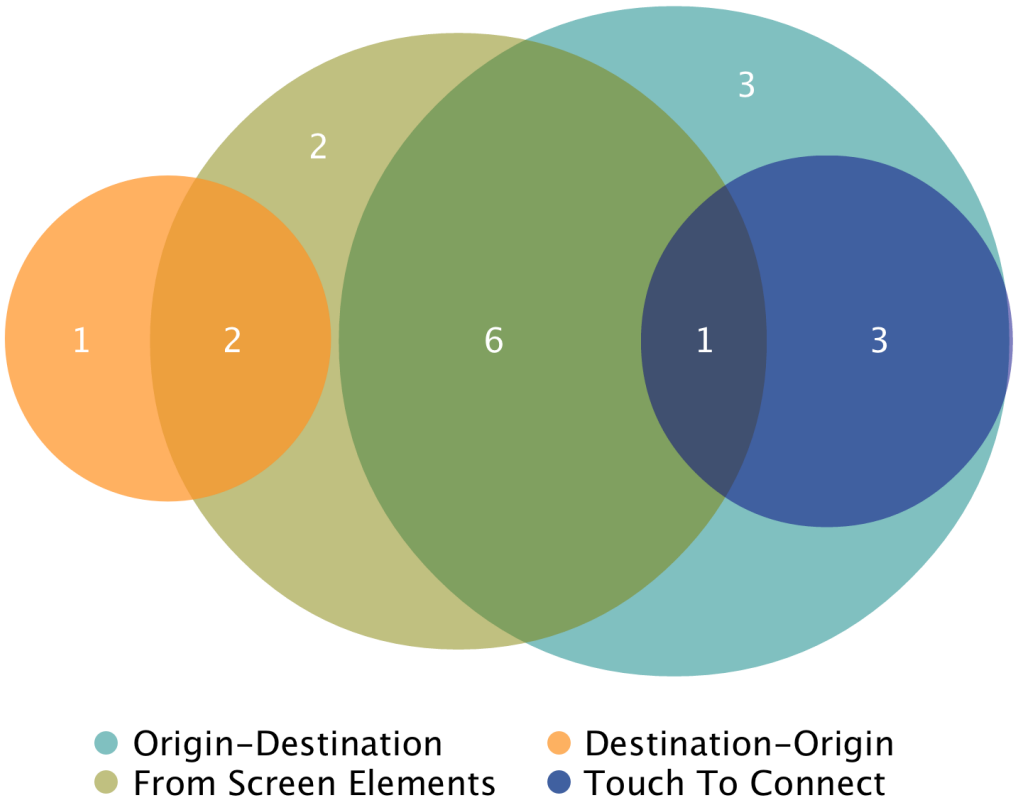

Figure 13 - Connection strategies used per participant

With these results we could also observe that participants using the Destination-Origin paradigm also tried to connect screens from the user answers but not from putting them together (touch to connect). On the other hand, all the participants that used touch to connect also used the origin-destination paradigm (Figure 13).

4.4 Summary

In this chapter we describe a series of participatory design sessions with non-expert programmers and high-fidelity DETACH prototype. The main findings from these sessions pertain to the variety of approaches participants were able to adopt to accomplish the same goal. Nevertheless we must prioritize the strategies which gathered more followers. As such, DETACH's primary approach towards the definition of screen connections will follow the origin-destination paradigm and the free canvas working space.

Our research points that non-expert programmers embrace desktop and paper metaphors in a virtual environment. We observed that our subjects employed similar screen organization strategies in the authoring environment regardless of operating a low-fi or hi-fi prototype during the participatory design sessions. It is important to stress that DETACH's final design will reflect these findings, to alleviate the technology transition impact which our stakeholders will be subject to.

Despite the success of the last prototype, where users clearly said that they liked the tool usability and gathering of all the functionalities in the same place, there was still space to improve it for the final DETACH product as:

- Some users missed the undo functionality, especially when deleting screens;
- Users liked to use keyboard shortcuts to perform actions such as deleting screens;
- Users liked to use template click in ways other than drag and dropping;
- Users were confused with the representation of the environment variables;
- The implemented screen analogy to post-its was so clear that to remove a wrongly chosen screen, instead of using the delete functionality, most of the users (~67%) prefer to put it back in its place.

Chapter 5

DETACH System Overview

"If you don't get the requirements right, it doesn't matter how well you do anything else.", Karl Wiegers (2004) [50].

After the specification of this project's goals, the analysis of the previous work, the outcome of a series of meetings with therapists and sessions with users we defined the requirements that should be verified by our system.

Software systems requirements engineering is the process of discovering the purpose for which the work is intended, by identifying stakeholders and their needs, and documenting these in a form that is amenable to analysis, communication, and subsequent implementation [51].

5.1 Use Cases

Use cases are a technique to identify the system requirements by representing the interactions it has with the different stakeholders. The technical details are hidden by a synthetic definitions of the multiple processes.

5.1.1 Stakeholders

First of all it is necessary to clearly specify who the different stakeholders of our final product are:

a) Non-Expert Programmers: In the first place there's the need to identify the elements our tool should offer to their target users. Based on this project domain, the first DETACH stakeholders are people that do not have programming skills. These are the typical users who will interact with DETACH to create mobile applications, so we involved them in the design process of our authoring tool. In light of the domain this tool focuses on, our attention was steered towards therapists and health professionals in general. However, we do not refrain people from other domains or even expert programmers from using the tool;
b) IT Professionals: The implementation of DETACH is carefully made having in mind the users who are going to continue the tool development in the future: Information Technologies professionals, the second stakeholders of DETACH. We focused the tool development at computer engineers such as Web and Android Developers;

c) Mobile Application End-Users: Finally, this project has to successfully use the proposed functionalities. When we focus on the DETACH mobile application there is no restriction or pattern we feel the need to specify about the people who are going to use it. These application can be in fact be executed by everyone, from young to old, to programmers or not. However, based on this project domain, we can focus mobile DETACH applications to patients undertaking a therapy treatment, our third and last stakeholders.

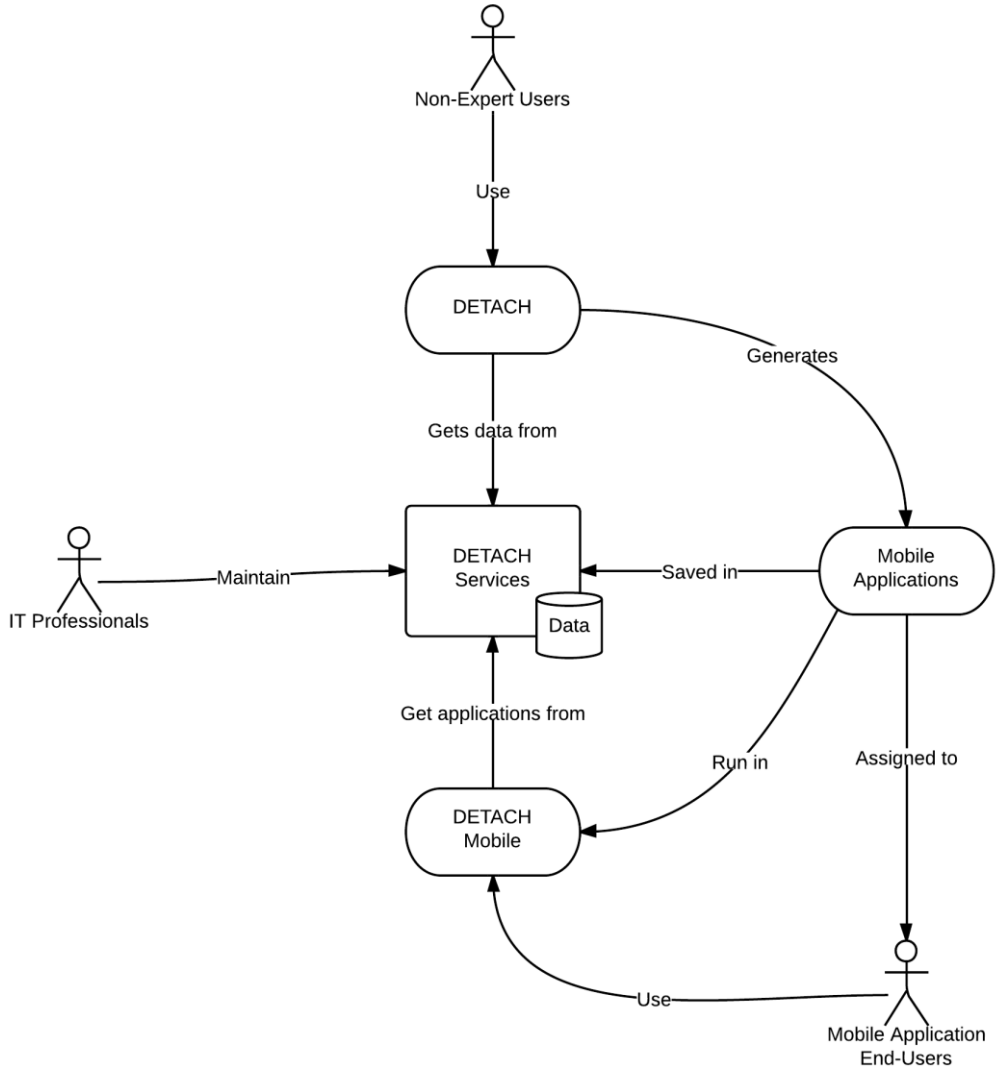

Figure 14 - Conceptual DETACH Framework

Figure 14 shows the conceptual framework that illustrates how our three stakeholders will interact with our system.

5.1.2 Use cases description

Based on the previous defined stakeholders, in this section we will identify, for each one, their possible interaction processes.

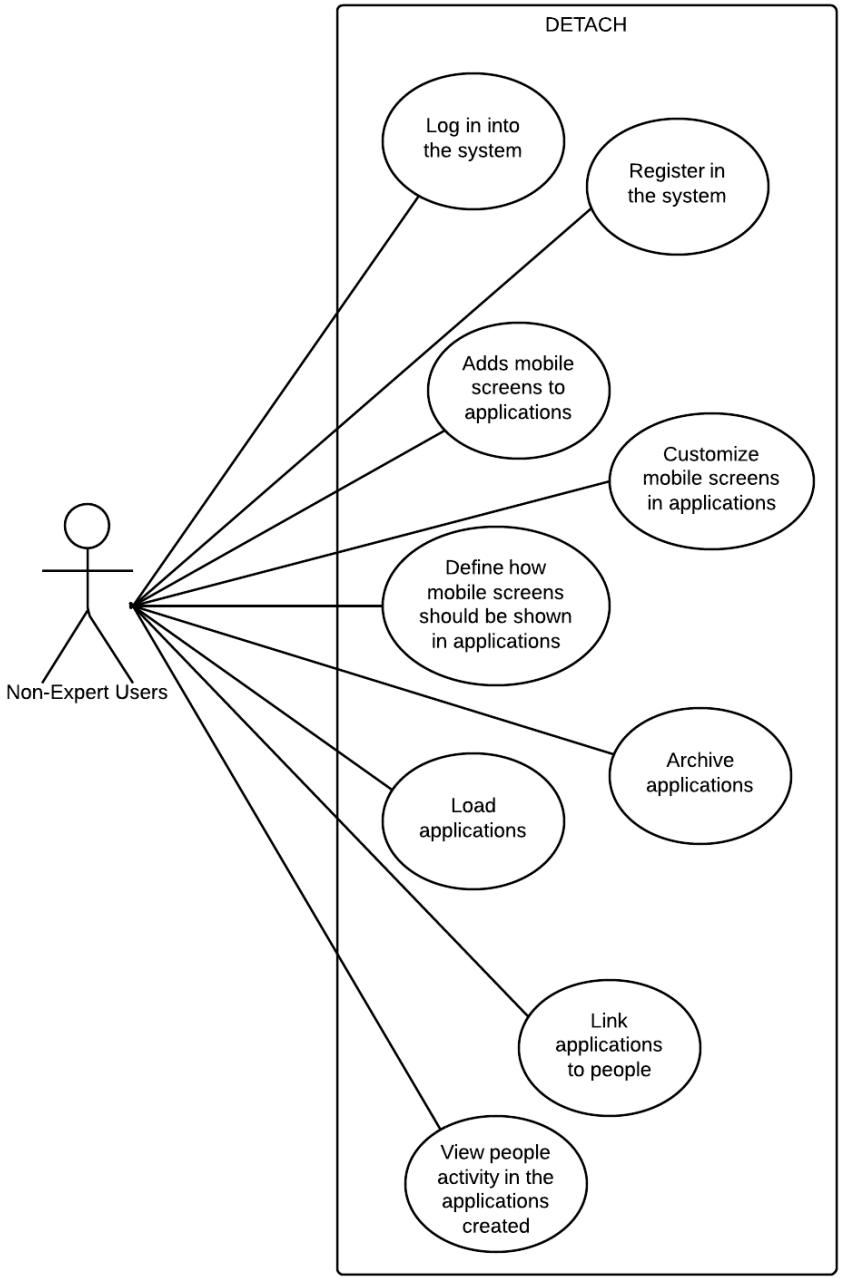

Figure 15 - Non-Expert Programmers use cases

Non-Expert Programmers use cases are (Figure 15):

- **Log in into the system** - The user must be able to keep his / her projects. Therefore, the system should offer the log in functionality to retrieve previous saved ones.
- **Register in the system** - In order to have a user account in the system, to save the multiple projects, the user must be able to register with his / her appropriate credentials.

- **Add mobile screens to applications** - A mobile application is composed by a set of different screens. The system should offer the possibility to add these screens to his / her current work.
- **Customize mobile screens in applications** - The user should be able to personalize the screen contents in order to adapt them to each person.
- **Define how mobile screens should be shown in applications** - The contents in a mobile application will change according to a specific screen / context behavior. The user should be able to specify this behavior to display the different mobile application screens.
- **Archive applications** - A created application should offer ways to be modified / continued in the future. In order to do this, the user should be able to archive them.
- **Load applications** - Previously created applications should offer a way to be loaded on user request.
- **Link applications to people** - The created mobile applications will target a specific person. The user should be able to define which person will run each application.
- **View people activity in the applications created** - Each application will have a record of the interactions made. The user should be able to review those interactions at any time.

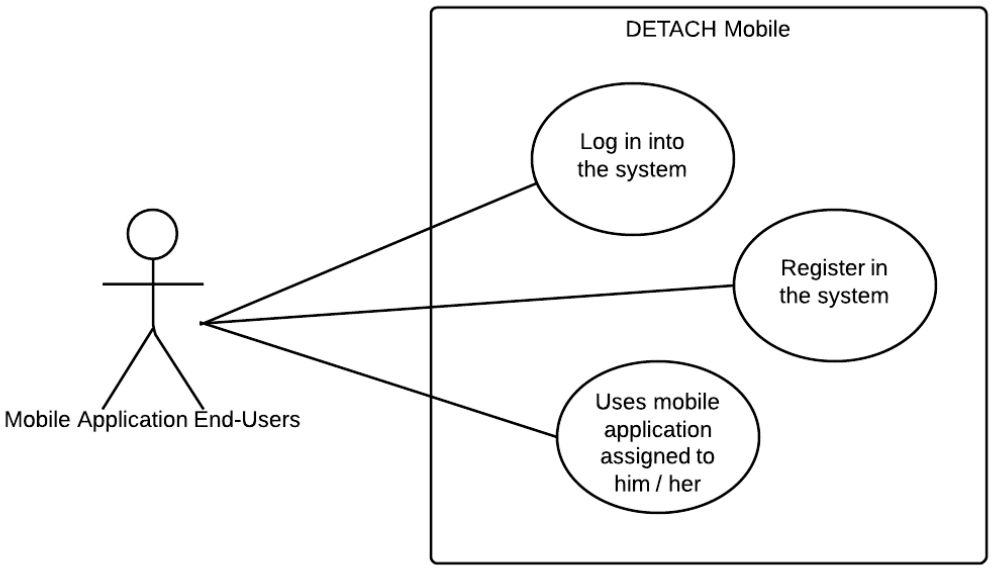

Figure 16 - Mobile Application End-Users use cases

Mobile Application End-Users use cases are (Figure 16):

- **Log in into the system** - The user must be able to get his / her projects. Therefore, the system should offer the log in functionality to retrieve the appropriate mobile applications assigned to him / her.
- **Register in the system** - In order to have a user account in the system, to get the multiple projects, the user must be able to register with his / her appropriate credentials.
- **Uses mobile application assigned to him / her** - The user must be able to interact with the mobile application created by its author.

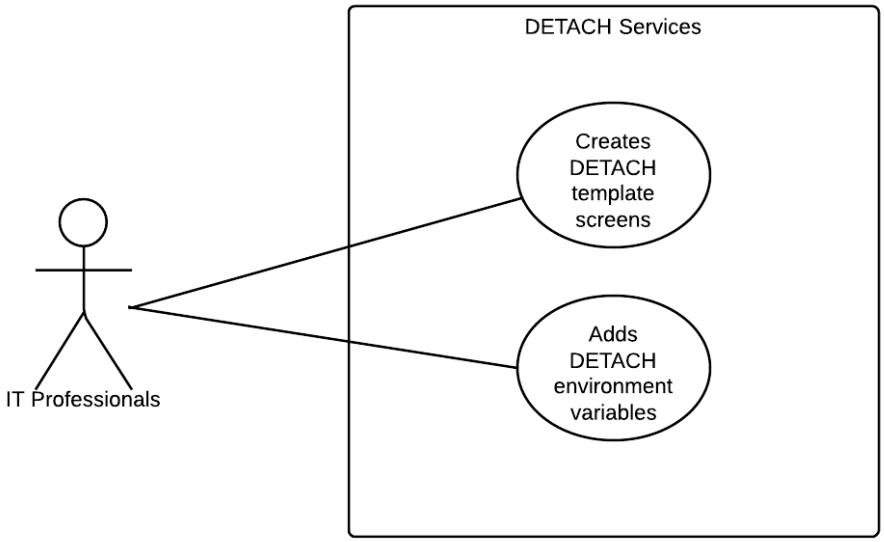

Figure 17 - IT Professionals use cases

IT Professionals use cases are (Figure 17):

- **Create DETACH template screens** - DETACH will offer a set of template screens to be used. In order to improve the system in the future, developers must be able to create new screens.
- **Adds DETACH environment variables** - Technology is always changing and the developers must be able to add new environment-based variables to the system.

5.2 Requirements

Currently several mobile applications addressing cognitive behavior therapy procedures are being used in clinical trials and for all of them therapists provide continuous screening and comments [44]. Several brainstorming sessions were conducted including a team of expert HCI researchers and the therapists specializing in different types of interventions and pathologies.

Based on these meetings, in the related work and in the sessions with end-users, we identified our project functional and non-functional requirements and grouped them by stakeholders. A full description of the system requirements is annexed to this document.

5.2.1 Non-Expert Programmers

This stakeholder should be able to easily create mobile applications by adding and organizing different template screens. He / she should be able to highly personalize them and define when they should be shown in the application. This can depend on other screens outputs or on context-based variables. Afterwards, this stakeholder should be able to assign the current application to a target mobile application end-user as well as store it for future modification. He / she should also be able to review the target mobile DETACH user activity in the created application.

5.2.2 IT Professionals

This stakeholder should be able to easily maintain and improve DETACH modules so that the tool can adapt to user requirements. This includes the addition of new template screens and environment variables to the interface.

5.2.3 Mobile Application End-Users

This stakeholder should be able to use the mobile application that was assigned to him previously.

5.3 Technology constraints

In this work two distinct components have to be developed:

- DETACH, the mobile applications authoring tool;
- DETACH Mobile, that is going to execute the mobile applications created in the above component;

5.3.1 DETACH

DETACH is aimed at people without programming skills. Having in mind that this range of users might not also be expert when using technology, it is important that the tool is the simplest possible to reach, requiring no installations or configurations. We also want it to be available to everyone and not just the users of specific working environments. This includes computers running Windows, Mac and Linux operating systems, for example, as well as tablets running Android or iOS operating system. We believe that it is important to being able to use this authoring tool in a specific place, and continue to work in another, where some of the times a different device, and therefore operating system, is being used.

In order to verify these previous goals we have developed DETACH as a web application. The interface is completely build with HTML5 and CSS3 and all the remaining tool functionality is supported by Javascript. As the initial requirements, it can be used across all main browsers in different operating systems. Not only Windows or Mac operating systems we also wanted our tool to be used in nowadays tablets, with iOS or Android OS. In order to do so, we've made DETACH content to adapt itself to different screen sizes, as well as providing different interaction methods to be used in these devices (such as tap instead of drag and drop of the screen templates).

DETACH server uses PHP language and MySQL to handle functions to:

a) Save, load, archive, restore and delete of user projects;
b) Load of template mobile screens;
c) Register and login DETACH and mobile DETACH users as well as recover and change their passwords;
d) Write and obtain user log files.

5.3.2 DETACH Mobile

From another point of view, it is important to understand what would be the best technology to implement the mobile component of this system. For that we had three possibilities:

- Through a native mobile application;
- Through a mobile web application, that could be executed in every mobile browser;
- Through a hybrid application, a mix of the above that consists in a native application with a web browser inside.

Bellow we can see in more detail what were the top benefits and disadvantages of each approach:

	Performance	Hardware access	Platform compatibility	Programming language
Native Application	High	High	Low	Java
Web Application	Low	Medium	High	HTML5, JavaScript and CSS3
Hybrid Application	Medium	High	Medium	HTML5, JavaScript and CSS3

Table 2 - Mobile DETACH possible implementation solutions comparison

Since we wanted that our application was widely used by everyone, the platform compatibility had a big impact on the elimination of building a native application. Instead, and having in mind that we wanted the application to be constantly available to the person, and not messed in between some browser tabs that the user can exit by mistake, we decided to adopt the hybrid possibility, starting our development for the Android platform. Therefore, DETACH Mobile uses Java with a WebView that handles HTML5, CSS3 and Javascript content the same way as DETACH. Currently it runs on top of the Android OS.

5.3.3 Architecture

DETACH and mobile DETACH use a thin client/fat server architecture.

The implemented solution requires an internet connection so that the DETACH and mobile DETACH users can access the created applications right away and anywhere. DETACH user can for example start a specific application from his / her home computer and continue it from the office laptop, without requiring any cables or file transportation systems (Figure 18).

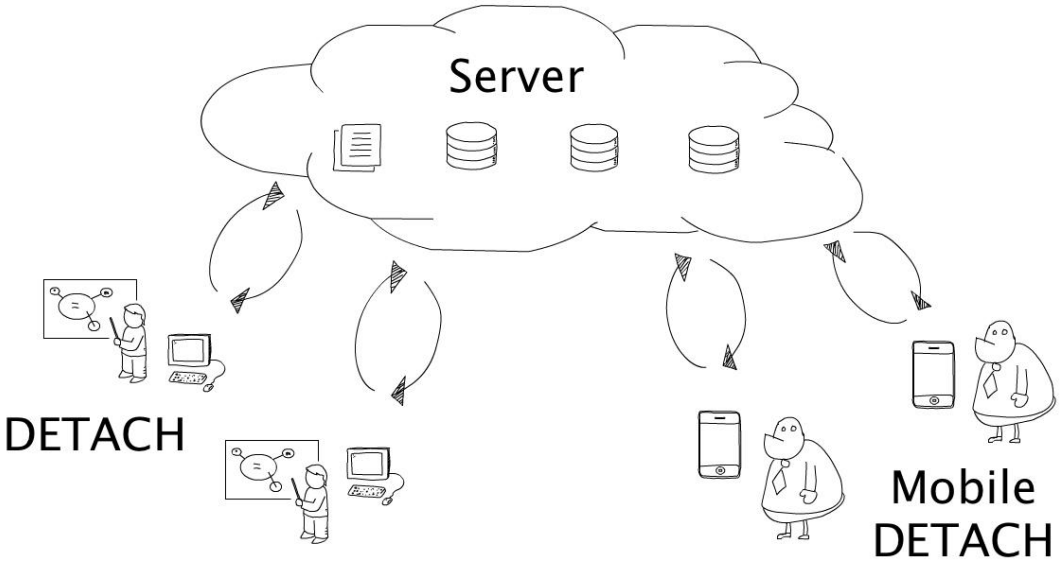

Figure 18 - DETACH architecture

Being a web application, DETACH maintains all its content on a web server. This contains a user and project database as well as the mobile screens templates and all programming code behind.

DETACH requires a connection to the server every time it is executed, in order to load the interface, to authenticate a user and to load, archive or modify a specific project. With the implemented solution the projects are automatically saved to the server in real time.

The mobile DETACH application, that interprets the applications created with DETACH, also requires a connection to the server in order to authenticate their users. This allows the loading of the last version of the application created to the authenticated user. It also requires a connection to the server every time there is the need to transmit user activities in a mobile application to DETACH, in order for them to be displayed in real time to the application's author.

5.4 Summary

Previous work together with some meetings with therapists and sessions with end-users allowed us to perceive our project requirements.

These requirements were associated with the different stakeholders of our project in a way that one could perceive how would they interact with the system. Lastly we have also identified some of the system requirements and technologies we found important to mention.

DETACH will be designed following the specifications of this chapter.

Chapter 6

DETACH for Non-Expert Programmers

DETACH's final prototype was developed based on the end-users' needs and interaction patterns identified during the low and high-fidelity prototypes experimental periods. From the end-users point of view, we came up with a clean and easy to use interface that offered all the features required in order to create, test and distribute a mobile application. Testing functionalities are available through DETACH run-time emulator that will be fully described in Chapter 8 side by side with the mobile framework.

6.1 Interface

DETACH's interface is divided in three areas (Figure 19):

a) A top section housing the available screen templates that we felt important from previous trials, related work and meetings with therapists (such as screens with open answers);
b) A central canvas to which the screen templates may be dragged upon and organized in an extendable canvas, as preferred by users in the initial trials;
c) A configuration panel on the right side that enables the customization of the selected screen(s)' elements and styling.

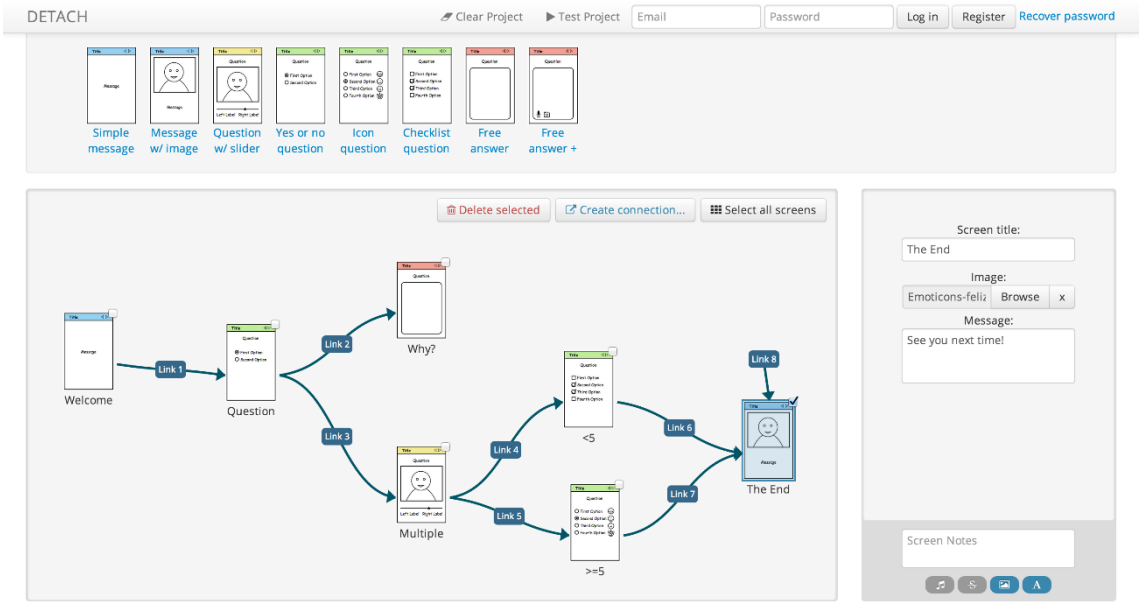

Figure 19 - DETACH final product interface

6.2 Features

Previous sessions with our high-fidelity prototypes allowed us to improve current and implement missing functionalities for our final product. The following functionalities assume the DETACH's user to have started the application.

6.2.1 User management

DETACH needed to manage its users and respective projects, in order to allow the application saving and future modification. Therefore it should offer the possibility to be used by multiple users simultaneously where each one can access his / her projects only. As result, we now provide ways for people to authenticate in the system and manage their projects.

Figure 20 - DETACH user management bar representation for non-authenticated users (top), authenticated users in a new project (middle) and authenticated users in a previously saved project (bottom)

When authenticated, besides project testing, a person can now load a previous project to improve, archive and assign it to a specific mobile DETACH user(s) - by associating the desired user(s) email(s) to the project. Afterwards the application author can also see his / her activity in the application. This includes perceiving what answers where provided, what buttons were clicked and at what time was shown each screen. This allows therapists to completely replace and improve current paper artefacts.

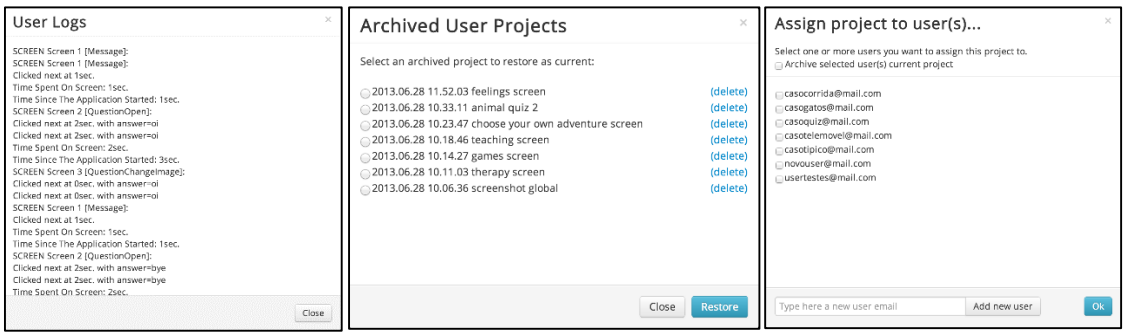

Figure 21 - DETACH user logs, archived projects and available users dialog

6.2.2 Application Styling

A functionality that was not yet implemented in previous prototypes and actually present in the initial requirements elicitation was the presence of screen styling, namely screen music, subliminal content, background and text styling (Figure 22). These could be specified to a single or multiple selected screens at once and allow the application to be highly personalized to each user needs.

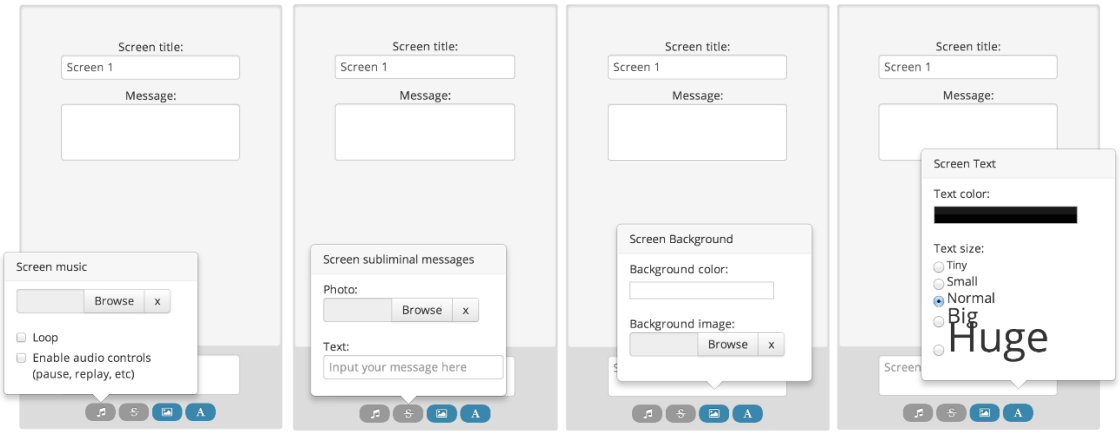

Figure 22 - DETACH application styling options

Screen music

Users can specify a sound or music to be played in the selected screen(s). In case the user assigned the same song to multiple screens, when displaying each screen the audio will:

a) Restart if it had already ended in the previous screen;
b) Continue playing if it still did not end in the previous screen.

This last situation was specially designed for cases where the background audio is a music file and the first case for when it is a short sound. Users can also specify if the background audio will loop or display audio controls in the application, enabling play / pause and stop functions.

Screen subliminal messages

Subliminal content was a requirement found in initial meetings with therapists and allows the specification of an image and/or text message that will appear in the screen for brief periods (entirely customizable). This is used to influence a person behavior and can therefore be applied in multiple treatments.

Formatting

In order to offer some degree of personalization, screens can have a specific background color and image. Their content (e.g. text messages mainly) can also be

customized in regards to size and color. This may prove its utility when defining an application for older people with eyesight problems.

6.2.3 Mobile screen templates

To this final product, new screens emerged from user requirements. While in previous prototypes we just provided 5 different template screens (Message, Animation, Question, Feelings and Free Answer Screens) we have extended the diversity of templates to a total of 8 (Figure 23). This emerged from initial meetings with therapists and trials with users.

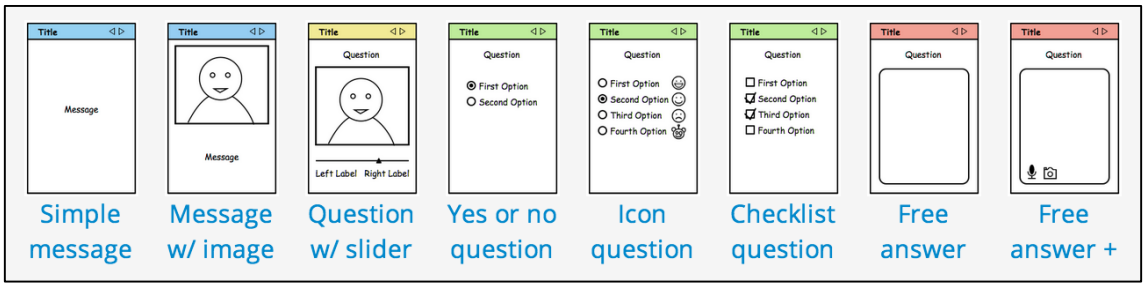

Figure 23 - DETACH mobile screen templates

We also revamped the organization of these templates within DETACH. Rather than following a random distribution, the order now reflects:

- Screens containing messages (blue color);
- Screens containing dynamic content that changes according to user answers (yellow color);
- Screens containing questions with possible choices (green color);
- Screens containing free answer questions (red color).

After a screen is clicked or pulled to the center canvas that same screen is added to the current application and the rightmost configuration panel is updated with its configurable elements. Each of these screens contains different elements that are used to generate the screen. Common to all screens are the screen title, screen notes, where users can take some appointments only visible to themselves, and screen add-ons, such as audio, subliminal content, background and text styles.

Simple message

This screen is intended to be used when there's some content that we want to transmit without any type of question. Its use can be made when some therapist want to teach how a patient should react to a specific situation (Figure 24).

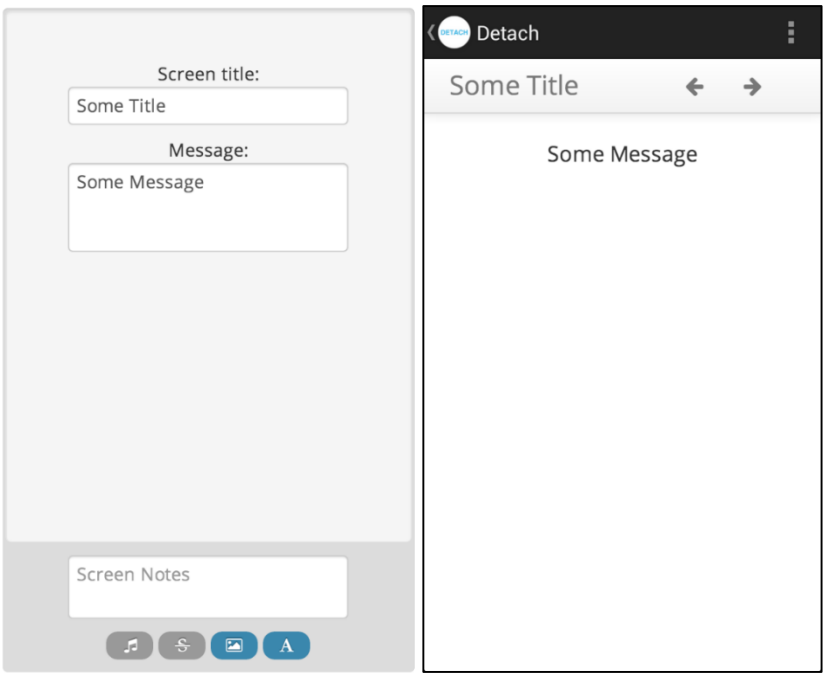

Figure 24 - Simple message example screen editing and respective run-time emulator result

Message with image

The message with image screen is similar to the previous one except the user can also input a specific image (animated or static) on top of a specific message (Figure 25).

This screen is also intended to be used when there's some content that we want to transmit without any type of question and is especially designed targeting younger users with reading difficulties.

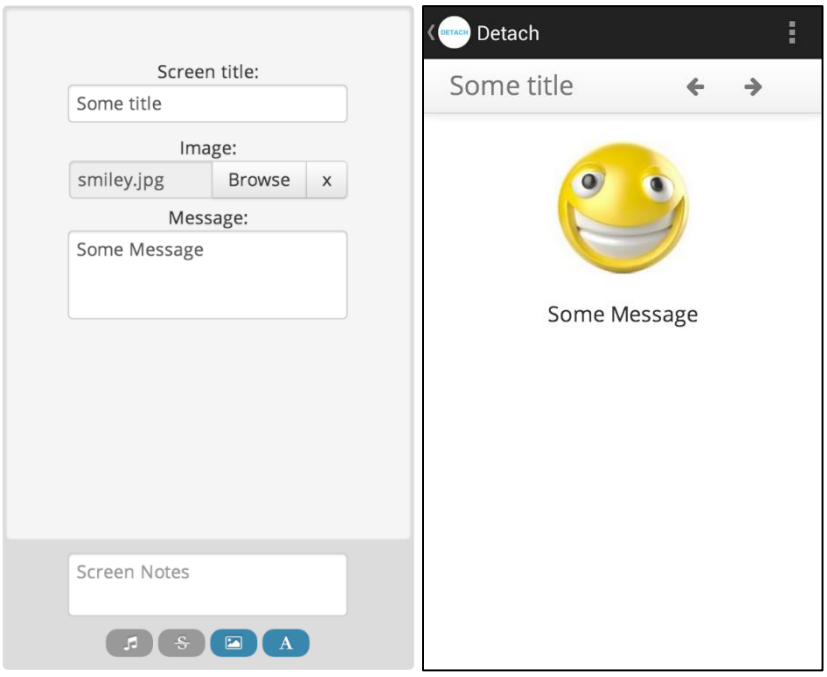

Figure 25 - Message with image example screen editing and respective run-time emulator result

Question with slider

This screen was actually a requirement from the initial reunions with therapists and allows users to dynamically change a screen image according to a range of answers. Typical questions ask people undertaking some phobia treatment to quantify their fear level. By inputting a series of images, these are assigned alphabetically to the answer slider value (Figure 26).

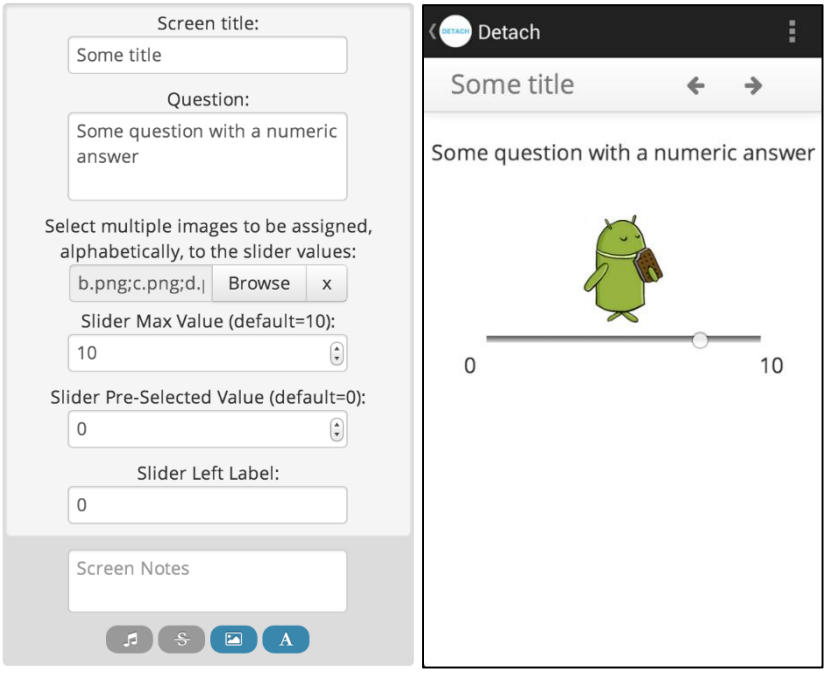

Figure 26 - Question with slider example screen editing and respective run-time emulator result

Yes or no question

This screen allows a user to ask a question with two possible answers (Figure 27). While yes or no question is given as screen name for better understanding, this screen can actually be used for other types of answers that contains two values: Hot and cold, up and down, etc.

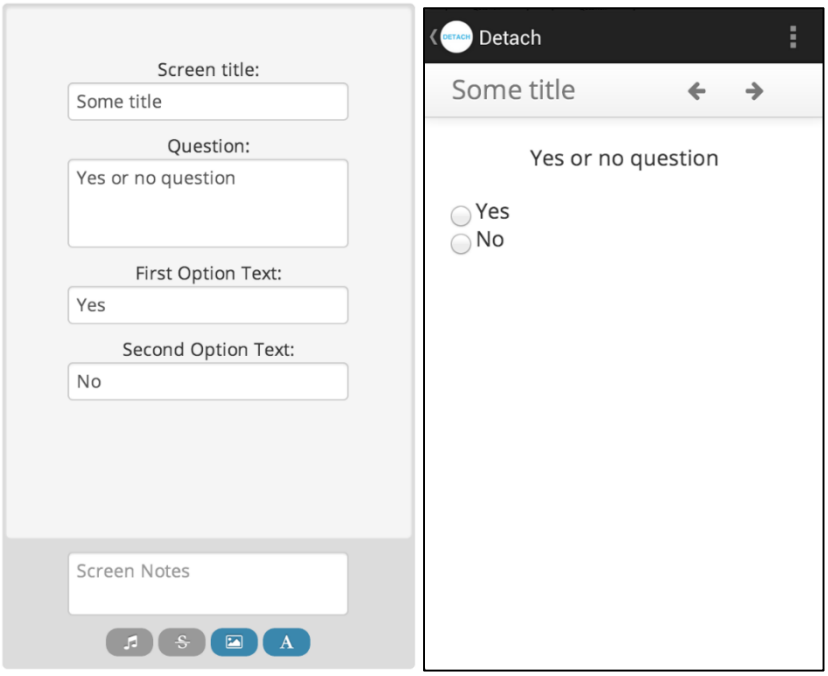

Figure 27 - Yes or no question example screen editing and respective run-time emulator result

Icon question

While the previous screen allowed some basic question creation, with the icon question one a user can assign an image to each answer possibility. This can be useful for psychological questions based on user feelings for example.

While we provide 4 fillable answers, they are all optional and their content can rely in pure text or image only (Figure 28).

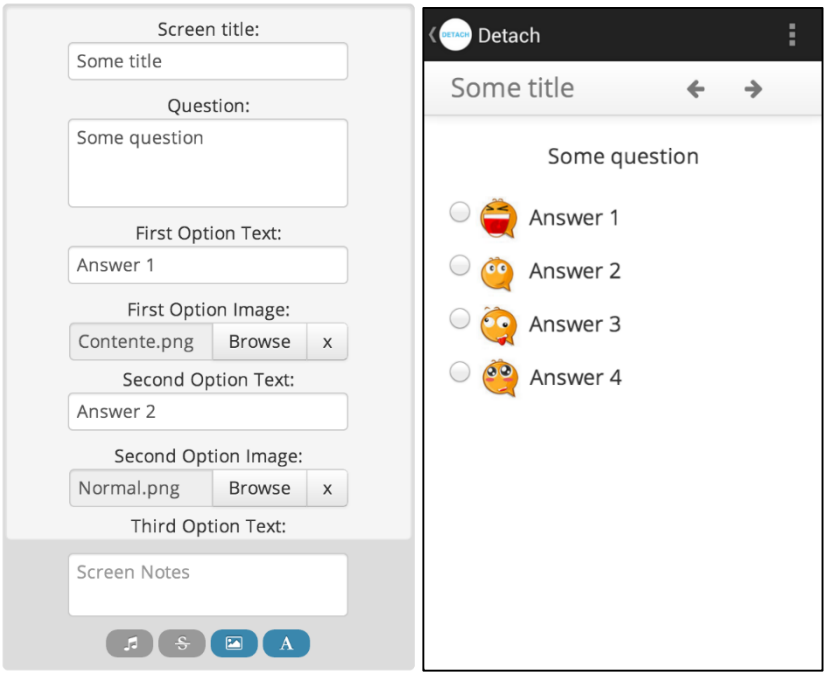

Figure 28 - Icon question example screen editing and respective run-time emulator result

Checklist question

The checklist question screen is based on the previous one, expect in this one, multiple answers are possible (Figure 29).

Examples uses include the treatments of pain disorders when patients are asked to indicate the body areas where they are having pain.

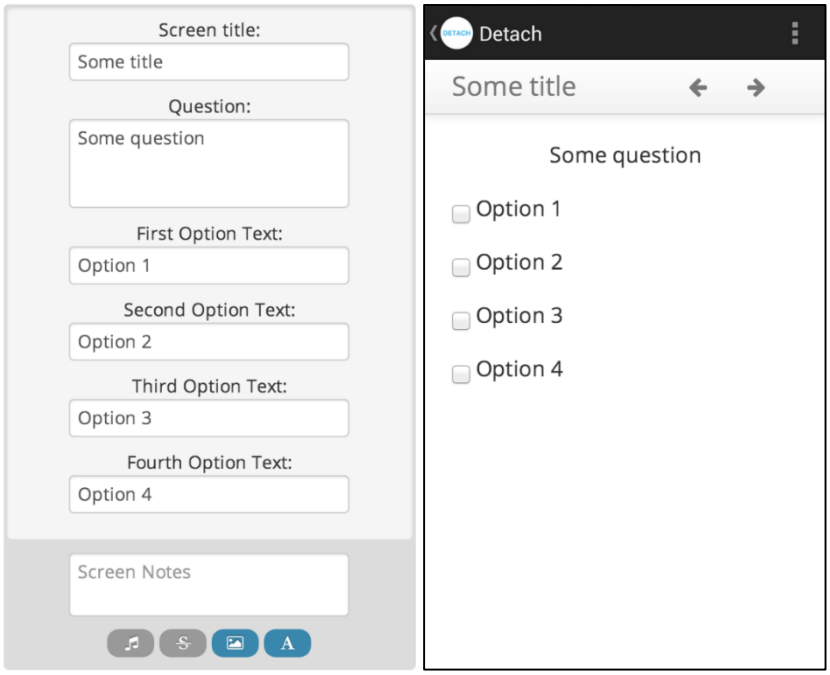

Figure 29 - Checklist question example screen editing and respective run-time emulator result

Free answer

While in previous question screens we restricted user answers, in the free answer screen, as pointed by the name, the user just needs to specify the question to be made (Figure 30). This type of screen is particularly useful when asking for thoughts or activity records.

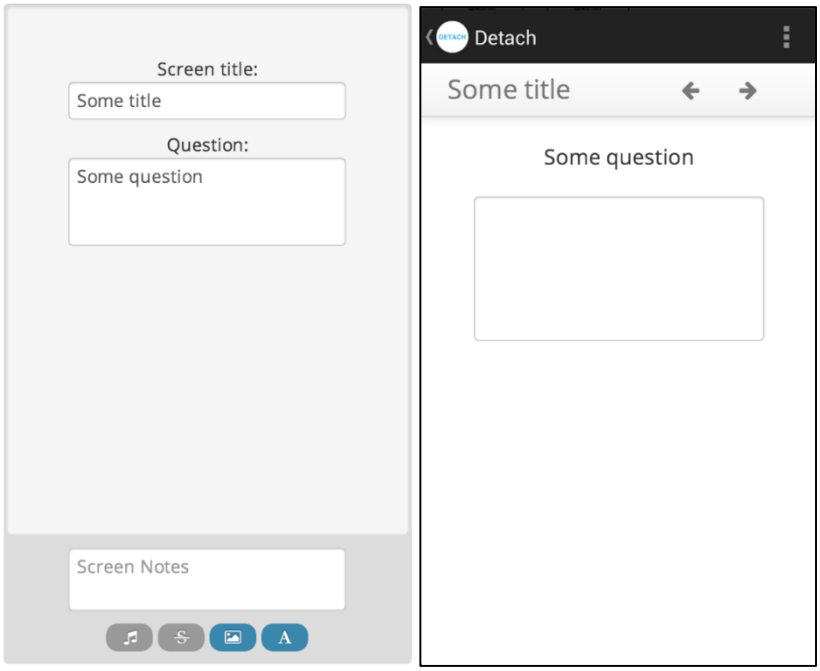

Figure 30 - Free answer example screen editing and respective run-time emulator result

Free answer plus

The last screen is in all similar to the previous one, except is also allows the mobile user to attach some picture, video or audio to the answer content (Figure 31).

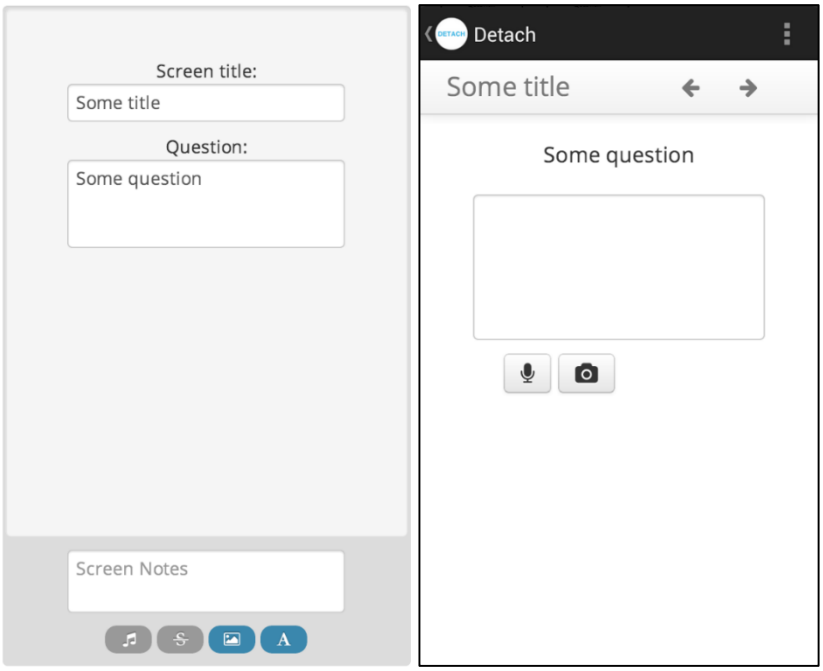

Figure 31 - Free answer plus example screen editing and respective run-time emulator result

Screen Deletion

Screen deletion was actually one of the functionalities that required some improvements. Particularly important was the fact that previous prototypes were too restricting concerning the way users were able to delete screens or recover them. For our final DETACH tool, we have implemented the methods that were tried to be used our participants (such as the drag and drop back to the screen templates area or the usage of keyboard shortcuts). We also complemented all these improvements with the addition of an undo functionality as requested by some of the users (Figure 32).

Figure 32 - Delete selected and undo delete DETACH functionalities buttons

6.2.4 Transitions

Also improved was the connection definition screen, that appears after clicking 'create connection' and specifying the destination screen, following the origin-destination paradigm found in previous trials. During low-fidelity prototype testing, we observed the need to include a way to connect screens based on a combination of events. In that way, DETACH now offers the possibility to specify more than one rule to trigger a connection, with proper verification of the conjunction operators AND / OR as required by the users (Figure 33).

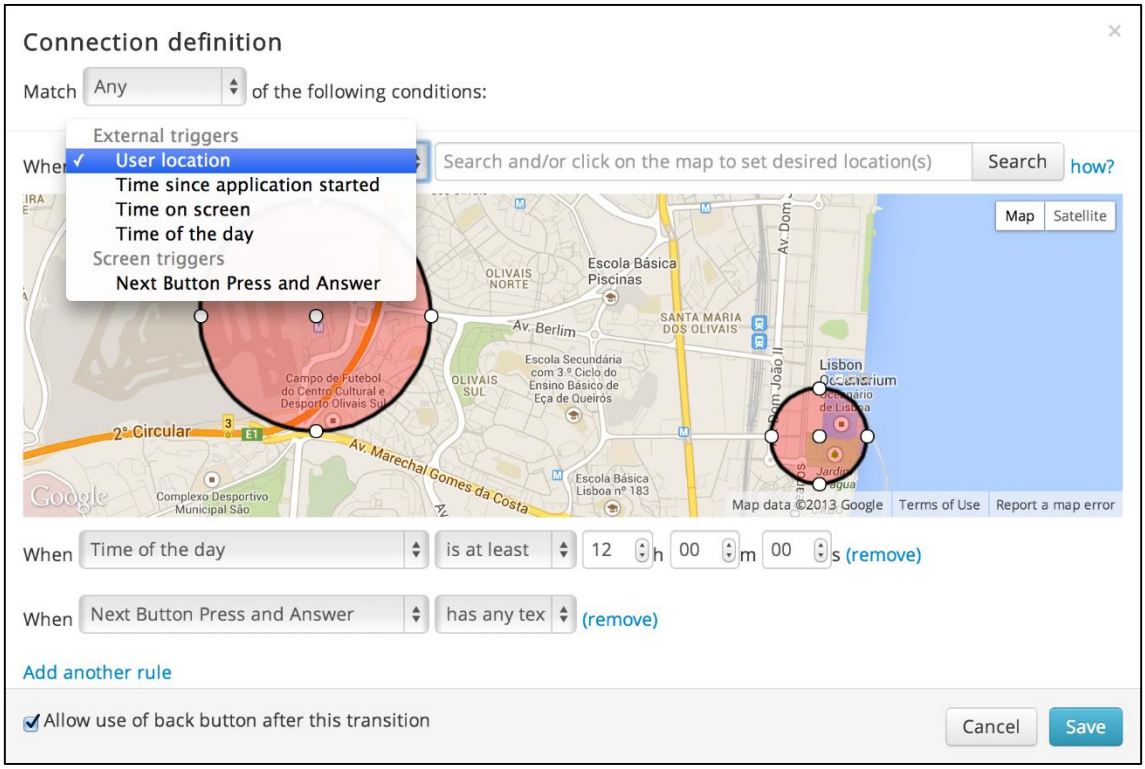

Figure 33 - Example connection condition rules specification

Screen triggers

Connections can be created based on the different screen triggers of each screen (Figure 34):

a) Simple message and message with image: These types of screens just allow actions based on the navigational next button click;
b) Question with slider: Contrasting the previous screens, this one can trigger actions based on user answers, creating different application flows;
c) Yes or no question: Similarly to the previous screen, with this one users can also create connections based on provided answers;
d) Icon question and Checklist question: These screens follow the same principle as the previous one except they let users check their 4 possible answers;
e) Free answer and free answer plus: These last screens actually allow the verification of a free answer content to create different application flows.

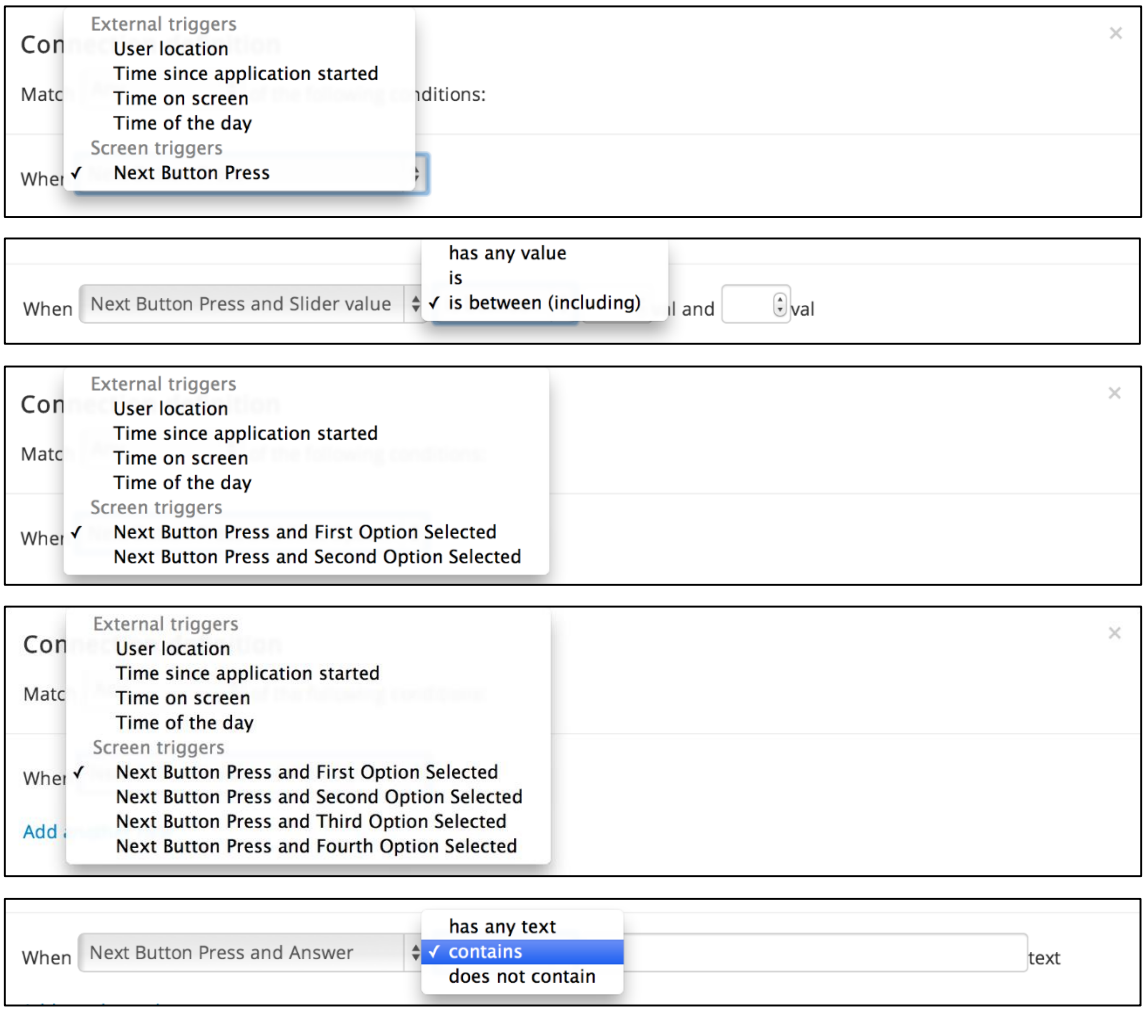

Figure 34 - Simple message and message with image screen triggers, question with slider screen triggers, yes or no question screen triggers, icon question and checklist question screen triggers and free answer and free answer plus screen triggers, respectively

Environment Variables

The usage of external data, which did not depend on screen contents, is now also available through the triggers (Figure 35):

a) User location: With the use of a map, users can make a specific connection to occur on one or more geographical points. In order to do that the user can indicate on the provided map the location(s), as well as the respective area(s) that will be associated with the current rule. Users can also quickly jump to a specific area by using the map search box.

b) Time since the application started, Time on screen, Time of the day: These three types of time values could also be observed with a sub-condition of is at least, is at most or is between specific values.

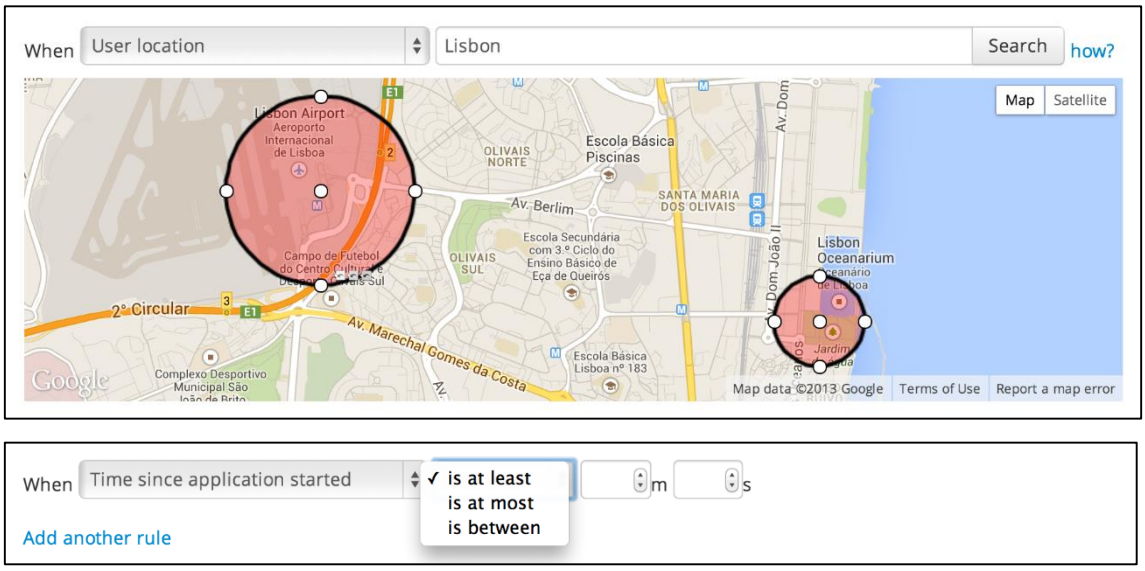

Figure 35 - Location and time based condition rules example

As the different types of connections, overlapping each other, confused users, those that do not depend on a specific screen but rather on environment variables were also simplified by a representation pointing to the screen that should be displayed (Figure 36).

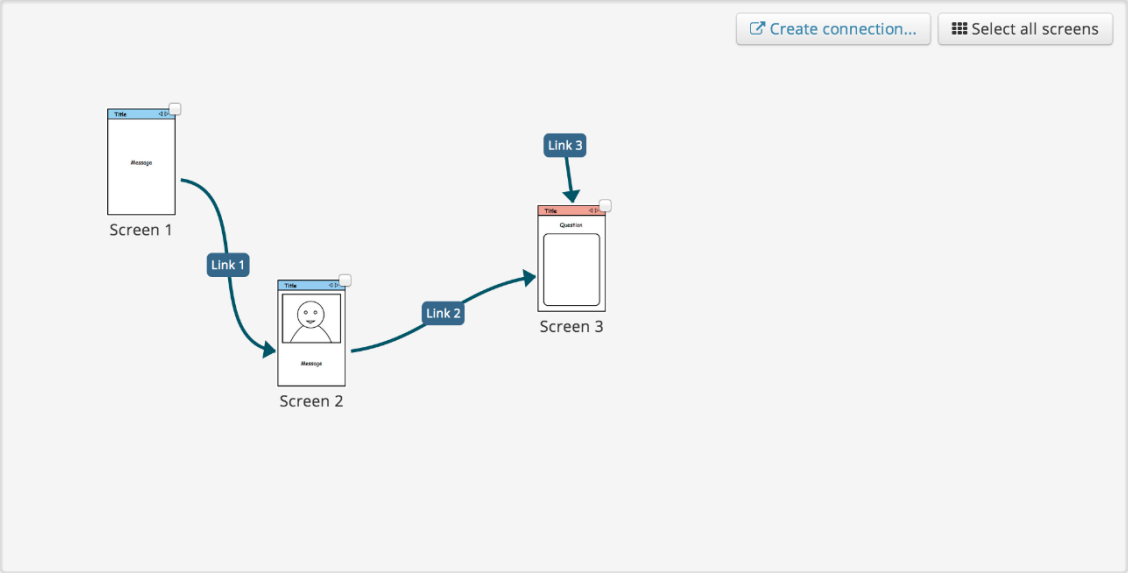

Figure 36 - Resulting representation of an application that uses transitions based on screen triggers and external environment triggers

6.2.5 Tutorials and samples

In order to explain application functionalities some tutorials and samples were also included.

Figure 37 - DETACH tutorials and samples dialog

This dialog in Figure 37 can be reached after clicking the DETACH application title and includes a set of quick tutorials of the main application functionalities as well as some sample projects that work similarly to the loading of previous projects.

6.2.6 Potential Scenarios

Although we've focused our work in the health area as the main DETACH intervention scenario, its application potential for other domains it's evident. In order to prove so, professionals of several other domains were interviewed in the final tests. Even though these interviews hadn't follow the initial scientific strictness, they managed to show the tool flexibility.

Therapy

Being the starting point of this project, we must mention that with the created DETACH tool, therapists can finally create their own applications. Animal, dark or social phobia are just few examples were this tool can be used in (Figure 38).

In a quick and easy way, therapists can now personalize and adapt all application content to each of their patients. The ability to personalize screen background, text color and size, besides all the remaining elements like images, sounds and even subliminal content, in the shape of text and/or image, allow an infinite set of applications, available in real time to the patients.

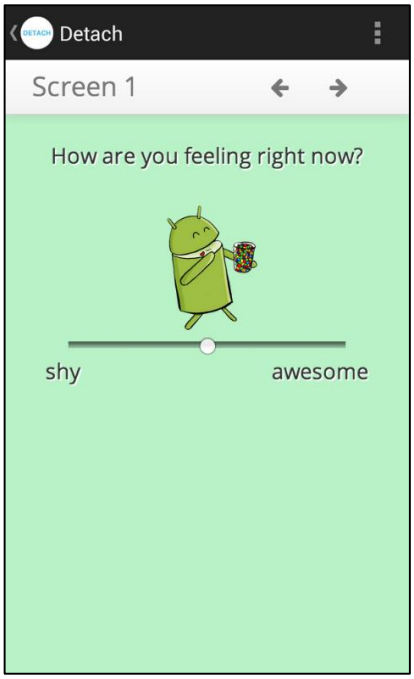

Figure 38 - DETACH therapy scenario example

Teaching

During the multiple evaluation sessions we've had with our last participants, the teaching area was by far the one that was most mentioned. It is indeed an area where a specific teacher can for example create an application to each subject and distribute it to the corresponding students (Figure 39). It's proven that teaching through technology has

further interest than conventional one with books. Besides, the fact that the application can display additional content such as animations, sounds and different application flows according to user interactions allow the application to have a further interest to the students.

Additionally to subject teaching, these applications can also, in the end, display little questionnaires where the results are automatically sent to the teacher.

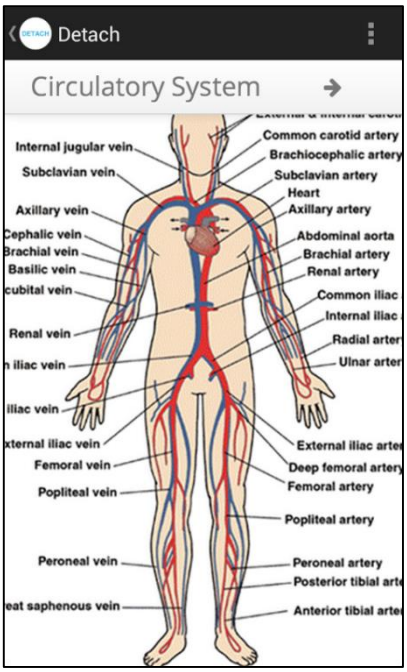

Figure 39 - DETACH teaching scenario example

Games

In this area, and using the tool capabilities, quiz games were mentioned (such as "who wants to be a millionaire"), that use questions with an answer time limit to make the player reach a specific level (Figure 40). After DETACH tool functionalities further explained, new possibilities to these kind of games that are not typically available were quickly perceived. For example, the ability to advance in the quiz according to the provided answers also allows the questions to adapt itself to the user knowledge. Lastly, and using the mobile application GPS access, we must mention the geo-referenced games, where a user has to cross multiple city spots in order to gain points.

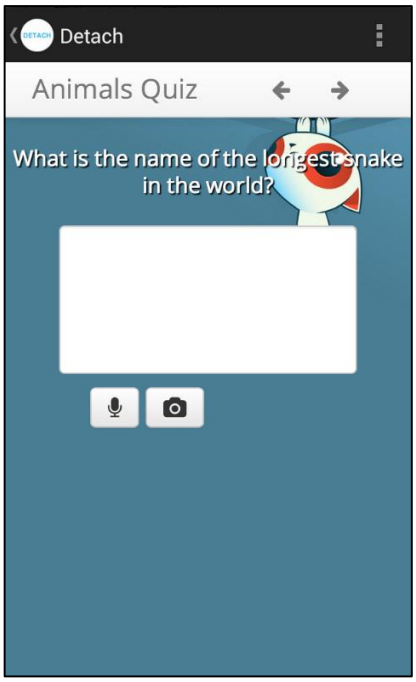

Figure 40 - DETACH gaming scenario example

Interactive stories

Based on the books that allow people to create their own adventure ("Make your own adventure"), that mention the page number a person should go according to what they think the character should do, this tool improves that concept by being able to tell the story, with further animations and sounds, and, according to a user answer, make this transition automatic (Figure 41).

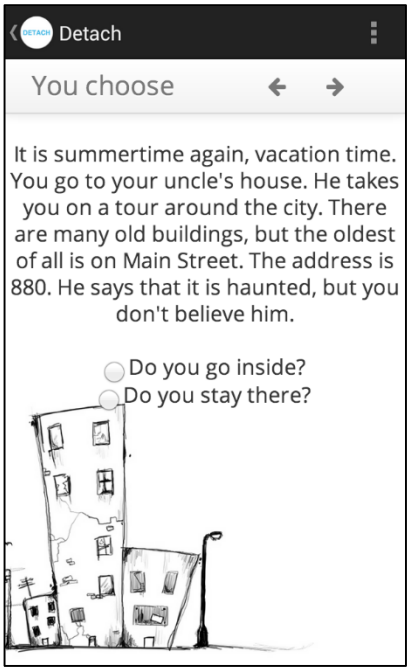

Figure 41 - DETACH interactive stories scenario example

Chapter 7

DETACH for Developers

From the Developers point of view we built DETACH components modularly so that they can be easily managed and improved in the future.

As shown in the highlighted part of Figure 42, these modules are loaded as soon as DETACH is started and comprise the screen templates list and the environment variables the end-user will have access to.

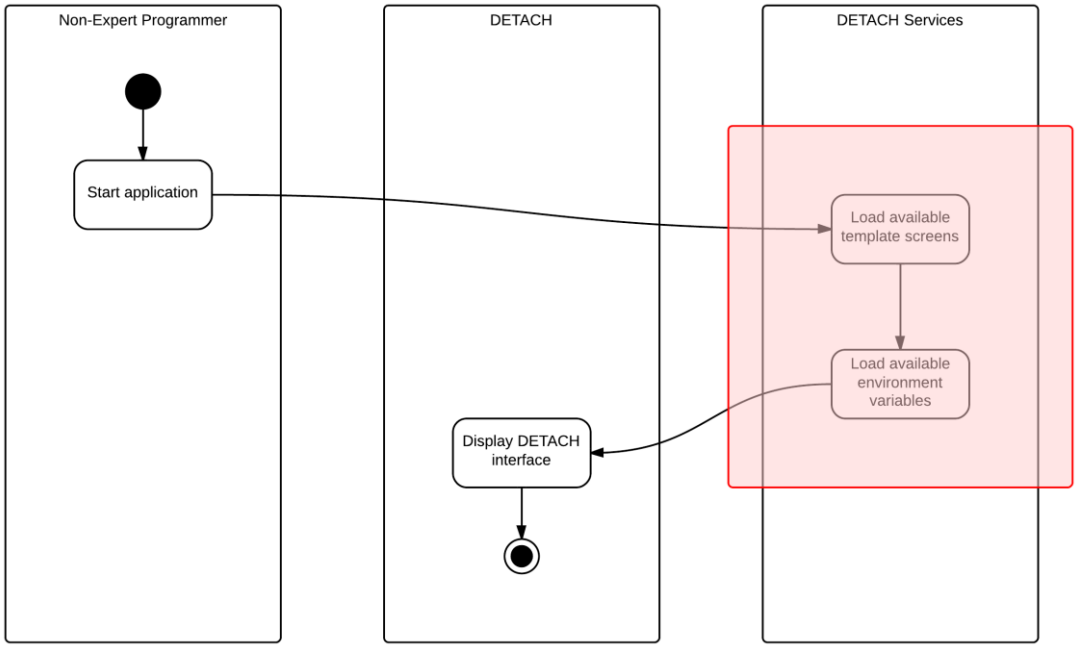

Figure 42 - DETACH start activity diagram

7.1 Adding Mobile Screen Templates

A developer can easily add a new mobile screen template apart from the ones we already made available. In order to do that, he / she has to define which screen contents and triggers will the screen have as well as their result when execution the application.

Each template screen is composed by three files, each one interpreted differently:

a) A XML one with the specification of the screen details that a user handles in DETACH, such as possible fields that could be personalized and appropriate triggers that the screen would make available for the transitions;
b) An image file that would represent the screen in the application;

c) A Javascript file containing information about how DETACH run-time emulator and mobile application would display and interpret the data contained in the XML file.

All these files should have the new screen name and be added to the DETACH server screens folder.

7.1.1 The image file

This file has to be in the .png format and will contain the DETACH image representation of that screen. For better representation the image should have 174 x 267 pixels. If a developer wants to follow the other image representations, Balsamiq Mockups or Swordsoft Layout software must be used to design the screen and a color tool to match the screen to it's respective type.

7.1.2 The XML file

The .xml file must contain the tags type, description, fields and triggers as seen in Figure 43.

```
<screen>
    <type>The screen type to be shown below the template screen image</type>
    <description>Screen description</description>
    <fields>
        <Text>Field description</Text>
        <Textarea>Field description</Textarea>
        <Number>Field description</Number>
        <SingleFile>Field description</SingleFile>
        <MultipleFile>Field description</MultipleFile>
    </fields>
    <triggers>
        <trigger id="1">Trigger 1 description</trigger>
        <trigger id="2">Trigger 2 description</trigger>
    </triggers>
</screen>
```

Figure 43 - Screen XML file structure

The <fields> tag describes all the fields the user can configure for this specific screen and contains a Text and/or a Textarea and/or a Number and/or a SingleFile and/or a MultipleFile, when the user can choose more than one media file for that screen. This tag can have as many fields as needed, repeated or not, represented by the type as tag, and by the field description as tag content.

The <triggers> tag contains all the textual descriptions of the triggers a screen has for the connections, as well as their identification numbers. These triggers will have to be coded in the JavaScript file.

7.1.3 The JavaScript file

The JavaScript file must contain the code to generate the screen and the code that will interpret every screen trigger. A full explanation of the functions to code is available in DETACH developers guide (annexed to this document).

Briefly, to generate the screen the developer should add the function new [ScreenName]Screen. In this function all the screen contents that were filled in DETACH should be correctly positioned in a container. These contents are automatically passed to this function by argument.

For every screen trigger, the developer has to add the function set[ScreenName]Trigger[TriggerId]. Each of these functions will be executed for the according XML trigger Id. The developer has to code what screen elements will execute the trigger and when.

7.2 Adding Environment Variables

Each environment variable in DETACH is composed by two distinct files:

a) A XML one with the specification of the environment variable available trigger(s) for transitions;
b) A Javascript file containing information about how DETACH run-time emulator and mobile application would interpret the trigger specified in the XML file.

All these files should have the new environment variable name and be added to the DETACH server sensors folder.

7.2.1 The XML file

The XML file works similar from the one present in each screen except it will not contain the fields tag. Apart from that tag, the triggers also have to be specified.

7.2.2 The JavaScript file

Also similarly to a template screen JavaScript file, the one that respects environment variables will have to contain the same set[ScreenName]Trigger[TriggerId] function for each trigger specified in the appropriate XML. The function new [ScreenName]Screen is not required for this JavaScript file.

7.3 Scenario

As an example to add a new mobile screen template to a therapy assignment, let us resume David's scenario presented in Chapter 2, where the patient felt distressed near hospitals. For this kind of scenario, his therapist Claudia found important to have a new

template screen that possesses two distinct images that would represent the before and after thoughts that were desired during the exposure process.

DETACH developers could easily add the desired new screen template by finding or designing an appropriate image for the new screen. Afterwards they would name that image BeforeAfter.png, for example, and place it inside the DETACH server screens folder. With that same name, BeforeAfter.xml and BeforeAfter.js files would also have to be added to the previous folder.

The BeforeAfter.xml file could contain the specification in Figure 44.

```
<screen>
    <type>Before and after</type>
    <description>A screen that displays two images, each one with an appropriate description</description>
    <fields>
        <Text>Before message</Text>
        <SingleFile>Before image</SingleFile>
        <Text>After message</Text>
        <SingleFile>After image</SingleFile>
    </fields>
    <triggers>
        <trigger id="1">Next button press</trigger>
    </triggers>
</screen>
```

Figure 44 - Example screen XML file specification

The BeforeAfter.js file would firstly have to place the screen contents in their desired positions as seen in Figure 45.

```
function newBeforeAfterScreen(div,screenId,screenContents){

    addTopNavigationBar(div,screenContents[0],"next"+screenId);

    var obj, obj2, conteudo;

    div.innerHTML+=screenContents[1];

    obj=document.createElement("img");

    if(screenContents[2]!=""){
        obj.src="projects/"+window.sessionStorage.getItem('projname')+"/"+screenContents[2];
        obj.style.height = '100px';
        div.appendChild(obj);
    }

    div.innerHTML+=screenContents[3];

    obj2=document.createElement("img");

    if(screenContents[4]!=""){
        obj2.src="projects/"+window.sessionStorage.getItem('projname')+"/"+screenContents[4];
        obj2.style.height = '100px';
        div.appendChild(obj2);
    }

}
```

Figure 45 - Example JavaScript function code to generate a screen that contains two messages and two images

Afterwards the function related to the new screen trigger has to be coded. This function indicates that the following screen is shown when the user clicks the next button and can be coded as seen in Figure 46.

```
function setBeforeAfterTrigger1(screenId,eventCondition,eventData,eventDest,allowbackcondition){

    document.getElementById("next"+screenId).onclick=function(arg1,arg2,count,back) {
        return function() {

            appendToLog("Clicked next at "+screentime+"sec.n");

            markThisConditionAsVerified(screenId+"-"+arg2,count);

            if(arg1=="Any" || checkIfAllConditionsChecked(screenId+"-"+arg2)){
                deactivateAllScreens();
                activateScreen(arg2,back);
            }else{
                markThisConditionAsNotVerified(screenId+"-"+arg2,count);
            }

        };
    }(eventCondition,eventDest,connectConditionCount,allowbackcondition);

}
```

Figure 46 - Example JavaScript function code to prepare and execute a screen trigger

The recently added template screen can be used next time an end-user loads DETACH and will be available as shown in

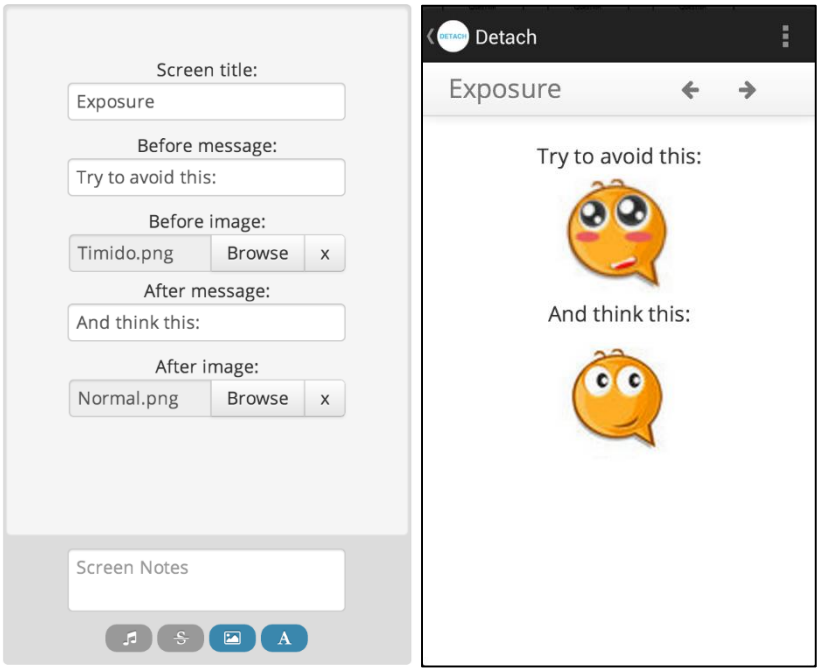

Figure 47 - Example screen editing and respective run-time emulator result

7.4 Summary

Our previous findings with both the participatory design and thinking aloud trials allowed us to build DETACH final product highly user-centered.

Participatory design results allowed us to perceive the importance of the material provided to the users. Created applications are as powerful as the tools we make available. We also identified distinct methods people use to organize and connect application screens. Nevertheless, DETACH used the one that had more followers, were an infinite canvas is provided for people to organize and connect their screens with representative arrows.

Thinking aloud trials allowed us to perceive the different connection patterns people use to create connections that depend on screen answers and environment variables. As with the previous results, DETACH followed the pattern used by most participants where users like to create connections from the origin to the destination screen.

From the developers point of view, we have also build detach in order to be easily maintained and improved in the future by adding new components according to user requirements.

Chapter 8

DETACH Mobile & Emulator

This chapter will focus on the framework that will interpret and run the created applications. This component is present both as an Android mobile application as well as a run-time emulator directly in DETACH interface.

8.1 DETACH Mobile

After a DETACH application is created and assigned to a specific user, that same user just has to download the DETACH mobile application, log in or register with his / her own email and start using the application that is retrieved from the server (Figure 48).

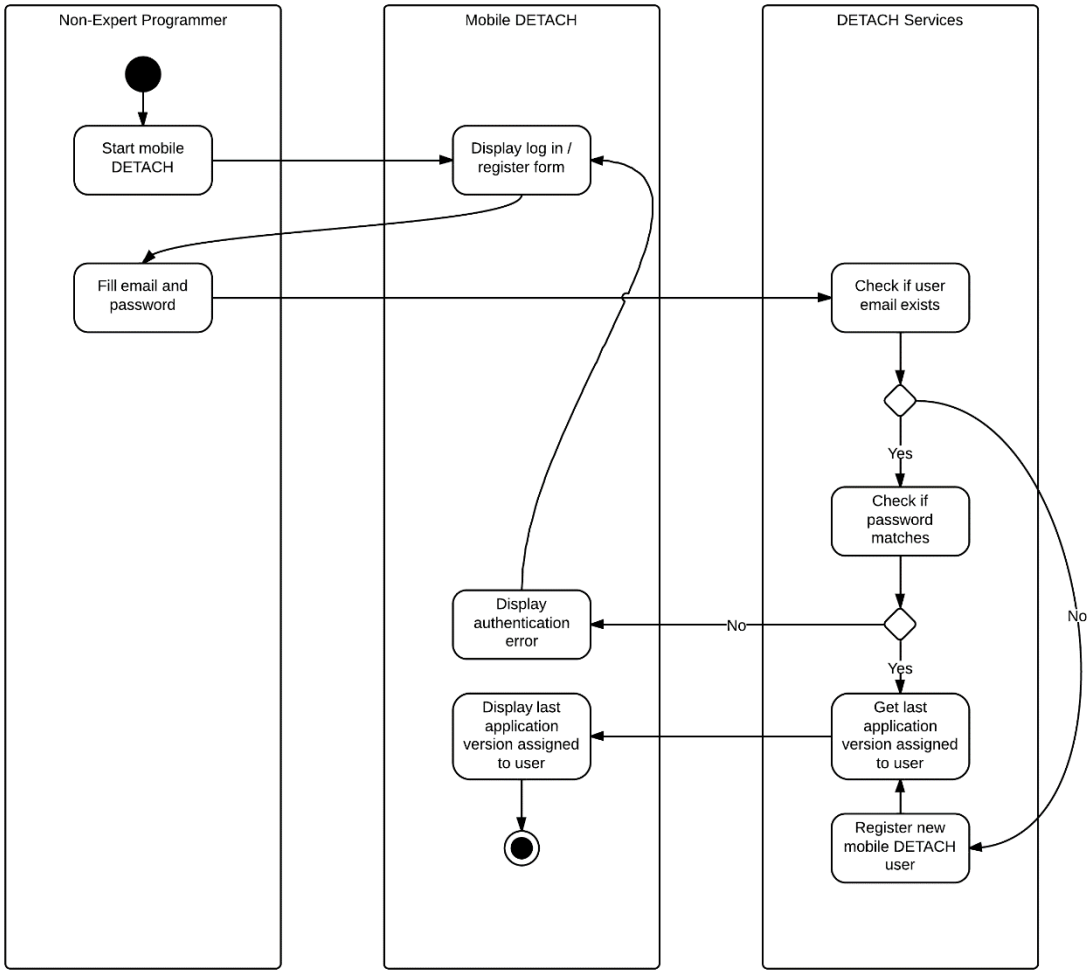

Figure 48 - Mobile DETACH activity diagram

A user can navigate through the application screens back and forward (if allowed by the application creator) to review information or change his / her answers. All that information is appended to a log file the application author will have access. Therefore, it is possible for him / her to review all transitions performed by the patients, along with their hesitations and the answers provided before sticking with a final one. Mobile DETACH requires that all questions have an answer, inhibiting a user from advancing in the application without any answer selected / provided.

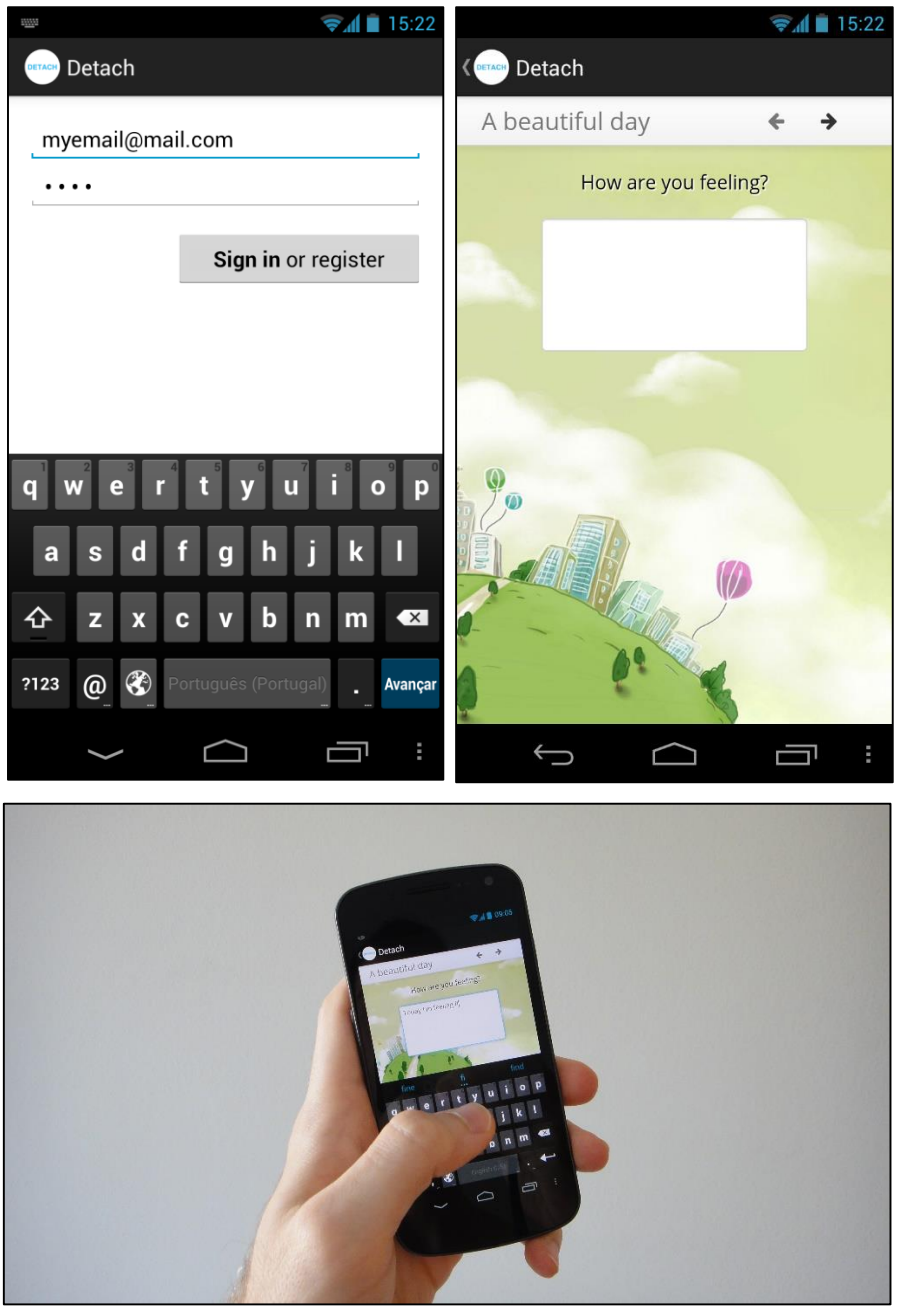

Figure 49 - Mobile DETACH authentication screen (top left), application example (top right) and usage (bottom)

8.2 Run-Time Emulator

An actual requirement identified during the hi-fidelity prototype sessions was the need to quickly preview how a mobile application created with DETACH would look in a smartphone. In the final version we implemented this functionality via an application emulator, allowing a person to test their application's interface, content, sound, styling and behavior in a mobile emulator. The emulator reflects current mid to high-range smartphones' screen dimensions. Android's top navigation bar (represented in the Figure 50 in black), which allowed a user to login into mobile DETACH with a different account, is disabled in the emulator. Additionally, any behavior relying on the device's sensors (e.g. GPS) is not triggered while testing an application with the emulator.

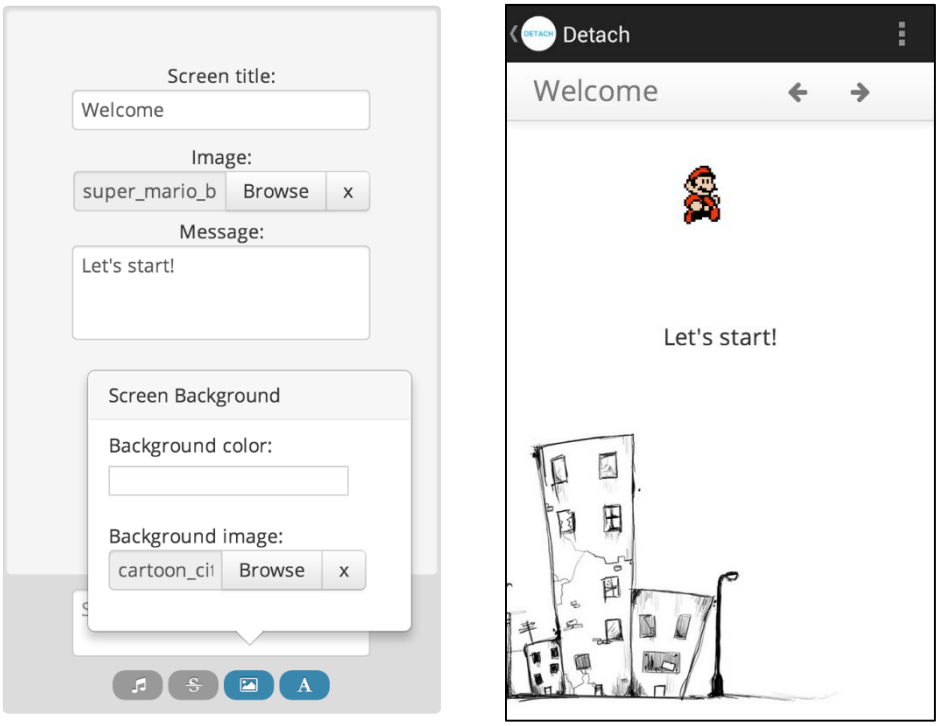

Figure 50 - DETACH example screen configuration and respective run-time result

To emulate an application, their contents have to be retrieved from the server in order to build the added mobile screens as well as to set their connections. Afterwards the user is presented with the created application ready to be used (Figure 51).

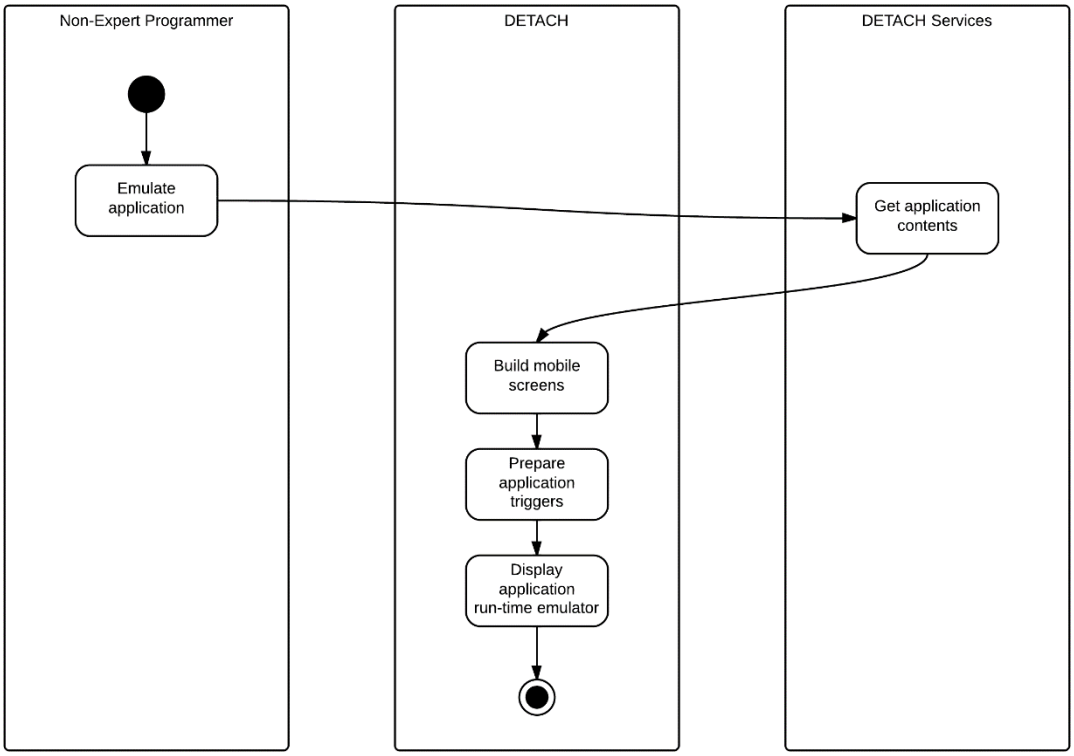

Figure 51 - Emulate application activity diagram

8.3 Summary

In this chapter we have presented the framework that is able to interpret and run DETACH created applications.

This framework is present in DETACH interface as a run-time emulator, as well as in an Android application inside a web view. Such implementation allowed us to reproduce application content in a similar way in both sides.

The Android component simplifies mobile users' lives by automatically loading the last application that was assigned to them as soon as they register / log into the system.

The run-time emulator content tries to mimic a mobile environment by having a representation of a native Android navigation bar. As this last one intend mobile users to go back to the log in / register initial screen, in DETACH interface it was disabled.

Chapter 9

Evaluation

After all the development process this tool was submitted to, there was the need to test the success of our final product.

9.1 Developers

In order to understand if the tool was ready to handle future updates, more specifically, to add new screen templates to the existing list, in this final tests we included a new type of users.

9.1.1 Participants

We reached to 11 participants, both students and employees, with at least the bachelor's degree in one area of Information Technologies. Participants, 10 male and 1 female, had an average of 24 years old (SD= 1.3). All of them were comfortable with both Portuguese and English languages and therefore able to correctly understand the provided tools. A detailed description of the participants is annexed to this document.

9.1.2 Tools & Equipment

In order to observe if the above task was easily understood and reached by IT professionals, we presented them the developers' guide we made available for anyone who wants to add a new template screen to DETACH (annexed to this document).

All the tests were made in a MacBook Pro 15" Retina Laptop with an external mouse attached.

9.1.3 Procedure

In these tests we made participants handle all the components a screen has in the programming code it hides. We therefore asked participants to create a new template screen with:

- One message field, one image field and a field with a number, in this order;
- Two distinct triggers: when the user clicked the button next and when the user clicked the image he just specified.

This way, participants had to deal with the three files each screen is composed by.

We informed participants from the beginning that they weren't being evaluated by their programming skills and that they could ask for help in that sense at any time. Tests were made individually in an illuminated faculty room.

9.1.4 Results

Participants preferred to start from a screen already made than one from scratch and all of them were perfectly comfortable with the first XML file required, as well as setting an appropriate picture for the screen.

When dealing with the Javascript file we however observed that there was still space for improvements. Even though all of the participants were able to add the new template screen in an average time of 35 min. and 24 sec., participants had to take some time to understand the functions they had to use. Even though we described when and how to use them in the provided guide, they would have liked to have a proper API with the descriptions of what exactly each function does. Besides, participants also found quite impressive the amount of information they had to provide to implement the screen triggers, suggesting that some of that code could be executed automatically/reused between functions.

9.2 End-users

In order to prove DETACH success we've conducted a last set of tests with a wide range of users.

9.2.1 Participants

13 users, with an average of 37 years old (SD= 13.4), 7 male and 6 female have participated in these tests. These users did not participate in any of the previous sessions. 11 users had not programming experience, accounting for the primary target demographic of the application. All of them were comfortable with both Portuguese and English languages and therefore able to correctly use this final version of DETACH. A detailed description of the participants is annexed to this document.

9.2.2 Tools & Equipment

These last tests were carried out in a MacBook Pro 15" Retina Laptop with an external mouse attached.

9.2.3 Procedure

We presented participants a script where they had to create an application that made use of all of the main tool functionalities (script annexed to this document). In order for

us to compare results between participants, all the applications had a requirement of a minimum of 9 screens and a maximum of 11. This number was achieved by us when trying to combine the main screens with two output ones. Therefore, this was the reasonable number to create an application in which the participants could use all the screens.

9.2.4 Results

Participants tested the tool and were able to create the script application even without any initial explanation of how the application worked.

- "After a short time I was perfectly comfortable and using the tool efficiently!"

Even though, and in order to use every DETACH main functionality, before finishing each session, we asked the participants to try to set a target screen more efficiently, instead of having to connect every screen to it as they all did. Nevertheless, after watching the tutorials, participants could use the functionality requested and therefore prove the tool help functionality. Even though users could complete the application and learn by themselves, they mentioned it would be even easier and better to present these tutorials right on the application opening.

- "This is so much fun!"

Trials took an average of 42min and 5sec.

- "If I wanted to do it again I would be so much faster now!"

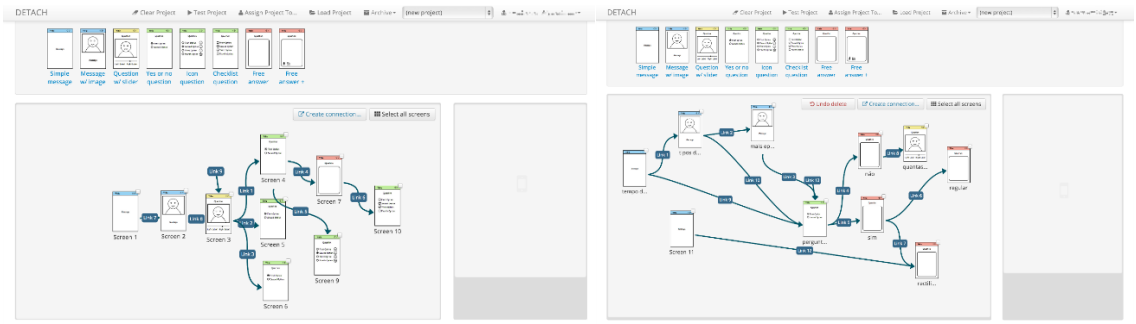

Figure 52 - DETACH final tests with end-users example results

Interface influence on the users

We verified that the order in which the template screens was presented in the tool highly affected their use. Most of the users started with the first question template screen to ask questions that should only be asked in a different type of screen. One of the users just used the first screen template to create the whole application without even checking the remaining ones. Another user created the application using the same order presented in the screen templates, mentioning that that was the order he understood that the screens

had to be created. In the end, we conclude that from that point of view, the tool would be much more user friendly if that first template screen would be the simplest one.

We also verified that the participants perceived that the working space would interpret the containing screens in the order they would be arranged, as they did not feel the need to connect the screens sequentially using rules. This approach was dismissed as they progressed into creating more complex applications with richer transition rules between screens.

Users like to follow their past experiences

Even being able to accomplish the requested task, and realizing how the application worked, users mentioned they would still prefer to have more information about the system, for example about the automatic saving function. Users would actually have preferred to have a manual saving button than knowing that all the information is being saved automatically.

Another functionality the tool tried to made simpler was the authentication one. We provided a unified user login/register form, in order to simplify the user registration process. However, roughly half the users started using it erroneously, expecting the register button to display a new page.

The tool's design (almost reminiscent of a desktop metaphor) also made some participants use the right mouse button click on top of elements in order to find hidden functionalities that might have been available. This behavior is linked with the user's expectations and interactions with their operating system.

We also observed that users prefer using the keyboard to perform some actions (e.g. escape to cancel, holding ctrl/shift to select multiple items and using arrow keys for navigation).

Additional functionalities suggested

Participants mentioned it would be interesting for the tool to have a small library with an initial set of images/sounds they could choose from, instead of just the ones they would have on their computers.

In order to improve efficiency participants also suggested that the connections between screens should have available an option to quickly change their direction. However, this functionality would not make sense as the conditions associated with each connection depend on the origin screen.

9.3 Summary

After building DETACH final product we evaluated it with both end-users and developers.

Although there is still some space for improvements, our final end-users participants were using the tool enthusiastically without any difficulties. After giving them the freedom to create an application that made more sense for them to be used we found out that most of them capitalized on DETACH to create learning applications, especially in the recipes domain. Such a choice allowed us to perceive other DETACH possible uses apart from the health one. As we presented, the tool is also able to create interactive games or story books, to name a few usage examples.

DETACH functionalities can also be improved as shown in the final trials with developers. New template screens or triggers can be added to the authoring tool by these professionals, allowing the tool to offer more and more improvements to their users according to new requirements.

Chapter 10

Conclusions & Future Work

This work started by perceiving users programming concepts in order to build DETACH, a system that comprises: a) a flexible enough platform that allow developers to easily add new components and enables non-programmer users to create powerful mobile applications; b) a framework that runs previously created mobile applications.

We therefore conducted a series of participatory design and thinking aloud trials with non-programmer users aiming to understand how they conceptualized programming. The results of interacting with low and high fidelity prototypes provided us with a set of interaction patterns and behaviors which we capitalized on in order to design the final DETACH product.

These studies behind DETACH development process resulted in the publication of three conference papers: one related to this project health domain (Pervasive Health) and two others related to the human-computer interaction (INTERACT and Interação).

With our final evaluation we proved that DETACH fulfilled all the initial requirements by allowing health professionals, and people with no programming concepts in general, to easy create powerful applications and run them in a mobile environment. On the other hand we have also verified that every developer was able to add a new usable component to the tool.

However, these final evaluations also showed that our authoring tool could still see some future improvements from the end-users point of view, such as:

- The addition of new environment variables (as heartbeat sensor):
  This was specially noted with the participants that created health applications when different types of patient monitoring were required;
- Creation of new template screens:
  New template screens with different content such as pdf files and video was also a request by one of our final participants.

From the developers point of view we have also verified that the amount of code they currently need to create screen triggers could be reduced in order to make this process easier.

The initial framework that was also made available to run the previously created applications also leaves some space for improvements that can encompass:

- Offline DETACH Mobile implementation:
  The possibility for a user to make one initial download of the mobile application assigned to him as well as a final upload of the recorded activity logs for the application creator;
- Export DETACH Mobile to other platforms:
  So that mobile DETACH can be used not only in Android OS smartphones but also in the ones with iOS and Windows Mobile, for example.

Future work also include the execution of some trials with the end-users of our mobile applications.

Chapter 11

Bibliography

[1] A. Smith, "Smartphone Adoption and Usage," Pew Internet and American Life Project, 2011.

[2] J. McGuire, COGNITIVE - BEHAVIOURAL APPROACHES An introduction to theory and research, HM Inspectorate of Probation, 2000.

[3] A. C. Butler, J. E. Chapman, E. M. Forman and A. T. Beck, "The empirical status of cognitive-behavioral therapy: A review of meta-analyses," *Clinical Psychology Review,* vol. 26, pp. 17-31, 2006.

[4] J. A. Cully and A. L. Teten, A Therapist's Guide to Brief Cognitive Behavioral Therapy, Department of Veterans Affairs, South Central Mental Illness Research, Education, and Clinical Center (MIRECC), 2008.

[5] W. Froggatt, "Cognitive-Behaviour Therapy," *Drug and Alcohol Review,* vol. 58, no. 1, p. 95, 2006.

[6] A. K. Das, "Computers in Psychiatry: A Review of Past Programs and an Analysis of Historical Trends," *The Psychiatric quarterly,* vol. 73, pp. 351-365, 2002.

[7] M. A. Grasso, "Clinical Applications of Hand Held Computing," *17th IEEE Symposium on Computer Based Medical Systems*, pp. 141-146, 2004.

[8] E. Mattila, J. Parkka, M. Hermersdorf, J. Kaasinen, J. Vainio, K. Samposalo, J. Merilahti, J. Kolari, M. Kulju, R. Lappalainen and I. Korhonen, "Mobile Diary for Wellness Management - Results on Usage and Usability in Two User Studies," *IEEE Transactions on Information Technology in Biomedicine,* vol. 12, no. 4, pp. 501-512 , 2008.

[9] M. E. Morris, Q. Kathawala, T. K. Leen, E. E. Gorenstein, F. Guilak, M. Labhard and W. Deleeuw, "Mobile Therapy: Case Study Evaluations of a Cell Phone Application for Emotional Self-Awareness," *Journal of Medical Internet Research,* vol. 12, no. 2, p. e10, 2010.

[10] A. Przeworski and M. G. Newman, "Palmtop computer-assisted group therapy for social phobia," *Journal of Clinical Psychology,* vol. 60, no. 2, pp. 179-188, 2004.

[11] M. D. G. Matthews, "In the Mood: Engaging Teenagers in Psychotherapy Using Mobile Phones," *Computers and Society,* pp. 2947-2956, 2011.

[12] F. A. Boujarwah, M. O. Riedl, G. D. Abowd and R. I. Arriaga, "REACT: intelligent authoring of social skills instructional modules for adolescents with high-functioning Autism," *ACM SIGACCESS Accessibility and Computing,* no. 99, pp. 13-23, January 2011.

[13] M. d. Sá and L. Carriço, "Fear therapy for children: a mobile approach," in *EICS '12 Proceedings of the 4th ACM SIGCHI symposium on Engineering interactive computing systems*, New York, NY, USA, 2012.

[14] S. Herman and L. Koran, "In vivo measurement of obsessive-compulsive disorder symptoms using palmtop computers," *Computers in Human Behaviour,* vol. 14, no. 3, p. 449–462, 1998.

[15] S. e. a. Robinson, "Aftercare intervention through text messaging in the treatment of bulimia nervosa - Feasibility pilot," *Intl Journal of Eating Disorders,* vol. 39, no. 8, pp. 633-638, 2006.

[16] D. Offer, K. I. Howard, K. A. Schonert and E. Ostrov, "To whom do adolescents turn for help? Differences between disturbed and nondisturbed adolescents.," *Journal of the American Academy of Child & Adolescent Psychiatry,* vol. 30, no. 4, pp. 623-630, 1991.

[17] N. A. o. C.-B. Therapists, "What is Cognitive-Behavioral Therapy?," [Online]. Available: http://www.nacbt.org/whatiscbt.htm. [Accessed 09 September 2013].

[18] L. Carriço, M. d. Sá, L. Duarte and T. Antunes, "Therapy: Location-aware Assessment and Tasks," in *AH '12 Proceedings of the 3rd Augmented Human International Conference*, 2012.

[19] P. C. M. Paredes, "CalmMeNow: Exploratory Research and Design of Stress Mitigating Mobile Interventions," in *CHI EA '11 CHI '11 Extended Abstracts on Human Factors in Computing Systems*, New York, 2011.

[20] M. Newman, "Technology in Psychotherapy: An Introduction," *Journal of Clinical Psychology,* vol. 60, no. 2, pp. 141-145, 2004.

[21] H. R. J. G. K. Christensen, "The use of e-health applications for anxiety and depression in young people: challenges and solutions," *Early intervention in psychiatry*, vol. 5 Suppl 1, no. June 2010, pp. 58-62, 2011.

[22] M. R. R. &. L. H. Cunningham, "The Cool Teens CD-ROM," *Youth Studies Australia*, vol. 25, no. 1, pp. 50-56, 2006.

[23] M. Cunningham, "Overview of Design Approaches in The Cool Teens CD and Other Computer Programs for Adolescent Anxiety," *E-Journal of Applied Psychology*, vol. 4, no. 2, 2008.

[24] M. W. V. R. R. L. H. S. C. H. J. Cunningham, "The Cool Teens CD-ROM for anxiety disorders in adolescents: A pilot case series," *European Child Adolescent Psychiatry*, vol. 18, no. 2, pp. 125-129, 2009.

[25] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. E. Smith and P. Steggles, Handheld and Ubiquitous Computing, vol. 1707, Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 304-307.

[26] G. Chen and D. Kotz, "A Survey of Context-Aware Mobile Computing Research," *Time*, vol. 3755, no. TR2000-381, pp. 1-16, 2000.

[27] B. N. Schilit and M. M. Theimer, "Disseminating active map information to mobile hosts," *Ieee Network*, vol. 8, no. 5, pp. 22-32, 1994.

[28] P. J. Brown, J. D. Bovey and X. Chen, "Context-aware Applications: from the Laboratory to the Marketplace," *Ieee Personal Communications*, vol. 4, no. 5, pp. 58-64, 1997.

[29] N. Ryan, J. Pascoe and D. Morse, "Enhanced Reality Fieldwork: the Context-Aware Archaeological Assistant," in *Computer Applications in Archaeology*, V. Gaffney, M. v. Leusen and S. Exxon, Eds., Tempus Reparatum, 1998.

[30] A. K. Dey, "Context-aware computing: The CyberDesk project," in *AAAI 1998 Spring Symposium on Intelligent Environments*, 1998.

[31] A. K. Dey and G. D. Abowd, "Towards a Better Understanding of Context and Context-Awareness," *CHI 2000 workshop on the what who where when and how of contextawareness*, vol. 4, no. What, Who, Where, When and How of Context-Awareness, p. 1–6, 1999.

[32] M. Sá, L. Carriço, J. Faria, I. Sá, N. Baloian and G. Zurita, "Geo-Referenced Collaborative Psychotherapy: Design and Evaluation of a Low-fidelity Prototype," in *MobileHCI '10 Proceedings of the 12th international conference on Human computer interaction with mobile devices and services*, New York, 2010.

[33] M. Sá, L. Carriço, J. Neca, N. Fernandes, P. Feiteira, R. Pereira, P. Bernardo, J. Faria and I. Sá, "Ubiquitous Geo-Referenced Social Skills Therapy," in *Proceedings of the 12th ACM international conference adjunct papers on Ubiquitous computing*, New York, 2010.

[34] L. Carriço, L. Duarte, M. d. Sá and T. Antunes, "From Paper to Personalized Digital Forms: Technology in Psychotherapy," in *Workshop on Interaction Design and Emotional Wellbeing at CHI2012*, Austin, Texas, 2012.

[35] L. Cranor and A. Apte, "Programs worth one thousand words: visual languages bring programming to the masses," *Crossroads - Special issue on programming languages*, vol. 1, no. 2, pp. 16-18, December 1994.

[36] I. Parker, "Absolute PowerPoint: Can a software package edit our thoughts?," *The New Yorker*, vol. 77, no. 13, p. 76–87, 2001.

[37] E. Tufte, "The Cognitive Style of PowerPoint," Graphics Press, Cheshire, 2003.

[38] S. R. Klemmer, A. K. Sinha, J. Chen, J. A. Landay, N. Aboobaker and A. Wang, "Suede: a Wizard of Oz prototyping tool for speech user interfaces," *Proceedings of the 13th annual ACM symposium on User interface software and technology*, vol. 2, pp. 1-10, 2000.

[39] J. A. Landay, "SILK: sketching interfaces like krazy," in *Conference companion on Human factors in computing systems common ground*, 1996.

[40] J. Lin, M. Newman, J. I. Hong and J. A. Landay, *DENIM: An Informal Sketch-based Tool for Early Stage Web Design*, 2002.

[41] J. Lin, M. W. Newman, J. I. Hong and J. A. Landay, "DENIM: An Informal Tool for Early Stage Web Site Design," in *Proceedings of CHI 2001 ACM Conference on Human Factors in Computing Systems*, New York, 2001.

[42] V. C. V. B. Segura and S. D. J. Barbosa, "Shape-based versus sketch-based UI prototyping: a comparative study," in *Proceedings of the 10th*

Brazilian Symposium on on Human Factors in Computing Systems and the 5th Latin American Conference on HumanComputer Interaction, 2011.

[43] V. C. V. B. Segura, S. D. J. Barbosa and F. P. Simões, "UISKEI: a sketch-based prototyping tool for defining and evaluating user interface behavior," in *AVI '12 Proceedings of the International Working Conference on Advanced Visual Interfaces*, New York, 2012.

[44] M. D. Sá, L. Carriço, L. Duarte and T. Reis, "A Mixed-Fidelity Prototyping Tool for Mobile Devices," in *AVI '08 Proceedings of the working conference on Advanced visual interfaces*, New York, NY, USA, 2008.

[45] J. Maloney, K. Peppler, Y. B. Kafai, M. Resnick and N. Rusk, "Programming by choice: urban youth learning programming with scratch," *Computer Science Education*, vol. 40, no. 1, pp. 367-371, 2008.

[46] J. Maloney, M. Resnick, N. Rusk, B. Silverman and E. Eastmond, "The Scratch Programming Language and Environment," *ACM Transactions on Computing Education*, vol. 10, no. 4, pp. 1-15, 2010.

[47] D. J. Malan and H. H. Leitner, "Scratch for budding computer scientists," *ACM SIGCSE Bulletin*, vol. 39, no. 1, p. 223, 2007.

[48] B. Bergvall-Kåreborn and A. Ståhlbrost, "Participatory design: one step back or two steps forward?," in *Design*, 2008.

[49] K. Moffatt, J. McGrenere, B. Purves and M. Klawe, "The participatory design of a sound and image enhanced daily planner for people with aphasia," *Proceedings of the 2004 conference on Human factors in computing systems CHI 04*, vol. 6, no. 1, pp. 407-414, 2004.

[50] L. Westfall, "Software Requirements Engineering: What, Why, Who, When, and How By Linda Westfall," *ASQs Software Quality Professional Journal*, no. 2004, p. 9–15, 2006.

[51] B. Nuseibeh and S. Easterbrook, "Requirements Engineering : A Roadmap," *Context*, vol. 1, pp. 35-46, 2000.

[52] M. D. S. M. D. Sa, L. C. L. Carrico and P. A. P. Antunes, "Ubiquitous Psychotherapy," *Ieee Pervasive Computing*, vol. 6, no. 1, pp. 20-27, 2007.

[53] M. De Sá, L. Carriço, L. Duarte and T. Reis, "Multi-purpose proactive m-Artifacts," *Symposium on Applied Computing*, p. 1629, 2008.

Chapter 12

Annexes

12.1 Participatory Design Participants Details

User #	Gender	Age	Working Area
01	Male	51	Insurance Professional
02	Female	45	Eye Doctor
03	Male	28	Eye Doctor
04	Female	35	Chemistry Engineer
05	Male	36	Businessman
06	Female	48	Designer
07	Male	46	Mobile Communications Engineer
08	Female	41	Psychologist
09	Female	28	Nurse
10	Female	38	Project Coordinator
11	Male	38	Salesman
12	Male	54	Marketing Assistant
13	Male	62	Medicine
14	Male	44	Diet and Nutrition
15	Female	31	Cardiopneumology
16	Female	50	Clinic analysis
17	Female	30	Radiotherapy
18	Female	49	Cardiopneumology
19	Female	56	Clinic analysis
20	Female	28	Cardiopneumology
21	Male	35	Cardiopneumology

22	Male	45	Physiotherapy
23	Male	49	Orthopedics
24	Female	57	Medicine
25	Male	62	Medicine
26	Female	57	Medicine
27	Female	26	Social assistance
28	Female	30	Social assistance

Table 3 - Participatory Design participants description

12.2 Participatory Design Presented Guidelines

Screen examples to show in the patients' phone (1):

- Message;
- Small animation;
- Screen for the patient to inform how he feels, where there's options to select (2).

Represent how would you say to the application which screen would it show in the patients' phone, and when or through what order, if you would have to make them vary according to some data, such as (3):

- Location;
- Heartbeat;
- Blood pressure;
- Time (in a specific screen/since the first screen showed up/time of the day/relative time).
- Possible user answers in previous screens (2);

Try to combine 1 + 2 + 3 as much as you can.

12.3 Participatory Design Presented Script

Imagine that you are a therapist wanting to help a patient that has hospitals phobia. Using the presented material, stick and/or draw whatever you feel the need to, in order to represent the following application your patient could be carrying in his mobile phone:

Begin your application by showing your patient a positive message in case his heartbeat is normal (consider values lower than 100bpm). Otherwise show him his current heartbeat levels.

In case you showed him a positive message, if he takes more than a minute to press any of the presented buttons try to understand what's wrong by showing him a list of emotions he can choose from. If he answers that he's happy, show him a positive animation.

In case his heartbeat was higher than 100bpm, if he's closer than 5 meters to a hospital, display the same list of emotions in order to know how he's feeling. If he takes more than 10 minutes to select an answer display a negative animation.

12.4 Thinking aloud presented script

Nome | Idade | Contacto | Área de especialização

Cenário: A Cláudia é uma rapariga de 26 anos que sempre viveu em Lisboa. Terminou agora os seus estudos e perante o cenário do país, decide tentar a sua sorte em Nova York. Infelizmente tem que enfrentar o seu maior medo: andar de avião.

Como seu/sua terapeuta, cabe-lhe a si ajudar a Cláudia a encarrar o aeroporto de Lisboa como qualquer outra zona banal.

Para isso, e visto que o voo da Cláudia é às 6h da manhã, terá a possibilidade de criar uma aplicação destinada ao telemóvel dela, que se portará como se fosse… você mesmo/a! Assim, poderá dizer à priori a esta aplicação como deverá ajudar a Cláudia, em todas as situações com que ela se possa deparar, continuando você descansadamente a dormir. Lembre-se portanto que terá que definir todos os casos de ajuda neste momento.

Para tal, use a interface fornecida, que indicará uma série de passos a executar quando a Cláudia chegar ao aeroporto, para:

1. Começar por encorajá-la com uma frase animadora;
2. Após 30 segundos tente perceber como ela se encontra, dando-lhe várias opções de escolha: tranquila, impaciente ou atrasada para o seu voo;
3. Para facilitar referências futuras, dê um nome a este ecrã;
4. Não queremos atrasar a paciente e portanto, caso a paciente tenha seleccionado a opção "atrasada", deseje-lhe apenas boa viagem;
5. No caso do estado da paciente ser de tranquilidade, pergunte-lhe apenas se prefere tentar continuar sozinha a sua viagem;
6. Se a resposta for positiva, apresente-lhe uma animação, juntamente com música de fundo (não necessita de escolher o ficheiro);
7. No caso de uma resposta negativa diga que a aplicação deve voltar ao início;
8. Já que voltámos atrás, redefinimos a condição inicial dos 30 segundos para 45;
9. Se a Cláudia estiver impaciente tente registar os seus pensamentos através de texto e áudio;
10. Após este registo apresente a mesma animação que criou anteriormente;
11. Esta mesma animação deverá também ser apresentada em qualquer momento que o sensor de batimentos cardíacos ligado à Cláudia e ao seu *smartphone* indique que os mesmos ultrapassaram os 120bpm.

Questões:

- Ao ter criado setas entre ecrãs, o que representam essas setas para si?
- É lógico para si a representação de transições que a qualquer momento mostrem um dado ecrã? Se não, como preferiria?
- E quando quis apagar um ecrã? Foi óbvio?
- Outras opiniões?

Obrigado ☺

12.5 Thinking aloud first phase participants details

User #	Gender	Age	Working Area
01	Female	22	Clinical Psychology
02	Female	23	Organizational Psychology
03	Male	21	Music
04	Female	49	Journalism
05	Male	23	Dental hygiene
06	Female	24	Architecture
07	Female	23	Occupational therapy
08	Male	44	Sports
09	Female	49	Communications calling center
10	Female	39	Project Management
11	Male	51	Insurance Department Management

Table 4 - Thinking Aloud first phase participants description

12.6 Thinking aloud second phase participants details

User #	Gender	Age	Working Area
01	Female	35	Business Project Management
02	Female	24	Account Management
03	Female	26	Design
04	Male	24	Mechanics
05	Male	24	Mechanics
06	Female	26	Digital Marketing Management
07	Male	23	Business Management

Table 5 - Thinking Aloud second phase participants description

12.7 Thinking aloud resulting task times

Below are the times spent in each task. Notice the times in red are representing tasks that the users couldn't complete alone. They are therefore the time they actually tried to complete it.

FIRST PHASE												
USER	T01	T02	T03	T04	T05	T06	T07	T08	T09	T10	T11	Total
01	04m:55s	05m:16s	00m:28s	03m:44s	04m:22s	01m:43s	01m:34s	01m:24s	00m:47s	01m:00s	02m:10s	27m:23s
02	10m:05s	07m:56s	00m:43s	03m:32s	04m:04s	01m:34s	02m:30s	00m:36s	01m:20s	01m:50s	01m:46s	35m:56s
03	04m:54s	05m:00s	00m:26s	04m:42s	00m:33s	03m:12s	02m:38s	00m:35s	01m:33s	00m:59s	00m:18s	24m:50s
04	05m:10s	02m:34s	01m:20s	00m:18s	02m:33s	01m:22s	01m:20s	00m:35s	01m:52s	00m:43s	01m:20s	19m:07s
05	04m:30s	04m:50s	02m:00s	00m:55s	04m:15s	00m:55s	01m:55s	00m:40s	01m:20s	00m:32s	02m:49s	24m:41s
06	02m:14s	06m:10s	00m:24s	01m:14s	03m:23s	02m:50s	02m:42s	00m:55s	01m:50s	01m:55s	05m:04s	28m:41s
07	01m:33s	08m:15s	00m:19s	03m:04s	06m:59s	03m:33s	02m:30s	00m:24s	03m:41s	00m:52s	03m:50s	35m:00s
08	00m:24s	06m:12s	01m:00s	03m:00s	03m:44s	02m:15s	00m:32s	01m:21s	01m:10s	01m:00s	03m:11s	23m:49s
09	02m:39s	07m:27s	00m:28s	03m:06s	01m:57s	02m:27s	01m:20s	02m:53s	01m:40s	01m:25s	04m:20s	29m:42s
10	01m:15s	05m:14s	00m:30s	01m:12s	03m:06s	01m:40s	01m:32s	01m:33s	00m:40s	01m:25s	03m:10s	21m:17s
11	02m:35s	08m:05s	00m:29s	13m:46s	02m:23s	01m:55s	02m:57s	02m:48s	02m:00s	00m:29s	04m:02s	41m:29s
AVG												28m:21s

Table 6 - Thinking Aloud first phase resulting task times

SECOND PHASE												
USER	T01	T02	T03	T04	T05	T06	T07	T08	T09	T10	T11	Total
01	00m:57s	04m:40s	00m:39s	02m:00s	00m:57s	01m:12s	00m:32s	00m:10s	00m:59s	00m:36s	02m:48s	15m:30s
02	01m:08s	04m:41s	00m:40s	01m:31s	01m:06s	02m:02s	00m:19s	00m:14s	00m:37s	00m:35s	01m:44s	14m:37s
03	01m:42s	04m:00s	00m:28s	03m:26s	03m:18s	01m:21s	00m:26s	01m:34s	02m:06s	00m:30s	02m:35s	21m:26s
04	02m:20s	05m:27s	00m:08s	00m:53s	00m:37s	00m:41s	00m:44s	01m:20s	01m:00s	00m:21s	03m:29s	17m:00s
05	00m:42s	05m:05s	00m:18s	01m:45s	00m:35s	00m:26s	00m:13s	00m:28s	00m:54s	00m:17s	01m:42s	12m:25s
06	02m:26s	04m:01s	01m:05s	02m:34s	03m:26s	01m:14s	00m:21s	00m:32s	00m:34s	01m:44s	01m:03s	19m:00s
07	00m:45s	03m:29s	00m:33s	01m:23s	01m:09s	01m:21s	00m:14s	00m:38s	00m:41s	00m:13s	01m:36s	12m:02s
AVG												16m:00s

Table 7 - Thinking Aloud second phase resulting task times

12.8 DETACH User Requirements

The functional requirements (FR) define the capabilities of the software product (what the software must do to add value for its stakeholders).

The non-functional requirements (NFR) define the characteristics, properties, or qualities that the software product must possess. They define how well the product performs its functions (what the software must be to add value for its stakeholders) [50].

FR 01	*User authentication*
Description	DETACH has to maintain a database of their users so it can register a new user or recognize an old one

FR 02	*Application screen adding*
Description	DETACH has to offer the possibility to add new mobile screens to an application
NFR 02.1	DETACH should offer a set of very simple screens each with a minimal set of input/output capabilities

NFR 02.2	Screens with animations or even subliminal messages were some of the propositions from therapists meetings
NFR 02.3	From related work it's also important that the tool offers a set of template screens with different designs, similarly to Microsoft PowerPoint different slide templates [52] [13] [53]

FR 03	*Application screen filling*
Description	The process of introducing content in a mobile application screen
NFR 03.1	The user must be able to input basic content as information and questions
NFR 03.2	The content can be in the form of text, image, animation and/or sound
NFR 03.3	Content freedom should be available to users and not limited to a small set of media

FR 04	*Application screen styling*
Description	The process of customizing a mobile screen
NFR 04.1	The user must be able to change a mobile screen looks based on the person who is going to use it (for example, increase text size for older people)

FR 05	*Application navigation specification*
Description	The process of linking the different mobile application content
NFR 05.1	The user must be able to change the mobile application content based on screen outputs and context information. Some suggestions include tracking the time of the day or the patient's location/heartbeat for contextual proactivity

FR 06	*Application storage*
Description	The process of store mobile applications for later use

FR 07	*Application user linking*
Description	The process of making a mobile application being associated with a specific person

FR 08	*Application loading*
Description	The process of loading a previously saved mobile application
NFR 08.1	The user can only load mobile applications that were created by himself

FR 09	*Application modification*
Description	The process of modifying a previously created application

FR 10	*User activity log viewer*
Description	The process of viewing the activity performed by the person who used that application
NFR 10.1	The user should be able to access information such as answers given to asked questions

12.9 DETACH Developer Requirements

FR 11	*DETACH modules extension*
Description	The process of being able to code new modules for DETACH
NFR 11.1	DETACH should offer ways to code new modules in an efficient way

12.10 Mobile DETACH User Requirements

FR 12	*User authentication*
Description	The mobile application should be able to authenticate the person who is using it

FR 13	*Application loading*
Description	The mobile application should load the correct application for the authenticated user

FR 14	*Activity logging*
Description	The mobile application should record the actions the person who is using the application makes
NFR 14.1	The mobile application should record user actions such as answers provided to questions
NFR 14.2	The mobile applications created should register by themselves user actions to be reviewed later by their therapists

FR 15	*Interpret actions*
Description	The mobile application should be able to interpret user actions
NFR 15.1	User actions such as answers and clicks should be analyzed in order to display the next application content

FR 16	*Context information access*
Description	The mobile application should access context information
NFR 16.1	Contextual information such as GPS location should be accessed for applications that require it

12.11 DETACH for developers guide

DETACH for Devolopers

How to add a screen template to DETACH

1 | Where to start:

Go to the project folder "screens" and find a name for your screen that's not already assigned to any screen in the folder (let's call it "BlinkingMessage"). This name will only be internal and therefore not visible to the end user.

2 | Create 3 files in this folder:

BlinkingMessage.xml
BlinkingMessage.js
BlinkingMessage.png

3 | Bear in mind:

The .xml file will contain data associated with the fields the user must fill in that screen
The .js file will contain data associated to how the screen will be generated giving those fields
The .png file will contain the DETACH image representation of that screen. For better representation the image should have 174 x 267 pixels. If you want to follow the other image representations use the Balsamiq Mockups or Swordsoft Layout software to design the screen and a color tool to match the screen to it's respective type.

4 | Fill the .xml file:

The .xml file must contain the following tags:

```
<screen>
    <type>The type you want to show to the end user (below the screen image representation)</type>
    <description>The screen description that appears when the user stops his mouse pointer over the screen image representation</description>
    <fields>
        ... (see explanation below)
    </fields>
    <triggers>
        ... (see explanation below)
    </triggers>
</screen>
```

5 | The .xml <fields> tag:

This tag contains all the fields the user can fill when configuring this screen.
Do NOT create the field corresponding to the screen title as this field is created automatically!
The fields can have the following types:

- Text (Represented by a single fillable line that cannot have line breaks)
- Textarea (Represented by a multiple fillable line that can have line breaks)
- Number (Represented by a single fillable line only accepting numbers)
- SingleFile (Allowing the user to choose one file)
- MultipleFile (Allowing the user to choose multiple file)

The <fields> tag can have as many fields as you want, represented by the type as tag, and by the field description as tag content.
See the following example:

```
<fields>
    <SingleFile>Image</SingleFile>
    <Textarea>Message</Textarea>
</fields>
```

6 | The .xml <triggers> tag:

This tag contains all the triggers that can make this screen transit to another.
Do NOT create the triggers corresponding to the global variables, as time and user location, as these triggers are created automatically! Create only the triggers you will have to program in the BlinkingMessage.js file. For example: on next button click, on image click, when first option selected, etc.
Add the triggers like this:

```
<triggers>
    <trigger id="1">First Option Selected</trigger>
    <trigger id="2">Second Option Selected</trigger>
</triggers>
```

Notice the triggers must have an id associated, which will allow you to identify the trigger you want to program in the BlinkingMessage.js file.
You can also create a more complex trigger, that can allow the user to choose from a dropdown menu, for example, has/has not a specific value in a screen answer. For that, after the complex trigger line add a Select tag with the options you want to make available. If an option allows the user to input extra data, you can also add the Text or Number tag to it with the appropriate description to the user. See the following example for better understanding:

```
<triggers>
    <trigger id="1">Answer
        <Select>
            <Option>has any text</Option>
            <Option>contains</Option>
                <Text>text</Text>
            <Option>does not contain</Option>
                <Text>text</Text>
        </Select>
    </trigger>
</triggers>
```

If you have more than one trigger make the first one (id=1) and its first select option (if there's a select) the default, as this will be chosen for automatic created connections. As you can see in the above case, if an automatic connection is created, it will be executed when the user answer as any text, as the automatic connections don't allow the specification of any trigger contents (in this case the text we could want to verify. If created automatically it would have the empty value).

7 | Fill the .js file:

The .js file must contain the code to generate the screen and the code that will interpret every screen trigger.
You have to create the following functions:

function new*Screen(div,screenId,screenContents) - replace * with the name you gave to the screen, in this case BlinkingMessage, so in the end: newBlinkingMessageScreen
function set*Trigger1(screenId,eventCondition,eventData,eventDest,allowbackcondition) - replace * with the name you gave to the screen, in this case BlinkingMessage, so in the end: setBlinkingMessageTrigger1

Create as many of this last function as the triggers defined in the .xml file, making the correspondence of the function number to the trigger id number.
For example: if you defined <trigger id="1">...</trigger><trigger id="2">...</trigger> you should create the functions **setBlinkingMessageTrigger1** and **setBlinkingMessageTrigger2**

Now, let's see how to code each function:

function new*Screen
Parameters:
- div (the screen itself, where you'll have to put all the elements you want)
- screenId (the id this screen will have)
- screenContents (an array with each position associated to the users filled values: screenContents[0] corresponds to the screen filled title, screenContents[1] corresponds to the filled first field you defined in the .xml file, screenContents[2] to the second, etc.)

Returning value:
(none)

Add the screen elements you want to the div parameter. When creating screen elements, define their id with a combination of a string and the screenId so that there's no multiple screens with the same elements id (ex: obj.id = screenId+"radioTop"). This id is useful to refer to the element if you are gonna need it when programming the trigger.

Don't forget to include the top navigation bar present in all screens. To do this call the function addTopNavigationBar(theScreenDiv,theScreenTitle,theNavigationBarNextButtonId) before adding anything else to the screen. In the end this is the code you should begin your function with:

```
function new*Screen(div,screenId,screenContents){

    addTopNavigationBar(div,screenContents[0],"next"+screenId);
    ...

}
```

function set*Trigger1
Parameters:
- screenId (the id this screen will have)
- eventCondition (mentioning if the connection associated with this trigger was created matching "Any" or "All" of the connection conditions)
- eventData (a string with content if this trigger contains a select dropdown menu. If that's the case this string contains: the select selected value followed by an underscore and, if available, the filled value of that select input. Example: "contains_book")
- eventDest (the screen id to transit to if the connection associated with this trigger is executed)
- allowbackcondition ("true" or "false" if the connection associated with this trigger allows the user to return to the previous screen)

Returning value:
(none)

Don't forget to add the user inputs, in this case associated with this trigger, in the users logs. For that use the function appendToLog(stringToAppend). If you want you can access the time on screen value with the global variable "screentime".

```
appendToLog("Clicked next at "+screentime+"sec.");
```

Also don't forget to only activate the eventDest screen according to the eventCondition. You should use the functions markThisConditionAsVerified and markThisConditionAsNotVerified if the condition in this trigger is verified or not, and, afterwards, if eventCondition!="Any", verify if all of the conditions to activate the eventDest screen are verified. This verification is made with the function checkIfAllConditionsChecked. All of these functions have the connection description as the parameter, in the shape "screenId-eventDest". markThisConditionAsVerified and markThisConditionAsNotVerified, besides that parameter also have to include the global variable "connectConditionCount" that identifies the condition number in that connection. See the example:

```
markThisConditionAsVerified(screenId+"-"+eventDest,connectConditionCount);

if(eventCondition=="Any" || checkIfAllConditionsChecked(screenId+"-"+eventDest)){
    ...
}
```

Finally activate the eventDest screen like this:

```
deactivateAllScreens();
activateScreen(eventDest,allowbackcondition);
```

Here's an example of a complete function of this type that, for a screen with radio buttons, activates the next screen when the first one is checked:

```
function set*Trigger1(screenId,eventCondition,eventData,eventDest,allowbackcondition){

    document.getElementById("next"+screenId).addEventListener('click',function(arg1,arg2,count,back) {
        return function() {

            appendToLog("Clicked next at "+screentime+"sec.");

            if(document.getElementById("radioTop" + screenId).checked) {

                markThisConditionAsVerified(screenId+"-"+arg2,count);

                if(arg1=="Any" || checkIfAllConditionsChecked(screenId+"-"+arg2)){
                    deactivateAllScreens();
                    activateScreen(arg2,back);
                }else{
                    markThisConditionAsNotVerified(screenId+"-"+arg2,count);
                }

            }else if(!document.getElementById("radioBottom" + screenId).checked) {

                //Code to warn the user an answer is required

            }

        };
    }(eventCondition,eventDest,connectConditionCount,allowbackcondition),false);

}
```


12.1 Final DETACH evaluation developers participants and times spent

User #	Gender	Age	Working Area	Time
01	Male	21	IT student	20m00s
02	Male	24	Android developer	36m50s
03	Male	23	Android developer	41m00s
04	Male	24	IT student	41m10s
05	Male	24	Web developer	31m24s
06	Male	24	IT student	31m50s
07	Female	26	Web developer	44m00s
08	Male	25	Android developer	36m00s
09	Male	23	IT researcher	50m55s
10	Male	23	Java developer	24m40s
11	Male	23	Web developer	31m30s

Table 8 - Final DETACH developers trials participants description and times spent

12.2 Final DETACH evaluation end-user guide

Imagine que é um professor numa instituição privada em que as turmas são de 5 elementos. Tem ao seu dispor a ferramenta DETACH ("DEsign Tool for smartphone Application Composition"), que vai hoje poder usar, para, numa das suas turmas, inovar no sistema de ensino e disponibilizar a esses mesmos alunos **suporte ao estudos através de uma aplicação móvel**.

A aplicação deve começar por avisar o aluno que este apenas dispõe de 30min. para estudar a matéria apresentada na aplicação. De seguida terá acesso a um pequeno questionário para averiguação dos conhecimentos adquiridos. No total a sua aplicação deve conter **no mínimo 9 e no máximo 11 ecrãs** distintos.

Para transmitir os conhecimentos a aplicação deve fazer uso não só de **descrições textuais, como sons, imagens e animações**.

O questionário, **activado após 30 min. do início da aplicação**, ou mal o aluno tenha chegado ao fim da visualização da matéria, deve apresentar vários tipos de perguntas, pelo menos:

- Perguntas de sim/não
- Perguntas de resposta aberta

Para cada uma destas perguntas, consoante a resposta do utilizador, **a aplicação deve tomar diferentes rumos** (ex: Uma pergunta sobre o corpo humano em geral, caso correctamente respondida, deve apresentar uma segunda sobre um detalhe particular do corpo humano, caso contrário deve abordar outra temática).

Todas estas perguntas deverão ter um **tempo limite de 5 seg. de resposta** e portanto, logicamente, não deverão permitir que o aluno recue nas mesmas (ao contrário do que acontece no principio da aplicação ao ser transmitida a matéria).

No final da criação da aplicação, teste a mesma e atribua-a aos seus alunos desta cadeira. Imagine por exemplo que os mesmos são os seguintes:

André Matos – 24anos – am@gmail.com

Cátia Silva – 23anos – cs@gmail.com

Bruno Fernandes – 25anos – bf@gmail.com

Sara Barroso – 24anos – sb@gmail.com

12.3 Final DETACH evaluation end-user participants and times spent

User #	Gender	Age	Working Area	Time
01	Female	26	Designer	54m45s
02	Female	24	Architect	36m00s
03	Female	45	Architect	37m00s
04	Male	46	Freelancer photographer	46m30s
05	Male	23	Tax consultant	40m25s
06	Male	42	Architect	35m20s
07	Female	54	Mathematics teacher	37m30s
08	Male	48	Geodesy teacher	38m35s
09	Female	23	Shopping advisor	40m40s
10	Female	53	IT teacher	48m00s
11	Male	25	Architect	33m30s
12	Male	23	Mobile developer	38m20s
13	Male	54	Product manager	60m30s

Table 9 - Final DETACH end-user trials participants description and times spent