UNIVERSIDADE DE LISBOA

FACULDADE DE CIÊNCIAS

DEPARTAMENTO DE INFORMÁTICA

Solução de Cibersegurança através de dispositivos móveis

Diogo Henriques Silva

Mestrado em Engenharia Informática
Especialização em Sistemas de Informação

Versão Pública

Trabalho de Projeto orientado por:
Professora Doutora Daniela Patrícia dos Santos Oliveira

2021

Agradecimentos

Foram dois anos de trabalho árduo que chegaram ao fim. É gratificante saber que concluí esta etapa com o apoio de várias pessoas.

À minha orientadora, Professora Doutora Daniela Oliveira, agradeço pela ajuda e orientação neste projeto.

Na Emvenci Business Services, quero agradecer ao Alexandre Aniceto pela oportunidade de participar neste projeto e pela atenção que teve comigo durante o mesmo. Agradeço também aos meus colegas na empresa, especialmente, ao André Gomes e à Mariana Santos, pela ajuda e companheirismo.

Um agradecimento muito especial aos meus pais, Vítor Silva e Ana Paula Silva, por me terem incentivado a fazer este Mestrado. Por todo o seu apoio, não só nesta fase, mas durante toda a minha vida.

Quero agradecer à minha namorada, Inês Fonseca, pelo amor e por me encorajar sempre a fazer mais e melhor.

Por último, agradeço aos meus amigos que em momentos de algum “desespero” me deram aquele empurrão que me permitiu terminar este projeto.

Muito obrigado a todos!

Para a minha família.

Resumo

A plataforma Cybersecurity Cloud é uma solução SaaS (*Software as a Service*), desenvolvida pela empresa Emvenci Business Services, projetada para ajudar o utilizador a obter informações importantes e melhorar as respostas de cibersegurança, através da integração de diferentes módulos que se focam em pontos como a gestão de incidentes e vulnerabilidades, formação e treino nas áreas de segurança de informação, agregação e transformação de análises de dados para rápida consulta, interpretação e decisão, partilha de informação crucial e compatibilidade com o RGPD.

Com o aumento do uso e dependência de dispositivos móveis o imediatismo de informação é potenciado, e quando falamos em questões de segurança esse ponto é precioso para evitar ou resolver rapidamente possíveis problemas. Atualmente, a plataforma não está otimizada e em certos casos não é suportada, em dispositivos móveis na sua versão *web*, o que limita o seu uso e acaba por afetar esse imediatismo na medida em que os computadores não possibilitam a mesma portabilidade e acessibilidade em comparação com dispositivos móveis.

O trabalho passa por desenvolver uma aplicação móvel baseada na plataforma *web* já existente, usando não só a tecnologia da mesma, Angular, como integrando a *framework* NativeScript que possibilita o desenvolvimento de aplicações nativas para iOS e Android. A solução móvel terá como objetivo possibilitar que utilizador reveja rapidamente as suas tarefas, que seja notificado de informações importantes a qualquer momento, potenciando tomadas de decisão mais eficientes e que aceda a toda a informação fornecida pela plataforma de uma forma mais dinâmica e imediata, sem a necessidade de ter um computador por perto.

Palavras-chave: Angular, NativeScript, cibersegurança, aplicação móvel, dispositivos móveis

Abstract

The Cybersecurity Cloud platform is a SaaS (Software as a Service) solution, developed by Emvenci Business Services, designed to help the user to obtain important information and improve cybersecurity responses, through the integration of different modules that focus on points such as the management of incidents and vulnerabilities, education and training in the areas of information security, aggregation and transformation of data analysis for quick consultation, interpretation and decision, sharing of crucial information and compatibility with the GDPR.

With the increase in the use and dependence of mobile devices, the immediacy of information is enhanced, and when we talk about security issues, this point is precious to avoid or quickly solve possible problems. Currently, the platform is not optimized and in some cases it is not supported, on mobile devices in its web version, which limits its use and ends up affecting this immediacy insofar as computers do not allow the same portability and accessibility in comparison with mobile devices.

The work involves developing a mobile application based on the existing web platform, using not only its technology, Angular, but also integrating the NativeScript framework that enables the development of native applications for iOS and Android. The mobile solution will aim to enable the user to quickly review their tasks, to be notified of important information at any time, enabling more efficient decision making and to access all the information provided by the platform in a more dynamic and immediate way, without the need to have a computer close by.

Keywords: Angular, NativeScript, cybersecurity, mobile application, mobile devices

Conteúdo

Lista de figuras viii

Lista de tabelas x

Abreviaturas xii

Capítulo 1 Introdução 1

1.1 Instituição de acolhimento 2

1.2 Motivação 2

1.3 Objetivos 3

1.4 Estrutura do documento 3

Capítulo 2 Conceitos e definições 5

2.1 Abordagens de desenvolvimento móvel 5

2.1.1 Abordagem nativa 5

2.1.2 Abordagem *web* 6

2.1.3 Abordagem híbrida 6

2.1.4 Comparação entre abordagens 7

2.2 Angular 8

2.3 *Frameworks* para desenvolvimento móvel 9

2.3.1 React Native 9

2.3.2 Ionic 9

2.3.3 Xamarin 10

2.3.4 Flutter 10

2.3.5 NativeScript 10

2.4 Padrões de arquitetura 11

2.4.1 Model-View-Controller (MVC) 11

2.4.2 Model-View-Presenter (MVP) 12

2.4.3 Model-View-View Model (MVVM) 12

2.5 REST e SOAP 12

2.5.1 REST 12

2.5.2 SOAP 13
2.6 Metodologias de desenvolvimento de software 13
2.6.1 Metodologias tradicionais 13
2.6.2 Metodologias *Agile* 14
2.6.3 Comparação entre metodologias 14
2.7 Arquitetura *multi-tenancy* 15
2.8 *Polling* e *long polling* 16
Capítulo 3 Análise de requisitos 19
3.1 Introdução à plataforma 19
3.2 Requisitos funcionais 21
3.3 Requisitos não-funcionais 21
3.4 *User stories* 21
Capítulo 4 Desenho da solução 22
4.1 Tecnologias utilizadas 22
4.1.1 Angular 22
4.1.2 NativeScript 22
4.1.3 Cybersecurity Cloud API 23
4.1.4 Intercom 23
4.2 Arquitetura 24
4.3 Componentes 24
Capítulo 5 Implementação 25
Capítulo 6 Testes 26
6.1 Testes *end-to-end* 26
6.2 Testes de compatibilidade 30
6.3 Testes de usabilidade 31
Capítulo 7 Conclusão 34
7.1 Trabalho realizado 34
7.2 Trabalho futuro 35
Bibliografia 36

Lista de figuras

Figura 2.1 – Arquitetura do NativeScript..... 11

Figura 2.2 – Estratégias de armazenamento de dados de arquiteturas multi-tenant 16

Figura 2.3 – Polling e long polling 18

Lista de tabelas

Tabela 2.1 – Comparação entre abordagens de desenvolvimento móvel 7

Tabela 2.2 – Comparação entre metodologias de desenvolvimento 15

Tabela 6.1 – Testes end-to-end 27

Tabela 6.2 – Dispositivos móveis utilizados nos testes de compatibilidade..... 30

Tabela 6.3 – Tarefas definidas para testes de usabilidade 31

Tabela 6.4 – Graus de dificuldade das tarefas dos testes de usabilidade 32

Abreviaturas

API – *Application Programming Interface*

CSS – *Cascading Style Sheets*

EBS – *Emvenci Business Services*

FTP – *File Transfer Protocol*

HTML – *HyperText Markup Language*

HTTP – *Hypertext Transfer Protocol*

JSON – *JavaScript Object Notation*

JWT – *JSON Web Token*

MVC – *Model-View-Controller*

MVP – *Model-View-Presenter*

MVVM – *Model-View-View Model*

MWA – *Mobile Web App*

REST – *Representational State Transfer*

RGPD – *Regulamento Geral de Proteção de Dados*

RWD – *Responsive Web Design*

SDK – *Software Development Kit*

SMTP – *Simple Mail Transfer Protocol*

SOAP – *Simple Object Access Protocol*

UI – *User Interface*

URI – *Uniform Resource Identifier*

URL – *Uniform Resource Locator*

UX – *User Experience*

XML – *Extensible Markup Language*

Capítulo 1

Introdução

As empresas têm acelerado a sua transformação digital e alterado as prioridades das tecnologias de informação face ao crescimento tecnológico em geral, e atualmente, devido à pandemia de COVID-19, toda essa evolução se tornou mais significativa, agravando, por consequência, a exposição a ciberriscos e a complexidade de gestão de colaboradores, que passaram a trabalhar remotamente.

A cibersegurança é uma questão importante em todas as empresas, independentemente do seu tamanho. Os efeitos de ciberataques, de falhas técnicas ou de negligência humana podem afetar seriamente a atividade de uma organização e, até mesmo, comprometer a continuidade dos negócios. O investimento em soluções de cibersegurança possibilitam que as empresas evitem perdas financeiras, protejam a sua imagem e vão de encontro às exigências do RGPD (Regulamento Geral de Proteção de Dados).

A necessidade de melhorar a segurança de informação e possibilitar uma menor complexidade a nível processual e de gestão exige o recurso a tecnologias, processos e políticas consolidadas e automatizadas de prevenção contra ameaças que garantam uma proteção mais eficiente e inteligente às empresas e todos os seus intervenientes.

Alguns procedimentos e práticas que melhoram a segurança de informação em empresas são a formação dos colaboradores, apresentando as ameaças às quais eles estão expostos e os comportamentos que devem ser evitados de modo a não comprometer a segurança dos dados, a implementação de políticas nas quais sejam estabelecidos, por exemplo, requisitos de controlo de acesso, padrão de palavra-chave, política de uso da Internet, entre outros, e ferramentas que ajudem na deteção, tolerância e mitigação a intrusões e vulnerabilidades.

O trabalho de projeto proposto pela Emvenci Business Services, no âmbito da disciplina de Projeto de Engenharia Informática, é focado no desenho e implementação de uma solução de cibersegurança para dispositivos móveis, baseada na plataforma *web* existente. A possibilidade de o utilizador ter acesso a informações importantes a

qualquer momento, sem a necessidade de ter um computador por perto, potencia tomadas de decisão mais eficientes que impactam positivamente o seu sucesso ao nível da segurança de informação.

A plataforma possui um módulo que fornece conteúdos digitais para formação a nível de cibersegurança, sendo isto um ponto importante, já mencionado acima, como uma prática útil para a melhoria da segurança dos colaboradores e das empresas. Na solução móvel a ser desenvolvida, a visualização destes conteúdos será mais dinâmica, acessível e cómoda, potenciando uma maior visualização e, consequentemente, uma melhor formação, na medida em que é mais propicio que o utilizador veja os conteúdos através do seu dispositivo móvel, por ser mais prático e, em geral, o dispositivo mais presente e utilizado ao longo do dia.

1.1 Instituição de acolhimento

A instituição de acolhimento é a Emvenci Business Services (EBS), uma empresa que trabalha a nível de consultoria, operações de processos de negócios e serviços de auditoria, tendo como objetivo ajudar os seus clientes a mitigar riscos e aproveitar oportunidades.

A EBS desenvolve a plataforma Cybersecurity Cloud, como uma solução SaaS (*Software as a Service*), que suporta a integração de diferentes módulos na área de cibersegurança, podendo estes serem individualmente subscritos pelos clientes, para os ajudar a obter informações importantes e melhorar as respostas de segurança.

1.2 Motivação

O mundo digital encontra-se num crescimento exponencial que fomenta a popularização de dispositivos móveis na medida em que mudamos a forma como comunicamos, trabalhamos e relacionamos, tornando importante o imediatismo de informação.

A par do aumento da utilização dos dispositivos móveis, o mercado de aplicações para estes também aumenta e a prova disso é o facto de tanto a App Store (iOS) como a Google Play Store (Android), em 2020, terem superado os seus números e alcançando recordes em termos de receitas, aplicações e downloads.

Através do desenvolvimento de aplicações móveis, em comparação com plataformas *web*, é possível obter uma maior compatibilidade tecnológica não só pelo facto de haver uma maior capacidade de utilizar o *hardware* e *software* do dispositivo, mas também por ser possível tirar partido de tecnologias já integradas no mesmo, como o uso de diferentes sensores, acesso a contatos, sistema de posicionamento global

(GPS), câmera, calendário, entre outros. Ao ter um acesso direto ao sistema operativo do dispositivo, as aplicações fornecem uma maior velocidade e fluidez nas suas interações, melhorando a experiência do utilizador. Notificações, autenticações biométricas, entre outras capacidades nativas dos dispositivos móveis asseguram uma maior segurança, portabilidade de informação e utilidade aos seus utilizadores.

O desenvolvimento de uma solução móvel para a plataforma Cybersecurity Cloud garante que seja possível o utilizador rever rapidamente as suas tarefas, tomar decisões e poder aceder a toda a informação fornecida pela plataforma sem a necessidade de ter um computador por perto.

1.3 Objetivos

A finalidade deste projeto é desenvolver uma solução de cibersegurança para dispositivos móveis, de modo a tornar a plataforma mais acessível a qualquer momento ao utilizador.

Este projeto é definido através dos seguintes objetivos:

- Compreender a arquitetura da plataforma e dos seus vários módulos;
- Definir os requisitos funcionais da solução, alinhados com a plataforma de cibersegurança existente;
- Desenhar e desenvolver a solução com recurso à *framework* NativeScript (usando Angular);
- Definir e executar testes funcionais para a entrada em produção da solução.

1.4 Estrutura do documento

O documento encontra-se dividido em 7 capítulos.

O primeiro capítulo é referente à apresentação do documento, introduzindo o projeto desenvolvido assim como a instituição de acolhimento, os objetivos e motivações associadas ao mesmo.

No capítulo 2 são apresentadas diferentes tecnologias e abordagens de desenvolvimento e introduzidos conceitos.

A identificação das principais funcionalidades a serem consideradas no desenvolvimento da solução é feita no capítulo 3, ao apresentar a plataforma em que a solução se baseia, requisitos funcionais, não funcionais e *user stories*.

O capítulo 4 incide sobre as tecnologias utilizadas, a arquitetura e desenho da solução, descrevendo os componentes a desenvolver e respetivas funcionalidades.

Todo o processo relativo à implementação das funcionalidades, bem como a estrutura da aplicação é descrito no capítulo 5.

No capítulo 6 são apresentados diferentes testes à solução e os resultados das suas execuções.

O último capítulo conclui o documento, verificando se os objetivos definidos foram concretizados, identificando pontos de reflexão e o trabalho futuro.

Por motivos de confidencialidade inerente ao trabalho desenvolvido e descrito nesta tese, o capítulo 5 e as secções 3.2, 3.3, 3.4, 4.2, 4.3 e 7.2 não são apresentadas nesta versão. Contudo, existe uma versão completa e confidencial.

Capítulo 2

Conceitos e definições

Neste capítulo está presente uma descrição do trabalho relacionado com o projeto desenvolvido. São apresentadas diferentes abordagens e *frameworks* para o desenvolvimento de aplicações móveis, metodologias de desenvolvimento de *software* e definições e conceitos de tecnologias.

2.1 Abordagens de desenvolvimento móvel

A escolha da abordagem de desenvolvimento móvel é importante e tem um grande impacto no ciclo de vida da aplicação.

As questões a ter nesta escolha têm que ver com a integração com o *hardware* do dispositivo, a segurança, o desempenho, a confiabilidade e as limitações de armazenamento.

De acordo com [1] existem três tipos de abordagem para o desenvolvimento de aplicações móveis: nativo, *web* e híbrido.

2.1.1 Abordagem nativa

As aplicações nativas são desenvolvidas para uma plataforma específica, isto é, são desenvolvidas usando SDKs e ferramentas de desenvolvimento próprias para o sistema. Por exemplo, Java, SDKs do Google ou Kotlin para Android, Objective C ou Swift para iOS e .NET para Windows.

A principal vantagem das aplicações nativas é o acesso completo aos recursos e APIs disponíveis em cada plataforma, o que permite obter um alto grau de desempenho, fazendo com que o código nativo seja ideal para construir aplicações complexas. Em [1], ao contrário do caso de linguagens interpretadas como o JavaScript, é explicado que o código nativo é compilado e consequentemente mais rápido. É também afirmado que com código nativo a criação de interfaces para o utilizador é mais eficiente, pois os *pixels* são desenhados diretamente no ecrã através de abstrações e de APIs nativas. As

aplicações nativas garantem ao utilizador um fluxo mais natural ao providenciarem uma melhor experiência de utilizador, integrando-se facilmente no sistema operativo.

Uma das grandes desvantagens desta abordagem é os elevados custos caso a aplicação seja implementada em diferentes plataformas, visto que não é possível partilhar o código entre plataformas.

2.1.2 Abordagem *web*

Esta abordagem de desenvolvimento usa o mesmo fluxo de trabalho que é normalmente associado ao desenvolvimento *web* em computadores. São contruídos *websites* em HTML, JavaScript e CSS que serão acedidos através de *browsers* no telemóvel. Não sendo aplicações nativas a nenhum sistema em específico, não necessitam de ser descarregadas ou instaladas, mas em contrapartida requerem uma ligação contínua à internet para serem e manterem a sua execução.

Existem duas formas pelas quais os programadores direcionam as aplicações para os dispositivos móveis: Responsive Web Design (RWD) e Mobile Web App (MWA). Com RWD, os programadores focam-se em modificar o layout e o display de websites que já existem de modo a que se adaptem a ecrãs mais pequenos, como são os dos dispositivos móveis. Com MWA, os dispositivos móveis são detetados e direcionados para uma aplicação da *web* otimizada onde é criada uma interface consoante as convenções especificas do dispositivo móvel.

2.1.3 Abordagem híbrida

A abordagem híbrida é uma combinação de uma aplicação *web* e de uma aplicação nativa. Este tipo de aplicações garante acesso ao *hardware* dos dispositivos e às APIs internas, em diferentes sistemas. Como referido em [2], as aplicações de abordagem híbrida consistem em duas partes: o código de *backend*, através de linguagens como HTML, CSS ou JavaScript, e uma *shell* nativa, que carrega o código utilizando *webview*.

Uma das suas vantagens é poder reutilizar a interface do utilizador em diferentes plataformas, pois a lógica de negócio é independente da plataforma, permitindo um desenvolvimento mais barato e mais rápido em comparação com a abordagem nativa, utilizando apenas uma linguagem. Em termos de experiência do utilizador, dependendo do dispositivo, as aplicações híbridas podem ter um aspeto pouco responsivo. Esta abordagem não é adequada para aplicações muito complexas, pois tem grande dependência de *plugins* para interagir com os recursos internos do dispositivo [3].

2.1.4 Comparação entre abordagens

O desenvolvimento de aplicações utilizando a abordagem nativa possibilita uma melhor experiência com uma interface rápida e objetiva, uma vez que esta abordagem permite o acesso aos recursos do dispositivo. Apesar da melhor performance, as aplicações nativas implicam mais código e linguagens para diferentes plataformas, o que acaba por elevar o custo de produção.

A abordagem *web* diminui este custo na medida em que o código é reaproveitado para diferentes plataformas, visto que a sua execução acontece no *browser*, independentemente do sistema. A sua desvantagem é o limitado acesso ao *hardware* do dispositivo e às suas APIs, resultando numa performance mais pobre.

Algures entre estas duas abordagens, encontra-se a abordagem híbrida, que possui as vantagens de ambas, diminuindo as desvantagens mais críticas, como o custo de desenvolvimento e o acesso a recursos e APIs nativas.

Tabela 2.1 – Comparação entre abordagens de desenvolvimento móvel

	Nativa	***Web***	**Híbrida**
Acesso a APIs nativas	Sim	Não	Sim
Distribuição nas lojas de aplicações	Sim	Não	Sim
Execução em diferentes plataformas	Não	Sim	Sim
Linguagens	Java, Objective C, Kotlin, Swift	HTML, CSS e JavaScript	HTML, CSS e JavaScript, TypeScript, C#, Dart
Custo e duração de desenvolvimento	Alto	Baixo	Moderado
Desempenho	Alto	Moderado	Moderado
Bases de código	Múltiplas	1	1
Requer internet	Não	Sim	Não

2.2 Angular

O Angular é uma *framework*, desenvolvida pela Google, utilizada para o desenvolvimento de aplicações em HTML e TypeScript, onde existem módulos que são responsáveis pelo fornecimento de um contexto de compilação para os componentes e estes definem Views que são conjuntos de elementos do ecrã que podem ser modificados consoante a lógica e os dados fornecidos. Alguns fundamentos desta *framework* são os seguintes:

- Módulos: Uma aplicação tem pelo menos um módulo raiz, geralmente denominado como AppModule que permite a posterior construção da estrutura a implementar dentro deste. Os módulos podem importar funcionalidades e variáveis de outros módulos, permitindo reutilização a nível funcional e estrutural, melhorando a organização dos componentes e da lógica funcional, especialmente em aplicações mais complexas.
- Componentes: A par do que acontece com os módulos, todas as aplicações têm pelo menos um componente raiz. Cada componente é definido por uma classe que contém os dados e lógica a aplicar, estando associado a um *template* HTML que define a View a ser apresentada. Para além disto, os componentes usam serviços que fornecem funcionalidades específicas sem ligação direta às Views. Os serviços podem ser injetados nos componentes como dependências, tornando o código modular, reutilizável e eficiente. A classe que corresponde a um componente possui a anotação do decorador @Component(). Um decorador tem como função anexar tipos específicos de metadados às classes, informando o sistema de quais as suas funções e os seus comportamentos.
- Serviços: Os serviços são usados quando existem certos dados ou funcionalidades que devem ser partilhados entre componentes, ou seja, que não se encontram associados a nenhuma View específica. A classe responsável por definir um serviço é anotada com o decorador @Injectable().
- Roteamento: O Router fornece um serviço que permite a definição de um caminho de navegação entre os vários estados da aplicação. Para definir regras de navegação, é necessário associar *navigation paths* aos componentes a apresentar. Ao modificar o URL a lógica trata de modificar a estrutura e componentes a serem apresentados.
- Observáveis: Os Observáveis são componentes que possibilitam a passagem de dados entre *publishers* e *subscribers*. Estes componentes são declarativos, isto é, definem uma função para publicar valores, mas esta não é executada até ser subscrita. O *subscriber* depois passa a receber

notificações para poder consumir os dados até a função terminar ou até cancelar a subscrição.

- HTTPClient: Geralmente a comunicação entre o *frontend* e o *backend* é efetuada através do protocolo HTTP. O HTTPClient fornece uma API HTTP simplificada baseada na interface XMLHttpRequest utilizada pelos *browsers*. Para usar esta API é necessário importar o módulo HTTPClientModule. Com este módulo é possível utilizar os seguintes métodos HTTP: GET, PUT, PATCH, POST e DELETE.

2.3 *Frameworks* para desenvolvimento móvel

Nesta secção serão apresentadas algumas *frameworks* que têm como objetivo simplificar o desenvolvimento de aplicações móveis para múltiplas plataformas, sendo os sistemas Android e iOS o foco deste projeto. Estando o desenvolvimento híbrido em expansão, existe uma alta diversidade a este nível, sendo que as seguintes *frameworks* são as mais notáveis atualmente.

2.3.1 React Native

O React Native é uma *framework* baseada numa outra para desenvolvimento *web*, React, desenvolvida pelo Facebook, que possibilita o desenvolvimento de aplicações móveis, tanto para Android, como para iOS, utilizando apenas JavaScript. Diferente de outras *frameworks* com esta mesma finalidade, todo o código desenvolvido com o React Native é convertido para a linguagem nativa do sistema operacional, o que aumenta a performance da aplicação desenvolvida. Esta *framework* não está dependente de HTML, permitindo desenvolver aplicações nativas, indistinguíveis de aplicações construídas em linguagens nativas [4].

2.3.2 Ionic

O Ionic é uma *framework* de desenvolvimento de *software open source* desenvolvido para permitir a construção de aplicações móveis. A sua estrutura encontra-se construída através de Angular, React ou Vue.js e Apache Cordova ou Capacitor, fornecendo várias ferramentas e serviços que facilitam e diminuem os custos desenvolvimento ao usar tecnologias *web*, como HTML, CSS e JavaScript, permitindo também que sejam desenvolvidas interfaces com aparência e comportamento nativos muito semelhantes às aplicações Android e iOS.

2.3.3 Xamarin

O Xamarin é uma *framework* usada para desenvolver aplicações móveis para Android, iOS e Windows utilizando apenas C# e uma única *codebase*. No Xamarin, cada implementação é feita de forma independente através de um projeto específico a cada plataforma, permitindo preservar as interfaces de utilizador nativas. Caso seja necessário aceder a bibliotecas nativas, esta *framework* possibilita o acesso às mesmas através da criação de *bindings* nativos.

2.3.4 Flutter

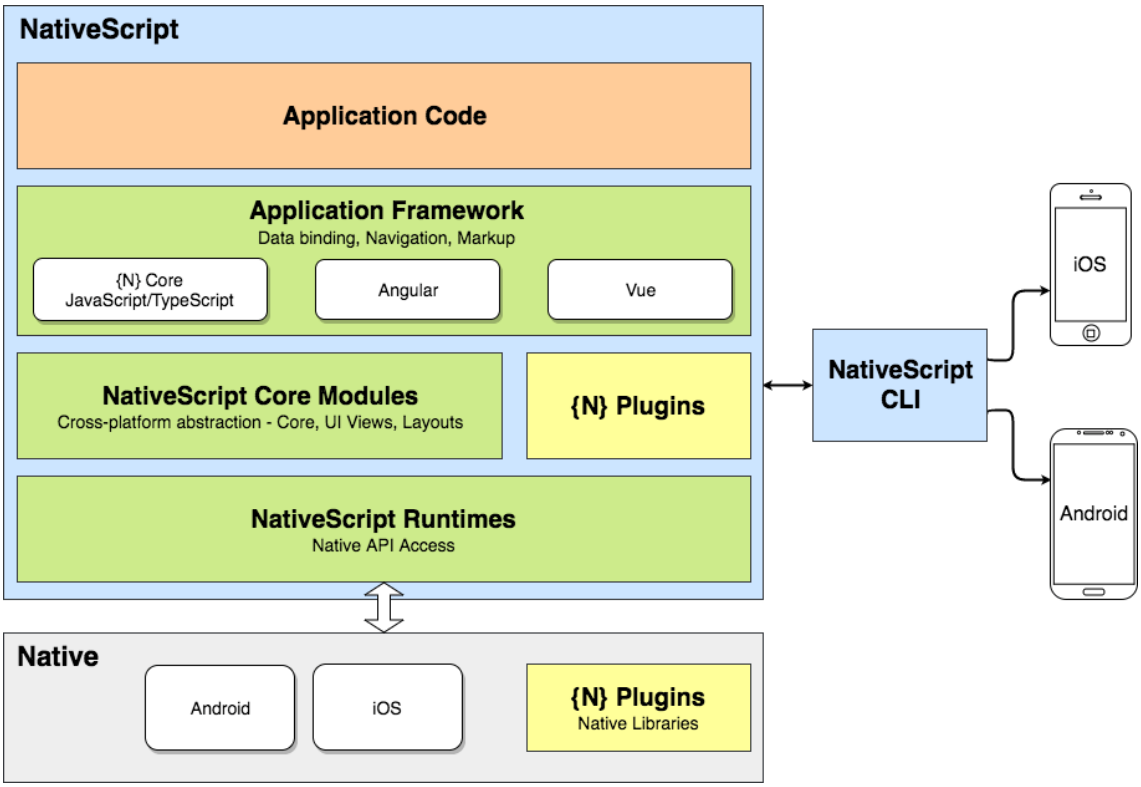
O Flutter é uma *framework* criada pela Google para desenvolver aplicações em várias plataformas com apenas um código-fonte. As aplicações desenvolvidas com esta *framework* são programadas através de Dart, uma linguagem de programação orientada a objetos que suporta tanto compilação *ahead-of-time* como *just-in-time*. Dart pode ser compilado para JavaScript, o que faz com que o código possa ser partilhado entre diferentes plataformas *web* e móveis.

2.3.5 NativeScript

O NativeScript é uma *framework* que apresenta conceitos de construção, apresentados na Figura 2.1, como: Runtimes, Core Modules, CLI e Plugins. A combinação destes possibilita o acesso direto a todas as APIs da plataforma nativa, seja iOS ou Android, através de JavaScript, TypeScript, Angular, React ou Vue.js. Além disso, o NativeScript contém um conjunto de abstrações multiplataforma com o objetivo de facilitar a rápida construção de aplicações de qualidade a partir da mesma base de código. Sendo a plataforma web desenvolvida em Angular, parte do código ou lógica funcional pode ser reaproveitado para o desenvolvimento da solução móvel. A curva de aprendizagem é também reduzida, visto que a linguagem entre as plataformas é a mesma (TypeScript).

O NativeScript e os seus *plugins* são instalados da mesma forma que o Angular, através de um gestor de pacotes, por norma, o NPM (*Node Package Manager*). Enquanto o Angular usa HTML para criar as Views, o NativeScript usa XML, sendo este tipo de arquivo usado nas linguagens nativas dos dispositivos móveis e por isso ser possível chamar elementos de UI nativos de cada plataforma. É ainda possível dentro do mesmo projeto, separar código para iOS e Android quando é necessário usar diferentes lógicas e UI para cada um.

Figura 2.1 – Arquitetura do NativeScript [5]



2.4 Padrões de arquitetura

Um padrão arquitetural é uma solução previamente estudada, testada e documentada de um problema que ocorre com frequência. O modelo ajuda na tomada de decisões do projeto de *software*. É o padrão arquitetural que define a estrutura de desenvolvimento do *software*. Os três padrões de arquitetura mais utilizados são: Model-View-Controller (MVC), Model-View-Presenter (MVP) e Model-View-View Model (MVVM).

Estes padrões têm como objetivo isolar ao máximo a camada de apresentação de um sistema. Em termos de nomenclatura, todas coincidem em suas duas letras iniciais: M e V, que se referem às camadas Model e View, respetivamente. Posto isto, assume-se que o que varia entre estes padrões é o conceito em torno da restante parte da sigla.

2.4.1 Model-View-Controller (MVC)

O padrão MVC foi, segundo [6], dos primeiros a ser concebido e é composto por três componentes independentes: Model, View e Controller. O Model gere a lógica de negócio e manipula os dados. A View tem como principal objetivo apresentar os dados ao utilizador, que são manipulados pelo Controller e posteriormente pelo Model. O Controller interpreta as ações que o utilizador fornece, informando a camada Model e View para alterar os dados/estado conforme o solicitado.

2.4.2 Model-View-Presenter (MVP)

Segundo [6], o padrão MVP deriva do padrão MVC e foi criado para automação de testes com o objetivo de aumentar a quantidade de código a ser testado. A arquitetura do MVP é composta por três componentes: Model, View e Presenter. O Model tem o mesmo conceito que no MVC, sendo responsável por responder a pedidos de informações sobre o seu estado e instruções para o alterar. A View é a componente responsável pela apresentação de informações. A diferença deste padrão em comparação com o MVC é a camada Presenter. Esta visa efetuar a ligação entre a View e o Model, respondendo a eventos e alterando o Model. Neste padrão as ações são encaminhadas pela View para o Presenter.

2.4.3 Model-View-View Model (MVVM)

Segundo [6], o MVVM é uma variante do padrão MVC e tem como objetivo separar a interface da lógica da aplicação, onde a View é responsabilidade de um *designer* em vez de um programador. A arquitetura MVVM é composta por três componentes: Model, View e View Model. O Model tem o mesmo conceito que no MVC. A View é responsável por definir a estrutura, *layout* e aparência do que o utilizador vê no ecrã. O View Model separa a View do Model, expondo métodos e comandos para manipular os dados. Este componente efetua a mediação entre os outros dois componentes, lidando com a lógica da View e interagindo com o Model ao apresentar os seus dados na View.

2.5 REST e SOAP

No contexto da computação móvel, o uso de serviços *web* tem permitido ultrapassar algumas limitações impostas pelos dispositivos móveis. Estes oferecem como benefícios a extensibilidade e a interoperabilidade entre diversas aplicações podendo ser executados nas mais variadas plataformas e *frameworks*. Os serviços *web* mais utilizados atualmente são o Representational State Transfer (REST) e o Simple Object Access Protocol (SOAP).

2.5.1 REST

O REST é uma arquitetura de *software* que define uma série de regras a serem adotadas na criação de serviços *web*. De acordo com [7], o REST segue uma arquitetura cliente-servidor na qual o cliente envia o pedido para o servidor, o servidor processa esse pedido e envia uma resposta. Estes pedidos e respostas do servidor são feitos numa abordagem baseada em recursos, representados por URI's (Uniform Resource Identifiers). Estes recursos podem ser localizados no servidor e posteriormente

modificados, apagados ou ser usados para extrair informação. Em REST, é utilizada uma interface uniforme para aceder aos recursos, sendo usado um conjunto fixo de métodos HTTP (GET, PUT, POST e DELETE). Nesta abordagem, cada transação é feita de forma independente e não é necessário manter dados de sessão do cliente no servidor. Tendo em conta a complexidade da integração relacionada com as questões de processamento e de consumo de dados que se aplicam aos dispositivos móveis, como observado em [8] e [9], o REST demonstra ser a abordagem mais indicada para desenvolver aplicações móveis.

2.5.2 SOAP

O SOAP é um protocolo de comunicação que foi desenhado de modo a que as aplicações desenvolvidas com diferentes linguagens e em diferentes plataformas troquem informação entre si, de forma estruturada. Este protocolo utiliza XML para formatar e estruturar mensagens, possibilitando a sua troca entre diferentes plataformas. Como referido em [7], as mensagens XML podem ser enviadas através de diferentes protocolos de transporte como HTTP, FTP ou SMTP. A sua extensibilidade faz com que tenham um tamanho considerável. As mensagens SOAP são envoltas por um envelope SOAP que contém um cabeçalho e o corpo da mensagem [9].

2.6 Metodologias de desenvolvimento de software

O ciclo de vida do desenvolvimento de *software* é definido, de acordo com [10], como o processo de manter ou construir sistemas de *software*. Em [11] são salientados os três objetivos principais do mesmo: garantir a criação de sistemas de qualidade, fornecer controlo administrativo sobre os projetos e maximizar a produtividade dos funcionários

Uma das primeiras decisões que se tem ao desenvolver um projeto é escolher qual a metodologia de desenvolvimento que se deve usar. Existem duas metodologias utilizadas no ciclo de vida do desenvolvimento de software: Tradicionais e *Agile.*

2.6.1 Metodologias tradicionais

De acordo com [10], a metodologia tradicional segue um modelo sequencial, ou seja, uma etapa deve ser executada após a outra, assim, uma tarefa não pode ser iniciada enquanto a anterior não for concluída. As metodologias tradicionais de desenvolvimento de *software* necessitam da definição e documentação de um conjunto estável de requisitos do projeto. Alguns exemplos destas metodologias são os modelos Waterfall, V-Model e Rational Unified Process.

Há quatro fases que caracterizam esta metodologia:

- Determinar os requisitos para o projeto e a duração da implementação das várias fases de desenvolvimento e ao mesmo tempo, tentar prever os problemas que podem vir a aparecer durante o projeto;
- Planear a arquitetura e fazer o desenho do sistema, produzindo uma infraestrutura técnica na forma de diagramas ou modelos;
- Desenvolver o código até se chegar aos objetivos pretendidos. Na fase de desenvolvimento há uma divisão de tarefas por várias equipas diferentes. Começa-se a fazer testes para endereçar problemas que possam surgir o mais cedo possível;
- Perto do fim do projeto, o cliente participa nos testes de modo a dar o seu *feedback*.

A grande desvantagem deste tipo de metodologias prende-se com o sucesso do projeto estar dependente de um conjunto de processos predeterminados, o que faz com que implementar mudanças durante o ciclo de desenvolvimento possa ser problemático em alguns casos. O *feedback* somente no final também pode ser um grande problema.

2.6.2 Metodologias *Agile*

A metodologia *Agile*, conforme [10], é baseada na ideia de desenvolvimento incremental e iterativo e que vai melhorado o *software* desenvolvido através do *feedback* do cliente. Assim, as fases dentro do ciclo de vida de desenvolvimento são revistas repetidamente. De acordo com o Manifesto *Agile*, os quatro principais fatores que caracterizam esta metodologia são os seguintes:

- Envolvimento do cliente o mais cedo possível;
- Desenvolvimento iterativo;
- Equipas auto-organizadas e que trabalhem em conjunto para atingir um objetivo comum;
- Capacidade de adaptação à mudança.

Em [12], atualmente, existem seis metodologias de desenvolvimento *Agile*: Crystal Methodologies, Dynamic Software Development, Feature-driven Development, Lean Software Development, Scrum e Extreme Programing. Ainda em [12], a principal prioridade destas metodologias é o retorno do investimento.

2.6.3 Comparação entre metodologias

A grande diferença entre as metodologias tradicional e as metodologias *Agile* é a capacidade que a última tem de entregar resultados com baixo custo, com rapidez e com requisitos que ainda poderão ser alterados.

As metodologias *Agile* dão enfase ao trabalho em equipa, e à rápida resposta à mudança, enquanto que as tradicionais priorizam contratos, planos e documentos de trabalho.

Tabela 2.2 – Comparação entre metodologias de desenvolvimento

	Tradicional	***Agile***
Requisitos do utilizador	Requisitos bem definidos e detalhados antes da implementação	Aquisição iterativa
Custo de voltar a desenvolver	Alto	Baixo
Direção do desenvolvimento	Fixa	Facilmente alterável
Testes	Depois do código estar completamente escrito	Em cada iteração
Envolvimento do cliente	Baixo	Alto
Qualidades extra dos programadores	-	Capacidades interpessoais e conhecimento básico da área de negócios
Escala de projeto recomendada	Grande escala	Baixa a média escala

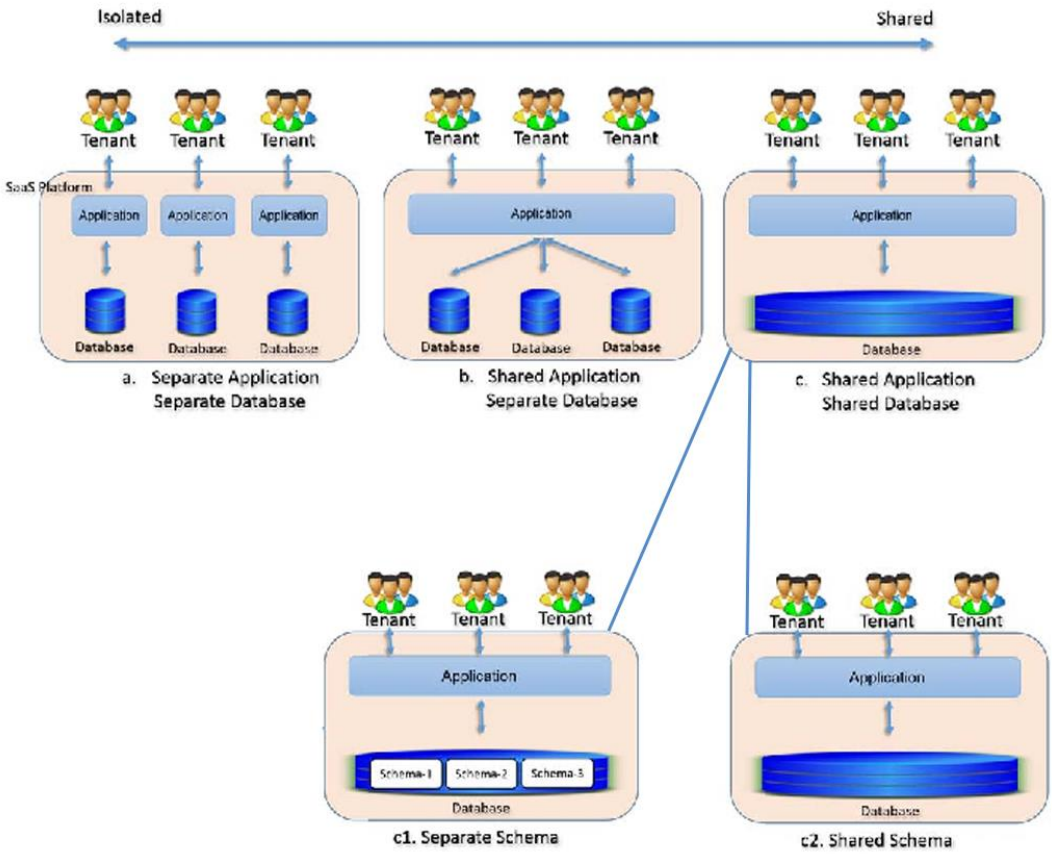
2.7 Arquitetura *multi-tenancy*

A EBS desenvolve um SaaS seguindo uma arquitetura *multi-tenancy*, onde existe apenas uma única instância do *software* em execução na infraestrutura do *service provider* à qual os múltiplos *tenants* (clientes) acedem. A arquitetura *multi-tenant* requer a customização de instância tendo em conta os requisitos dos diferentes *tenants* [13] e [14], comparado com a arquitetura *multi-user*.

Conforme [14], existem três estratégias de armazenamento de dados numa arquitetura *multi-tenant*, representadas pela Figura 2.2. Na primeira, (a) Separarate Application Separate Database, cada utilizador tem o seu próprio software e base de dados completamente isolados dos restantes *tenants*. Na segunda estratégia, (b) Shared Application Separated Database, os vários *tenants* partilham o mesmo software, mas é

atribuída a cada *tenant* a sua própria base de dados, não havendo partilha de tabelas de dados dos diversos *tenants*. A terceira estratégia, (c) Shared Application Shared Database, implica que os *tenants* partilhem o mesmo *software* e a mesma base de dados. Esta está subdividida em duas estratégias, Separated Schema (c1), onde cada *tenant* tem as suas tabelas de dados separadas das tabelas de dados dos restantes *tenants*, e a estratégia Shared Schema (c2), onde as tabelas de dados contêm informação de todos os *tenants*. A EBS desenvolveu o seu *software* de acordo com a estratégia Shared Schema (c2).

Figura 2.2 – Estratégias de armazenamento de dados de arquiteturas multi-tenant [13]



2.8 *Polling* e *long polling*

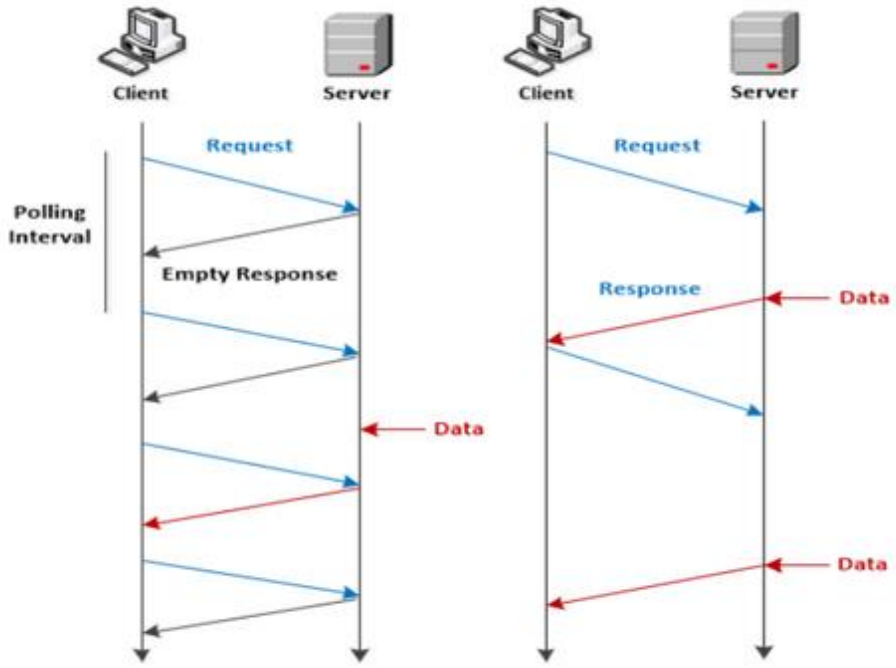
Como referido em [15], quando um cliente abre um *browser* para visitar uma página *web*, é enviado um pedido HTTP ao servidor *web* que a serve, sendo que este recebe o pedido e devolve a resposta. Esta resposta pode ser antiga quando o navegador finalizar a construção dos elementos. Por esta razão, ter informações em tempo real pode sempre atualizar o conteúdo de forma manual, mas esta abordagem não é eficiente.

Posto isto, para fornecer aplicações *web* em tempo real, foram conseguidas várias tentativas para entregar mensagens ao cliente utilizando *polling* e *long polling*.

O *polling* consiste em criar uma rotina que envia um pedido de tempos em tempos ao servidor para perguntar se existem mudanças no modelo de dados da aplicação. Caso isto se verifique, a resposta ao pedido contem esses dados e os mesmos podem ser atualizados na aplicação. Esta técnica consome uma grande quantidade de recursos quando a aplicação tende a ser escalável. Efetuar repetidos pedidos a um servidor desperdiça recursos, uma vez que cada nova ligação que chega deve ser estabelecida, os cabeçalhos HTTP devem ser analisados, uma consulta para novos dados deve ser realizada, e uma resposta (geralmente sem novos dados para oferecer) deve ser gerada e entregue. A ligação deve então ser fechada e quaisquer recursos limpos. Em vez de ter de repetir este processo várias vezes para cada cliente até que novos dados para um determinado cliente se tornem disponíveis, foi criado o *long polling* que é uma técnica em que o servidor opta por manter a ligação de um cliente aberta o máximo de tempo possível, fornecendo uma resposta apenas após a disponibilização de dados ou um limiar de tempo limite.

A implementação do *long polling* é principalmente uma preocupação do lado do servidor. Do lado do cliente, é apenas necessário gerir um pedido para o servidor. Quando a resposta é recebida, o cliente pode iniciar um novo pedido, repetindo este processo sempre que necessário. A única diferença desta técnica para o original *polling*, no que diz respeito ao cliente, é que quando este usa *polling* pode deliberadamente deixar um pequeno período de tempo entre cada pedido de modo a reduzir a sua carga no servidor, e pode responder a intervalos com pressupostos diferentes do que faria para um servidor que não suporta *long polling*. Com o *long polling*, o cliente pode ser configurado para permitir um período de tempo mais longo (através de um cabeçalho *Keep-Alive*) ao ouvir uma resposta.

Figura 2.3 – Polling e long polling



Capítulo 3

Análise de requisitos

Este capítulo tem como objetivo descrever os requisitos associados à aplicação desenvolvida no âmbito deste projeto. A análise de requisitos é um processo essencial pois permite descrever características e funcionalidades que o sistema deve ter.

A EBS desenvolve os seus produtos seguindo uma metodologia *Agile*, nomeadamente Scrum. Nesta metodologia, o *scrum master*, que é o gestor do produto, tem a autoridade de decidir o que fará ou não parte do produto final. Após a definição de requisitos, este desenvolve o *backlog*, isto é, uma lista de tarefas por prioridade, que leva ao cumprimento dos requisitos definidos. A execução de tarefas encontra-se dividida em períodos de tempo pré-determinados, denominados de *Sprints*, havendo reuniões diárias entre a equipa que visam atualizar o estado de desenvolvimento do produto.

De seguida é introduzida a plataforma em que a solução se baseia e são enunciados os requisitos funcionais, não funcionais e *user stories* relevantes para o desenvolvimento da solução.

3.1 Introdução à plataforma

A solução móvel é baseada na plataforma web desenvolvida pela EBS, o portal Cybersecurity Cloud.

Projetada para fornecer ao utilizador uma experiência integrada, esta plataforma foca-se em pontos como a gestão de incidentes e vulnerabilidades, formação e treino nas áreas de segurança de informação através de conteúdos educacionais produzidos também pela empresa, agregação e transformação de análises de dados para rápida consulta, interpretação e decisão, partilha de informação crucial e compatibilidade com o RGPD.

A plataforma Cybersecurity Cloud disponibiliza aos clientes vários módulos, podendo existir um conjunto de subscrições ativas:

- "Policy Manager", permite efetuar toda a gestão do ciclo de vida das políticas e procedimentos das organizações, e assim melhorar a sua visibilidade interna e otimização dos controlos ao padronizar processos;
- "Privacy Manager", permite responder a vários requisitos de diferentes normas de proteção de dados pessoais, em particular ao RGPD (Regulamento Geral de Proteção de Dados), sendo as funcionalidades mais relevantes:
    - Inventário e gestão das atividades de processamento de dados
    - Inventário, análise e resposta a incidentes de violação de dados pessoais
    - Registo e resposta a pedidos de titulares de dados pessoais
    - Inventário e gestão dos consentimentos, individuais, para tratamento de dados pessoais
- "Phish Simulator", possibilita a criação de simulacros de *phishing*, por *email* e *sms*, com várias capacidades de personalização para testar os colaboradores das empresas (clientes) em cenários de *phishing*, seguros, mas similares a ataques reais;
- "eLearning", fornece conteúdos digitais, em particular vídeos e ilustrações gráficas, tipicamente utilizando formato MP4 e PDF, para formação a nível de segurança de informação e privacidade;
- "Threat & Vulnerability Manager", mantém e investiga todas as vulnerabilidades de forma centralizada
- "Log Analytics", transforma dados de modo a torná-los mais compreensíveis para facilitar decisões.

Umas das tecnologias utilizadas para a construção desta plataforma e que se relaciona diretamente com o desenvolvimento da solução móvel é o Angular. Esta *framework* é utilizada para contruir a interface (*frontend*) de aplicações *web* do tipo SPA (Single Page Applications) com recurso a HTML, CSS e TypeScript. Uma SPA é uma aplicação *web* construída numa só página, na qual a interação e a navegação entre as sessões de uma página ocorrem de maneira a qual não é necessário recarregar o website em cada uma dessas mudanças.

A principal característica desta *framework* que se encontra relacionada a este trabalho é o seu suporte *cross-platform*, que neste caso tem como objetivo manter a plataforma *web* e desenvolver a solução móvel continuando a utilizar o Angular e integrando o NativeScript.

3.2 Requisitos funcionais

A presente secção está omissa devido à confidencialidade inerente ao projeto.

3.3 Requisitos não-funcionais

A presente secção está omissa devido à confidencialidade inerente ao projeto.

3.4 *User stories*

A presente secção está omissa devido à confidencialidade inerente ao projeto.

Capítulo 4

Desenho da solução

Este capítulo descreve a arquitetura e o desenho da solução, explicando de que forma permitem cumprir os requisitos definidos no Capítulo 3. Serão apresentadas tecnologias utilizadas na implementação da solução, a arquitetura que esta adota e os componentes que fazem parte da mesma.

4.1 Tecnologias utilizadas

Nesta secção são apresentadas as principais tecnologias utilizadas na implementação da solução. O uso da *framework* NativeScript permite o desenvolvimento móvel através da *framework* de *frontend*, Angular. A solução consome a API da Cybersecurity Cloud de modo a aceder e armazenar dados de forma persistente. A integração de *live chat* passa pela utilização de um SDK para este fim.

4.1.1 Angular

O *frontend* da plataforma Cybersecurity Cloud é desenvolvido com recurso a esta *framework* e esta foi também utilizada para o desenvolvimento da solução móvel em conjunto com o NativeScript.

A grande parte da lógica da solução é efetuada através de serviços ou componentes criados com o Angular. Os módulos definem as dependências de cada componente, sejam estas serviços ou outros componentes. O roteamento de páginas é também realizado com esta *framework.* Os pedidos ao *backend* são executados em serviços que utilizam o HTTPClient e retornam observáveis que são subscritos para retornar dados, enviá-los para a camada lógica e posteriormente apresentá-los na interface.

4.1.2 NativeScript

A solução móvel foi desenvolvida seguindo uma abordagem híbrida. A *framework* utilizada foi o NativeScript, com a vertente Angular, que permite desenvolver

aplicações multiplataforma especialmente focadas em dispositivos móveis, no caso deste trabalho, iOS e Android.

O NativeScript, para além de ter a capacidade, acima mencionada, de gerar aplicações para os diferentes sistemas, trata de interagir com elementos e *use cases* mais nativos que não conseguem ser resolvidos através do Angular. Os *templates* das Views não utilizam estrutura, nem elementos HTML, mas sim elementos do NativeScript que se ligam diretamente a elementos nativos dos sistemas. A solução acede a recursos nativos, atualmente não suportados pelo desenvolvimento de alto nível da *framework* (NativeScript).

4.1.3 Cybersecurity Cloud API

A solução móvel consome uma API própria do produto da EBS e partilha o mesmo nome da plataforma, Cybersecurity Cloud. Esta API é desenvolvida em Golang (GO) e segue uma arquitetura REST. Esta arquitetura é mais flexível, permitindo que as implementações sejam muito customizáveis face às necessidades. O REST permite retornar diferentes formatos de mensagens, como HTML, JSON, XML e texto simples. Sendo também mais leve e atingindo um maior desempenho, esta arquitetura é adequada à solução na medida em que é necessário gastar o mínimo de recursos do dispositivo e comunicar com a API de forma eficiente.

Todos os *endpoints* desta API, exceto os que, no *workflow* da mesma, se encontram antes da autenticação do utilizador, encontram-se protegidos por uma verificação de *token* (JWT). Para além de confirmar se o *token* é válido, o *backend* verifica as permissões do utilizador associado ao mesmo e consoante estas permite ou não uma resposta de sucesso, podendo até alterar o modelo de dados retornados para diferentes permissões.

4.1.4 Intercom

A plataforma Cybersecurity Cloud integra um *live chat* com recurso ao *software* Intercom. A solução móvel, como definido na análise de requisitos, deve permitir ao utilizador aceder também ao *live chat* que é utilizado pela plataforma, sendo este partilhado entre ambas ao ser associado ao utilizador.

O Intercom é um sistema de *live chat* para equipas de apoio, vendas e marketing. Este sistema tem 3 módulos principais: o módulo de *live chat* para comunicar entre utilizadores, o módulo educativo para definir e aceder às perguntas mais frequentes através de artigos e o módulo para o envio de *e-mails* automatizados, não utilizado pela plataforma, para converter utilizadores em compradores.

No caso desta solução móvel, é utilizado o SDK do Intercom direcionado a ambos os sistemas operativos, Android e iOS. Este SDK permite usar o Intercom Messenger na solução, conversar entre a equipa de suporte e utilizadores, enviar notificações de mensagens, mostrar a central de ajuda (artigos) e seguir eventos.

4.2 Arquitetura

A presente secção está omissa devido à confidencialidade inerente ao projeto.

4.3 Componentes

A presente secção está omissa devido à confidencialidade inerente ao projeto.

Capítulo 5

Implementação

O presente capítulo está omisso devido à confidencialidade inerente ao projeto.

Capítulo 6

Testes

Um processo fundamental nas diversas fases de desenvolvimento de *software* é a execução de testes. Esta fase destina-se à avaliação da funcionalidade da solução, com a intenção de descobrir se esta cumpre os requisitos especificados, e à identificação de pontos de melhoria para garantir uma maior qualidade no produto final. Como consta em [16], a fase de testes pode ser dividida em 4 subfases: a perceção do ambiente da solução, a seleção casos de teste, a execução e avaliação destes e a medição do seu progresso.

A EBS tem uma equipa de *testers* que colaborou na elaboração e execução de testes. Este capítulo apresenta o processo de testes *end-to-end*, de compatibilidade e de usabilidade que foram desenvolvidos nas diversas fases de implementação da solução, de modo a garantir que, a cada lançamento, se garantia que a solução cumpria com os requisitos e que as funcionalidades previamente definidas ou *on the fly* continuavam a cumprir com os objetivos definidos.

6.1 Testes *end-to-end*

Os testes de *end-to-end* são uma técnica que testa toda a solução ao longo do seu *workflow* para garantir que esta se comporta como esperado. Este tipo de testes identifica as dependências do sistema, garantindo que todos os componentes integrados na solução trabalham em conjunto.

Como referido anteriormente, o planeamento dos testes foi desenvolvido em colaboração com a equipa de *testers* e posteriormente executado a cada nova versão da solução. Sendo que existem componentes que interagem entre si, este processo de testes é realizado iterativamente de modo a garantir a operacionalidade de componentes ligados aos desenvolvidos na versão em que são executados os testes.

Na Tabela 6.1 são descritos os testes realizados aos diversos componentes e funcionalidades da solução. A tabela encontra-se dividida em quatro colunas: o

identificador do teste, o módulo da aplicação referente este, a descrição do teste e o objetivo do mesmo.

Tabela 6.1 – Testes end-to-end

ID	Módulo	Descrição	Objetivo
E2E-01	Autenticação	Escolher o ambiente de *tenant*	No componente Tenant, após tocar 10 vezes no logotipo da solução, deve aparecer um novo campo de seleção com os seguintes ambientes: *Testing*, *Staging* e *Production*.
E2E-02	Autenticação	Autenticar utilizador	Após escolhido o ambiente de *tenant*, no componente Autenticação, devem ser inseridas e submetidas credenciais de acesso. Se erradas, a solução deve apresentar o erro, caso contrário, segue para o portal destinado ao utilizador, consoante as suas permissões.
E2E-03	Autenticação	Sair da conta	Estando o utilizador dentro do portal, no menu apresentado no componente Mais, quando toca em "Sign out", a solução deve sair da conta do utilizador e apresentar o componente Login.
E2E-04	Portal	Aceder a notificações	Estando o utilizador dentro do portal, este deve conseguir aceder às notificações que se encontram no componente Notificações. O conteúdo das notificações deve ser igual ao que é apresentado na plataforma *web*.
E2E-05	Portal	Aceder ao *live chat*	Estando o utilizador dentro do portal e tendo permissões administrativas, este deve conseguir abrir o *live chat* através do botão "Live Chat" do menu do componente Mais.
E2E-06	Portal	Alterar idioma do utilizador	Estando o utilizador dentro do portal, este deve conseguir alterar o seu idioma no componente Conta, no campo de seleção

			"Preferred Language". Esta alteração deve refletir-se na plataforma web.
E2E-07	Portal	Habilitar ou desabilitar comunicações de novidades	Estando o utilizador dentro do portal, este deve conseguir habilitar ou desabilitar comunicações de novidades da plataforma no componente Conta. Esta alteração deve refletir-se na plataforma *web*.
E2E-08	Portal	Alterar tema	Estando o utilizador dentro do portal, este deve conseguir alterar entre o tema escuro e claro no componente Definições da aplicação.
E2E-09	Portal	Filtrar conteúdos	Estando o utilizador dentro do portal, este deve conseguir filtrar os conteúdos visualizados dentro dos componentes dos módulos.
E2E-10	eLearning	Aceder ao módulo eLearning	Estando o utilizador dentro do portal e tendo a subscrição do módulo de eLearning, dependendo das suas permissões, este deve conseguir visualizar os conteúdos e campanhas ao navegar para o módulo através do botão "Modules" do menu. No portal de utilizador os conteúdos devem apresentar-se por categorias. No portal de administrador deve ser possível aceder aos detalhes das campanhas e eventos de utilizadores.
E2E-11	eLearning	Visualizar conteúdo	No componente eLearning, ao abrir um conteúdo, este deve apresentar os seus detalhes e opções de escolha de idiomas. Após a submissão, a solução deverá apresentar o reprodutor de vídeo ou leitor de texto, conforme o tipo de conteúdo.
E2E-12	Policy Manager	Aceder ao módulo Policy Manager	Estando o utilizador dentro do portal e tendo a subscrição do módulo de Policy Manager, dependendo das suas permissões, este deve conseguir visualizar

			as políticas ao navegar para o módulo através do botão “Modules” do menu. No portal de utilizador os conteúdos devem apresentar-se em duas categorias distintas: “Pending Documents” para documentos ainda não aceites e “Last Attested Documents” para documentos aceites. No portal de administrador deve ser possível aceder aos detalhes das políticas e eventos de utilizadores.
E2E-13	Policy Manager	Visualizar política	No componente Policy Manager, ao abrir uma política, este deve exibir o conteúdo da mesma através do leitor de texto e possibilitar a alteração de idioma.
E2E-14	eLearning, Policy Manager	Responder a questionários	Depois de finalizar a visualização de um conteúdo no portal de utilizador, caso este tenha um questionário associado si, a solução deve apresentá-lo. A solução deve ter em conta as configurações definidas na criação do questionário e conteúdo.
E2E-15	Phishing Simulator	Aceder ao módulo de Phishing Simulator	Estando o utilizador dentro do portal de administrador e tendo a subscrição do módulo de Phishing Simulator, este deve conseguir visualizar as campanhas ao navegar para o módulo através do botão “Modules” do menu. Neste portal deve ainda ser possível aceder aos detalhes das campanhas e eventos de utilizadores.

Alguns dos problemas detetados ao executar estes testes prendiam-se com o facto de inicialmente haver falhas no código que levavam a problemas de armazenamento local ou remoto de algumas informações, emergindo diferenças de resultados entre esta solução e a plataforma *web*. Outro problema relacionava-se com a verificação de subscrições e permissões e o que isso impactava na apresentação dos módulos.

A iteratividade da execução destes testes, ao longo da implementação de novos componentes, fez com que o número de problemas detetados entre versões fosse

diminuindo significativamente. Este aspeto fez com que fosse mais simples e rápido que novas versões seguissem para a abordagem de preparação de lançamento da solução nas lojas de aplicações móveis.

6.2 Testes de compatibilidade

Um dos vários tipos de testes que devem ser executados de forma a garantir a compatibilidade em diferentes dispositivos, são os chamados testes de compatibilidade. Estes testes têm também como propósito perceber se a solução responde como expectável, em termos de interface e performance. A solução foi desenvolvida e pensada para ser suportada em dispositivos de diferentes sistemas operativos, Android e iOS. Tendo em conta os requisitos da solução e dependências da mesma, foi decidido suportar dispositivos Android com versão igual ou superior 8.0 e iOS com versão igual ou superior a 9.0.

Tabela 6.2 – Dispositivos móveis utilizados nos testes de compatibilidade

Dispositivo	**Sistema**	**Versão**
iPhone 8	iOS	14.7.1
iPhone 8 Plus	iOS	14.7.1
iPhone 11 Pro Max	iOS	14.7.1
Samsung Galaxy J5	Android	8.0
Samsung Galaxy S8 Plus	Android	9.0
Samsung S10	Android	11.0
Blackberry Key2	Android	8.1

Após executados os testes de compatibilidade nos dispositivos apresentados na Tabela 6.2, foram encontradas algumas diferenças de performance e de interface de utilizador.

Relativamente à performance, ao abrir e fechar componentes e na navegação entre estes, dispositivos com menos poder de processamento apresentavam um maior tempo de resposta a esta ações, em comparação com dispositivos mais recentes. Esta diferença era também notável entre dispositivos com sistema Android e outros com sistema iOS. Após realizadas otimizações que se encontram descritas no capítulo de implementação, a execução destas ações tornou-se mais rápida e fluída em todos os dispositivos, melhorando assim a experiência de utilizador.

Quanto aos problemas de interface de utilizador, estes prendem-se com dois fatores:

- As versões mais recentes de ambos os sistemas permitem definir, nas suas definições internas, um tema claro ou escuro que impactava a apresentação da solução, mesmo escolhendo um tema na aplicação;
- A interface da solução apresentava problemas de ajuste de ecrã em dispositivos que possuem *notch*, isto é, dispositivos que têm uma área na parte frontal que fica por cima do ecrã, normalmente para acomodar a câmera frontal ou sensores.

Para resolver o problema do tema, nos manifestos foram inseridas regras para que o sistema não impacte o tema escolhido na solução. No problema da apresentação da solução em dispositivos com *notch*, foram feitas alterações aos componentes para que estes se ajustem a todos os ecrãs.

6.3 Testes de usabilidade

Os testes de usabilidade são uma forma de garantir que as soluções se encontram adaptadas aos utilizadores e às suas tarefas, tentando mitigar resultados negativos da sua utilização. O objetivo destes testes é avaliar as seguintes características: até que ponto a solução é eficaz, isto é, até que ponto as funcionalidades e desempenho da solução satisfazem as tarefas para as quais foi desenvolvida e se esta é eficiente, ou seja, quantos recursos, como o tempo ou o esforço, são necessários para a utilizar. Estes testes fornecem ainda a capacidade de obter informações e sugestões de utilizadores, de modo a melhorar o produto final.

As tarefas definidas para serem executadas nos testes de usabilidade, foram realizadas por sete indivíduos que desempenham funções na EBS como desenvolvedores, *testers* e *designers*, com diferentes conhecimentos prévios sobre o uso da plataforma *web*. Todos utilizaram contas com permissões de administração para puderem aceder a ambos os portais. O *tenant* tinha os três módulos subscritos e ativos. As campanhas de *eLearning*, associadas às suas contas, eram conteúdos em formato de vídeo e possuíam questionário. Estas tarefas, descritas na Tabela 6.3, contam com a interação do utilizador nas principais funcionalidades da solução.

Tabela 6.3 – Tarefas definidas para testes de usabilidade

ID	Descrição
T1	Escolher o *tenant* associado à conta e proceder à autenticação.

T2	Navegar até ao módulo de eLearning, presente no portal de utilizador, abrir uma campanha associada a si e visualizar o conteúdo.
T3	Responder ao questionário da campanha que acabou de visualizar.
T4	Navegar até ao módulo de Phishing Simulator, presente no portal de administrador, abrir uma campanha e visualizar os eventos de um utilizador.
T5	Navegar até ao módulo de Policy Manager, presente no portal de utilizador, abrir uma política associada a si, visualizar o seu conteúdo e aceitar a mesma.

De modo a interpretar resultados, para além de sugestões dos utilizadores, foi pedido a estes que, durante a realização das tarefas, avaliassem o grau de dificuldade de execução de cada tarefa, presente na Tabela 6.3, numa escala de 1 a 3. O 1 corresponde a fácil, o 2 a um grau médio de dificuldade e o 3 a difícil.

Tabela 6.4 – Graus de dificuldade das tarefas dos testes de usabilidade

Tarefa	**Média real de grau de dificuldade**	**Média arredondada de grau de dificuldade**
T1	1,29	1
T2	1,71	2
T3	1,14	1
T4	1,57	2
T5	1,29	1

Ao observar a Tabela 6.4 é percetível que não existe um grande grau de dificuldade ao executar tarefas na solução, não havendo qualquer registo de grau 3. A solução pode ser considerada intuitiva e o seu *workflow* facilita a sua utilização. As tarefas com grau 2 de dificuldade resultam do facto de ser a primeira vez que os utilizadores usam a solução e terem de aprender como funciona a navegação entre portais e componentes. Por exemplo, a tarefa TF4, que implica trocar de portal e, posteriormente, trocar de módulo teve um registo médio de grau 2, mas após esta aprendizagem, a tarefa TF5 que implica o mesmo, encontra-se no grau 1.

Juntamente a estas avaliações, os utilizadores informaram que tiveram alguns problemas que foram melhorados em posteriores versões da solução. De seguida são apresentados esses problemas e as respetivas soluções para os mitigar:

- Alguns utilizadores informaram que por vezes não sabiam em que módulo se encontravam, sendo que nada o indicava. Para isto, foi adicionado o nome do módulo, em que o utilizador se encontra, ao topo do componente;
- No componente de eLearning, em modo de utilizador, na apresentação dos diversos conteúdos e campanhas, não existia nenhuma indicação do formato do conteúdo. De forma a resolver este problema, foi adicionado um símbolo no topo das *thumbnails* dos conteúdos que indica se estes são em formato PDF;
- Alguns botões de navegação ou para abrir e fechar componentes tinham uma área de toque muito limitada, o que impactava a responsividade expectável da solução. As áreas de toque de diversos botões foram aumentadas para facilitar a execução das ações.

Capítulo 7

Conclusão

Este capítulo apresenta uma visão geral sobre o projeto realizado ao longo do estágio. Para além de um resumo do trabalho realizado, que relata os diferentes pontos de vista deste projeto, académico e profissional, e menciona os principais pontos que levaram a determinadas tomadas de decisão. A perspetiva do possível trabalho a desenvolver no futuro é descrita no fim deste capítulo.

7.1 Trabalho realizado

Este trabalho foi desenvolvido no âmbito da disciplina de Projeto de Engenharia Informática e teve como finalidade o desenvolvimento de uma solução de cibersegurança para dispositivos móveis. Esta solução conta com a integração dos módulos "eLearning", "Policy Manager" e "Phish Simulator" em dois diferentes portais, o de administrador e o de utilizador. O primeiro facilita a consulta de diferentes dados a nível administrativo e o segundo permite realizar ações como visualizar conteúdo e realizar campanhas. O trabalho foi desenvolvido durante o estágio numa empresa de acolhimento, a Emvenci Business Services.

Tendo em conta que o objetivo deste estágio passou por desenvolver um projeto que permitisse abordar novas tecnologias e metodologias de trabalho, pode concluir-se que este foi cumprido. A definição de conceitos importantes, que auxiliaram a estruturação do projeto, foi possível devido a conhecimentos adquiridos durante o percurso académico. Do ponto de vista profissional, o desenvolvimento deste projeto possibilitou a aprendizagem de metodologias de trabalho e a perceção de como um projeto é desenvolvido em ambiente empresarial.

A utilização da *framework* NativeScript permitiu o desenvolvimento da solução, cumprindo os requisitos definidos, tendo sido possível realizá-lo com recurso a algumas tecnologias já utilizadas na plataforma *web*, como por exemplo, o Angular. Um dos principais requisitos da solução, é a compatibilidade em ambos os sistemas, Android e iOS, requisito este que é possibilitado por esta *framework*. No início do projeto, foi

investigado e pensado o uso de *code sharing* do Angular NativeScript, de modo a desenvolver a solução móvel no mesmo repositório da plataforma *web*. Isto é, o mesmo repositório teria a plataforma *web* e a solução móvel, e estas teriam algumas definições e implementações próprias, mas partilhariam algumas lógicas e serviços. Devido à grande dimensão da plataforma *web*, do seu repositório e das diversas dependências não suportadas nos diferentes sistemas, a ideia de usar o *code sharing* foi abandonada e decidiu-se manter dois repositórios individualmente. Mesmo em repositórios diferentes, sendo que a plataforma e a solução utilizam a mesma linguagem de programação e *framework*, alguns serviços e lógica presentes na plataforma, foram reaproveitados e adaptados no desenvolvimento da solução.

A EBS utiliza CI/CD no desenvolvimento e entrega da sua plataforma e o mesmo processo foi utilizado nesta solução móvel, com algumas diferenças devido à necessidade de comunicar com as lojas de aplicações. O desenvolvimento da solução seguiu sempre o seguinte *workflow*: desenvolver ou corrigir problemas em *continuous integration*, gerar *builds*, colocar nos ambientes de testes das lojas de aplicações, e, caso haja problemas, o processo volta ao início, caso contrário, o código da versão que se encontra em testes junta-se ao de produção e a versão é promovida ao ambiente de produção da loja, tornando-se disponível a todos os utilizadores.

7.2 Trabalho futuro

A presente secção está omissa devido à confidencialidade inerente ao projeto.

Bibliografia

[1] "Approaching Mobile, Understanding the Three Ways to Build Mobile Apps.," [Online]. Available: https://www.telerik.com/docs/default-source/whitepapers/choose-right-approach-mobile-app-developmentbb581d10116543e79a9febdb187fd0a3.pdf?sfvrsn=0.

[2] "Mobile app development: native vs web vs hybrid," [Online]. Available: https://clearbridgemobile.com/mobile-app-development-native-vs-web-vs-hybrid/.

[3] "Hybrid apps vs Native apps: what should you build?," [Online]. Available: https://themanifest.com/development/hybrid-apps-vs-native-apps-which-should-you-build.

[4] "React Native: Bringing modern web techniques to mobile," [Online]. Available: https://code.fb.com/Android/react-native-bringing-modern-web-techniques-to-mobile/.

[5] "Technical Overview - NativeScript Docs," [Online]. Available: https://v7.docs.nativescript.org/core-concepts/technical-overview.

[6] "Difference between MVVP and MVP," [Online]. Available: http://www.differencebetween.net/technology/difference-between-mvvm-and-mvp/.

[7] S. Mumbaikar e P. Padiya, *Web Services Based On SOAP and REST Principles*, pp. 1-4, 2013.

[8] C. M. Garcia e R. Abilio, *Systems Integration Using Web Services, REST and SOAP: A Practical Report*, pp. 34-41, 2017.

[9] M. Ali, M. F. Zolkipli, J. M. Zain e S. Anwar, "2018".*Mobile Cloud Computing with SOAP and REST Web Services.*

[10] Y. B. Leau, W. K. Loo, W. Y. Tham e S. F. Tan, *Software Development Life*

Cycle AGILE vs Traditional Approaches, vol. 37, nº 1, pp. 162-167, 2012.

[11] Bender RBT Inc., *Systems Development Lifecycle: Objectives and Requirements,* 2003.

[12] T. Dyba˚e T. Dingsøyr, *Empirical studies of agile software development: A systematic review,* pp. 833-859, 2008.

[13] G. Karataş, F. Can, G. Doğan, C. Konca e A. Akbulut, *Multi-tenant architectures in the cloud: A systematic mapping study,* pp. 1-4, 2017.

[14] C.-P. Bezemer, A. Zaidman, B. Platzbeecker, T. Hurkmans e A. ’. Hart, *Enabling Multi-Tenancy: An Industrial Experience Report,* pp. 1-8, 2010.

[15] N. M. Edan, *Limitations of Using Pooling and Long Polling Mechanisms,* 2018.

[16] J. A. Whittaker, *What Is Software Testing? And Why Is It So Hard?,* 2000.