Modelando produtos IoT com a abordagem DDM

Modeling IoT products with the DDM approach

DOI:10.34117/bjdv7n1-267

Recebimento dos originais: 10/12/2020
Aceitação para publicação: 08/01/2021

Ana Patricia Fontes Magalhães Mascarenhas
Doutora em Ciência da Computação
Instituição: Universidade Salvador
Endereço: Avenida Luís Viana, 3100-3146 Pituaçu - Salvador - BA, 41720-200
E-mail: ana.fontes@unifacs.br

Sérgio Martins Fernandes
Doutor em Engenharia da Computação pela USP
Instituição: Universidade Salvador
Endereço: Avenida Luís Viana, 3100-3146 Pituaçu - Salvador - BA, 41720-200
E-mail: sergiomfernandes63@gmail.com

Gabriel Borges Calheiros
Estudante de Ciência da computação
Instituição: Universidade Salvador
Endereço: Avenida Luís Viana, 3100-3146 Pituaçu - Salvador - BA, 41720-200
E-mail: gabi_borges01@hotmail.com

Fábio Duarte Freitas
Estudante de Ciência da computação
Instituição: Universidade Salvador
Endereço: Avenida Luís Viana, 3100-3146 Pituaçu - Salvador - BA, 41720-200
E-mail: freitasfdf28@gmail.com,

Victor Fernandes Baião Raton
Estudante de Ciência da computação
Instituição: Universidade Salvador
Endereço: Avenida Luís Viana, 3100-3146 Pituaçu - Salvador - BA, 41720-200
E-mail: vfbraton@gmail.com

Guilherme L. G. Lefrançois
Estudante de Ciência da Computação
Instituição: Universidade Salvador
Endereço: Avenida Luís Viana, 3100-3146 Pituaçu - Salvador - BA, 41720-200
E-mail: gui.lefrancois21@gmail.com

Marcela Braga Bahia
Estudante de Ciência da Computação
Instituição: Universidade Salvador
Endereço: Avenida Luís Viana, 3100-3146 Pituaçu - Salvador - BA, 41720-200
E-mail: mahbraga0@gmail.com

RESUMO

O Desenvolvimento Dirigido por Modelos (DDM), é uma abordagem de desenvolvimento de software onde os modelos são os artefatos principais do desenvolvimento. Nesta, modelos de aplicações são construídos em alto nível de abstração e convertidos em modelos menos abstratos até a geração do código fonte do sistema. A abordagem DDM tem se mostrado apropriada para desenvolver aplicações inseridas em ambientes heterogêneos, tais como os que envolvem diferentes plataformas e linguagens de programação, pois possibilita que sistemas sejam modelados independente de plataforma e que os modelos sejam reusados para geração de código em plataformas específicas. Neste contexto estão inseridos os sistemas para Internet das Coisas (IoT – *Internet of Things*), responsáveis por interconectar e integrar dispositivos do mundo físico ao mundo virtual. A IoT propõe que qualquer objeto físico possa se conectar à internet através de algum tipo de tecnologia. Desta forma, lida frequentemente com dispositivos diversos que utilizam tecnologias próprias e precisam estar interconectados para prover um serviço. Nesta direção, nosso trabalho investiga o uso de DDM no desenvolvimento de produtos IoT. Neste artigo, especificamente, apresentamos o projeto de um produto IoT para monitoramento e controle de reservatórios de água em residências com o objetivo de identificar os modelos que devem ser construídos ao longo do desenvolvimento de produtos desta natureza. O produto foi projetado e implementado manualmente. O trabalho mostrou que diversos modelos foram necessários para projetar hardware e software, tais como modelo de requisitos, arquitetura, e máquinas de estado. Estes modelos serão adaptados em projetos futuros seguindo a abordagem DDM para possibilitar a geração automática de código.

Palavras-chave: Desenvolvimento dirigido a modelos, Internet das coisas, IoT, linguagens de modelagem específicas de domínio.

ABSTRACT

Model Driven Development (DDM), is a software development approach where models are the main artifacts of development. In this, application models are built at a high level of abstraction and converted into less abstract models until the system's source code is generated. The DDM approach has proven appropriate for developing applications inserted in heterogeneous environments, such as those involving different platforms and programming languages, because it allows systems to be modeled platform-independent and models to be reused for code generation in specific platforms. In this context are inserted the systems for Internet of Things (IoT), responsible for interconnecting and integrating devices from the physical world to the virtual world. IoT proposes that any physical object can connect to the Internet through some kind of technology. In this way, it frequently deals with diverse devices that use their own technologies and need to be interconnected to provide a service. In this direction, our work investigates the use of DDM in the development of IoT products. In this article we specifically present the design of an IoT product for monitoring and control of water reservoirs in homes with the aim of identifying the models that must be built throughout the development of products of this nature. The product was designed and implemented manually. The work showed that several models were necessary to design hardware and software, such as requirements model, architecture, and state machines. These models will be adapted in future projects following the DDM approach to enable automatic code generation.

Keywords: Model driven development, Internet of things, IoT, domain specific modeling languages.

1 INTRODUÇÃO

Na ciência da computação é notório o aumento da complexidade dos sistemas, seja por fatores técnicos, como heterogeneidade de plataformas, ou por fatores não técnicos, como a própria complexidade dos domínios hoje automatizados pelos sistemas de informação. Um exemplo de sistema complexo que tem ganhado cada vez mais notoriedade são os sistemas para Internet das coisas (IoT). IoT possibilita que objetos do cotidiano das pessoas possam estar interconectados à internet provendo serviços diversos. Contudo, questões desafiadoras tanto sociais quanto tecnológicas ainda precisam ser tratadas, como por exemplo, como alcançar interoperabilidade entre os dispositivos interconectados, como estes dispositivos podem ser adaptados a mudanças no ambiente, como prover segurança, entre outros e pesquisas estão sendo realizadas para desenvolvimento de soluções que apoiem os requisitos da IoT [1], [2].

Dentre as abordagens que estão sendo investigadas para apoiar o desenvolvimento de sistemas para IoT [3][4][5] está o Desenvolvimento Dirigido por Modelos (DDM) [6], pois utiliza mecanismos de abstração para lidar com a complexidade no desenvolvimento de software. Modelos são utilizados para enfatizar características relevantes de um domínio, projetar a arquitetura de sistemas, detalhar o comportamento, dentre outros aspectos que envolvem o desenvolvimento de aplicações computacionais. Na DDM os modelos em alto nível de abstração são construídos e convertidos para níveis mais baixos, através de uma cadeia de transformações até gerar o código fonte da aplicação [7]. Na abordagem DDM linguagens específicas de domínio (DSLs) são comumente usadas na construção dos modelos, pois elas encapsulam o conhecimento do domínio possibilitando a criação de modelos mais expressivos [8].

Nosso trabalho investiga o uso de modelos na especificação de produtos IoT e a possível utilização da abordagem DDM para posterior geração do sistema. Neste artigo, especificamente, apresentamos o projeto de um produto IoT para monitoramento e controle de reservatórios de água em residências com o objetivo de identificar os modelos que devem ser construídos ao longo do desenvolvimento de produtos desta natureza. Os resultados deste trabalho serão utilizados como ponto de partida para desenvolver produtos IoT na abordagem DDM.

A metodologia utilizada para desenvolver esse projeto se iniciou com a aquisição de conhecimento teórico sobre DDM e IoT através da leitura de artigos. Em seguida construímos pequenos protótipos experimentais para consolidar o conhecimento, tais como: um semáforo, um relógio digital, um jogo de carro, entre outros. Em um segundo

momento definimos um produto IoT a ser construído de forma tradicional (sem o uso da abordagem DDM) com o objetivo de identificar as etapas necessárias para a construção e principalmente as necessidades de modelagem relacionadas ao projeto. O produto desenvolvido compreende duas partes: uma parte física com placas Arduino, sensor, atuadores, entre outros elementos; e um software web responsável pelo controle da parte física. Essas partes foram representadas em modelos diversos. A etapa final de desenvolvimento do projeto consistiu na realização dos testes do produto em laboratório e do registro dos resultados em relatório técnico.

A Seção 2 apresenta o referencial teórico relacionado ao projeto. Em seguida a Seção 3 apresenta alguns trabalhos que também investigam o uso de DDM na construção de produtos IoT. A Seção 4 apresenta o produto construído enfatizando os modelos especificados. Finalmente na seção 5 são apresentadas as considerações finais e trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Esta seção apresenta os conceitos de IoT e desenvolvimento dirigido por modelos.

2.1 DESENVOLVIMENTO DIRIGIDO POR MODELOS (DDM)

O Desenvolvimento Dirigido a Modelos (DDM) é uma abordagem de desenvolvimento de software que usa modelos como os artefatos principais do desenvolvimento. Desta forma, caracteriza-se pela mudança da ênfase do desenvolvimento, antes no código, para os modelos. Neste contexto, modelos em alto nível de abstração são transformados de maneira automática / semi-automática, através de uma cadeia de transformações, em modelos menos abstratos até chegar ao código fonte da aplicação [6].

Modelos são especificados a partir de metamodelos, os quais são definidos em uma linguagem de modelagem através da restrição e/ou extensão dos construtores da linguagem para representar um espaço de aplicações possíveis dentro de uma ou mais áreas de conhecimento. Um metamodelo contém a semântica e a sintaxe necessárias para a construção dos modelos, ou seja, os construtores disponíveis da linguagem de modelagem, suas relações e regras de modelagem. Quando um modelo é especificado a partir de um metamodelo, ele representa uma instância daquele metamodelo [7].

Os modelos devem obedecer a uma relação de conformidade com seus metamodelos. Um modelo é dito conforme com um metamodelo se estiver sintaticamente correto e atender as restrições impostas pelo metamodelo [7].

Modelos são convertidos em outros modelos e em código através de transformações de modelos. A transformação de modelos consiste em mapeamentos entre modelos fonte e modelos alvo. Entende-se por modelo fonte o modelo de entrada de uma transformação e por modelo alvo o modelo resultante da transformação. Na execução de uma transformação, modelos alvos podem ser gerados ou modificados a partir de modelos fonte ou podem ainda ser apenas checados se atendem as relações estabelecidas na transformação [6].

Uma transformação é definida no nível de metamodelo e compreende um conjunto de regras que juntas descrevem como modelos podem ser transformados em outros modelos. Estas regras mapeiam elementos do metamodelo do modelo fonte no metamodelo do modelo alvo [6].

2.2 INTERNET DAS COISAS E DDM

A Internet das Coisas (IoT) é a rede de coisas físicas e aparelhos virtuais que se comunicam e interagem entre si. O termo "coisas" refere-se a uma ampla gama de diferentes dispositivos como veículos, dispositivos vestíveis (wearables), sensores físicos (por exemplo, um sensor de umidade), sensores virtuais (por exemplo, um analisador de teclas), etc. Essas coisas estão interconectadas e podem trocar dados, o que permite monitorar e controlar dispositivos remotamente [2].

Como hardware embutido (um facilitador chave para dispositivos IoT) está se tornando mais acessíveis, mais e mais dispositivos são desenvolvidos e produzidos todos os anos. Devido ao crescente número de dispositivos complexidade dos sistemas, a IoT enfrenta diferentes desafios, como (i) heterogeneidade, (ii) disponibilidade de dispositivos, (iii) grande quantidade de dados, (iv) segurança e privacidade, etc. Surgiram vários conceitos de IoT para enfrentar alguns desses desafios [1].

Uma forma, entre outras, de lidar com a complexidade do desenvolvimento de soluções IoT é o uso de técnicas de DDM. Assim, linguagens de modelagem, incluindo DSLs, são usadas junto com técnicas como modelos semânticos, geração de código e transformações para reduzir o esforço e os custos de desenvolvimento de software IoT. Por exemplo, através do poder de abstração de modelos, a representação de um sistema pode ser mais compreensível. Modelos podem ser alinhados para criar uma representação

integrada de sistemas de diferentes pontos de vista a fim de (i) raciocinar sobre a consistência, (ii) melhorar o estrutura sem alterar o comportamento geral, e (iii) traduzi-los para outros formalismos para geração de código e simulação. Vários ambientes de modelagem oferecem suporte a ferramentas para essas tarefas, que podem ser adaptadas para a linguagem de modelagem usada.

3 TRABALHOS RELACIONADOS

O uso da abordagem DDM para desenvolver sistemas para IoT já tem sido investigado em alguns trabalhos. Por exemplo, em [11], [12] e [13] os autores utilizam DDM para modelar a inclusão/adaptação de dispositivos na rede. Já em [14] a DDM é utilizada para modelar a composição de serviços e em [15] a auto adaptação do dispositivo às mudanças do ambiente em que está inserido. Alguns trabalhos propõem DSLs para modelar IoT [16], [17], outros contemplam recursos para geração de código [18]. Embora os estudos sejam preliminares, já apontam direções que indicam a contribuição que a DDM pode dar para essa nova categoria de sistemas.

Este trabalho se diferencia dos demais por investigar quais os possíveis modelos que podem ser utilizados na especificação das várias partes que compõem o desenvolvimento de produtos IoT em direção a adaptação desses para o contexto de DDM

4 MODELAGEM DO SISTEMA DE MONITORAMENTO E CONTROLE DE CAIXAS DE ÁGUA

Esta seção apresenta a especificação do Sistema de Monitoramento e Controle de Caixas de Água (MoCoCA), enfatizando os modelos construídos para projetar o sistema. Inicialmente será feita uma breve introdução sobre o sistema e em seguida apresentados os modelos especificados.

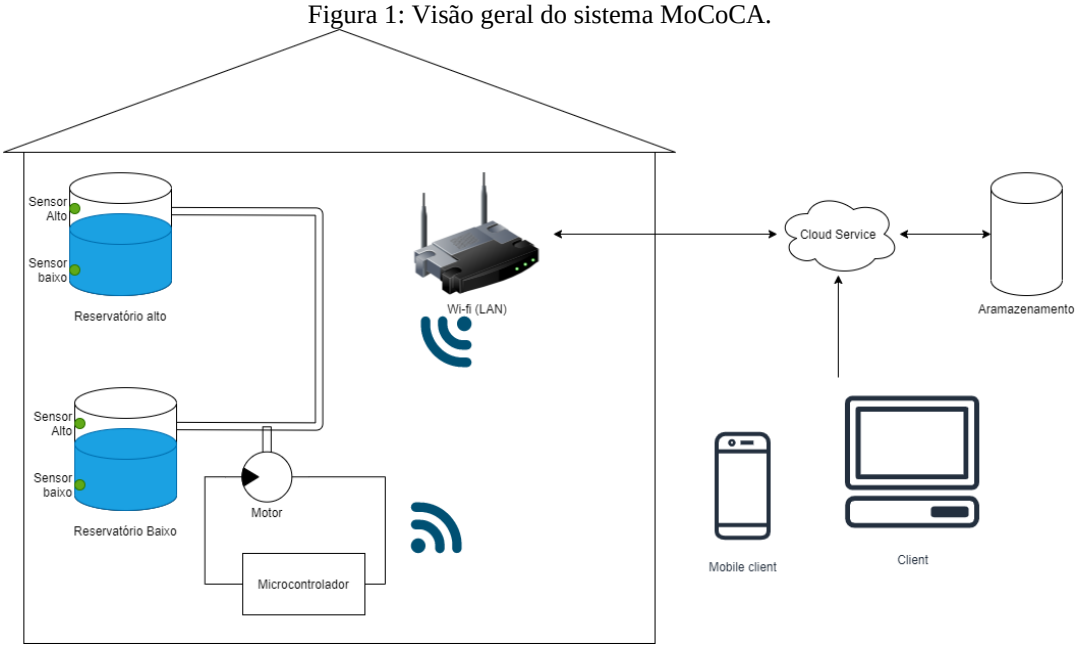
A Figura 1 ilustra a visão geral de uma residência e como o Sistema MoCoCa funciona nesta residência, onde podemos ver os dois reservatórios monitorados por sensores de volume cada, (nível mínimo e nível máximo de cada reservatório), uma bomba hidráulica (eletrobomba) um dispositivo embarcado (microcontrolador) , um dispositivo com sinal de rede *wireless* (roteador *wifi*) e os componentes do serviço (*cloud service* , armazenamento, clientes)

De forma resumida os sensores coletam as informações, passando estas ao dispositivo embarcado que as delega por meio do sinal *wireless* ao *cloud service* (usando

a rede local como *gateway* de saída). O atuador pode ser acionado remotamente por meio do serviço que se comunica com o dispositivo embarcado.

O produto desenvolvido visa controlar o sistema de distribuição de água entre dois reservatórios de uma residência. É possível acompanhar os níveis de água em cada reservatório e acionar ou desligar a bomba que enche esses reservatórios. Todo o controle é monitorado pelo usuário através de um aplicativo de celular conectado em tempo real ao sistema físico instalado na residência. O sistema pressupõe que em uma residência existe o reservatório principal de água e um reservatório superior secundário. O reservatório principal fornece água para o secundário que por sua vez abastece a residência.

Figura 1: Visão geral do sistema MoCoCA.



O projeto é composto por sensores, responsáveis por fazer a leitura de informações como volume e pressão hidráulica do sistema; atuadores, válvulas de controle e motores que podem alterar o comportamento do sistema; um dispositivo embarcado [10] o qual se conecta ao sensores e atuadores e interpreta tais informações, além de se comunicar com os serviços web por meio do MQTT (*Message Query Telemetry Transport*).

Os dados coletados do processo bem como o estado dos atuadores e caixas d'água são monitorados e exibidos em uma interface em tempo real, para acompanhamento por possíveis usuários finais, além de estarem disponíveis por meio das API's clientes do serviço, assim permitindo a integração deste com outros sistemas tais como BotNets, e sistemas que implementam Big Data, IA, entre outros.

A produção do sistema foi de grande relevância para o projeto devido aos importantes dados obtidos durante o processo, que serão úteis para a segunda fase do projeto que está se iniciando.

5 ESPECIFICAÇÃO DO SISTEMA MOCOCA

A ideia geral deste projeto consiste na gestão de um sistema hidráulico doméstico de forma mais inteligente e que possa ser integrada a outros serviços e sistemas modernos. O sistema monitora e controla os reservatórios de água da residência e o objetivo é manter o reservatório superior cheio na maior parte do tempo.

O desenvolvimento deste sistema compreendeu várias etapas relacionadas ao hardware, software e comunicação entre eles. Neste artigo enfatizamos os modelos definidos para projetar a arquitetura de todo o produto e os modelos relacionados ao software.

Para o desenvolvimento do sistema foram utilizadas as linguagens C++ (Arduino framework) e Javascript, banco de dados MongoDB, além de runtime Node.js e a biblioteca React.js.

Os seguintes modelos foram construídos ao longo do projeto: modelo de requisitos do sistema, modelo de arquitetura do software, modelo do banco de dados e diagrama de estados, ilustrados a seguir.

5.1 MODELO DE REQUISITOS DO SISTEMA

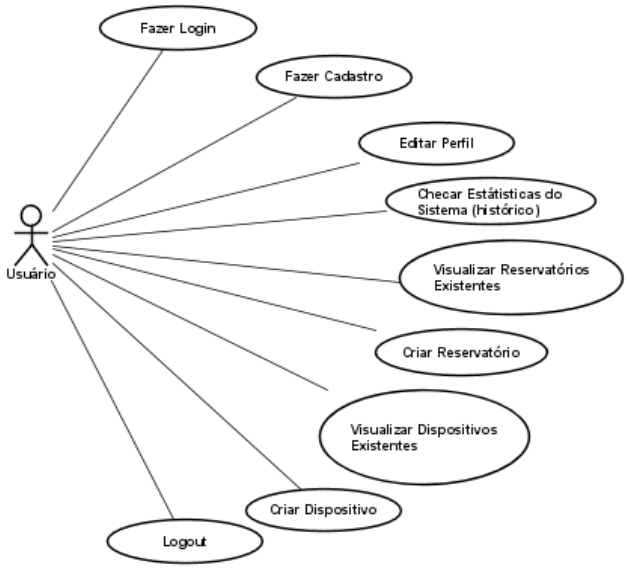
Trata-se das características e limitações que o sistema deve atender para ser considerado válido ou concluído no que diz respeito ao seu desenvolvimento [9]. Tais requisitos serão divididos em dois grupos, requisitos funcionais que tratam de necessidades claras e delineadas da solução e requisitos não funcionais que tratam de características genéricas para a solução.

Considerando a proposta da solução e os problemas que a mesma tendem a resolver, foram definidos os seguintes requisitos funcionais.

- O sistema deve permitir ao usuário acompanhar em tempo real o estado das “Coisas” que o sistema contempla
- O usuário deve dispor de ferramental que o permita criar e editar coisas associadas a um dispositivo microcontrolador voltado para a comunicação
- O usuário deve possuir um histórico das mudanças ocorridas
- O sistema deve dar suporte a notificações em tempo real

Com base nestes requisitos foi então modelado o diagrama de casos de uso que ilustra os requisitos funcionais derivados para o sistema web que será utilizado pelo usuário, proprietário da residência (Figura 2). Através da interface, registrando um novo usuário e autenticando-se, é possível interagir com o sistema físico para criar novos reservatórios d'água, visualizar reservatórios existentes, criar e visualizar dispositivos (atuadores e sensores) e checar o histórico de atividades do sistema (exibe os dispositivos criados e datas de modificação).

Figura 2: Diagrama de casos de uso com os requisitos da aplicação Web.



A partir da modelagem dos requisitos funcionais, foram analisadas as necessidades desta solução para definirmos os requisitos não funcionais. No mercado atual existe uma gama de empresas e casos de sucesso a respeito de aplicações de larga escala e interatividade tais como Uber, Facebook, AWS (Amazon Web Service). Ainda que existam variações destes modelos, ambas possuem em comum a perspectiva de que uma aplicação moderna deve pelo menos garantir uma eficiência na entrega de resultados, além de promover uma experiência contínua de qualidade para os usuários.Tais empresas definem como fundamento para tais aplicações a implementação de conceitos tais como tolerância a falhas, alta performance e fácil gerenciamento. Considerando esses aspectos, definimos os seguintes requisitos não funcionais.

- **Escalabilidade**: capacidade de aumentar a capacidade de demanda de uma forma fácil, pois nessa solução para otimizar o custo benefício teremos que evitar

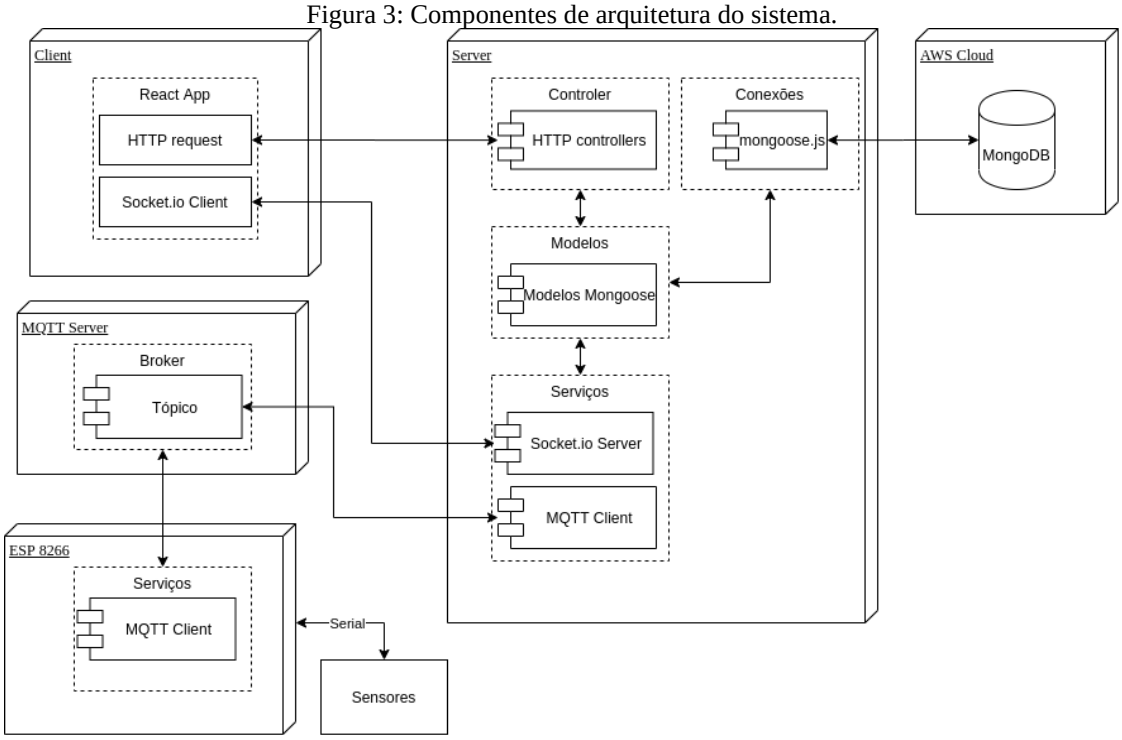
ociosidade de recurso computacional. Dessa forma a solução precisa "escalar" estar apta a inserção de novos dispositivos (tanques de água, sensores e atuadores) sem que seja necessário a intervenção do desenvolvedor. Também deve ser possível desativar os recursos ociosos.

- **Suporte multiplataforma**: capacidade de atender a diferentes tipo de clientes, não apenas em termos de software ou hardware, mas em termos de contexto, pois há uma diferença entre enviar e receber dados de um aplicativo de um dispositivo mobile e o envio e recebimento de dados para um agente microcontrolador ou um dispositivo embarcado.
- **Alta performance**: capacidade de gerenciar uma quantidade massiva de dados que podem ser alterados diversas vezes em um período de tempo.

5.2 MODELO DE ARQUITETURA DO SISTEMA

Com o intuito de atingir tais requisitos, implementou-se uma solução análoga a um serviço usando Node.js para o recebimento, armazenamento e comunicação entre os componentes, aliado ao protocolo de comunicação MQTT e Websocket, além do uso de HTTP para consumo do serviço em geral. Tal solução pode ser vista no modelo de componentes ilustrado na Figura 3 a seguir.

Figura 3: Componentes de arquitetura do sistema.

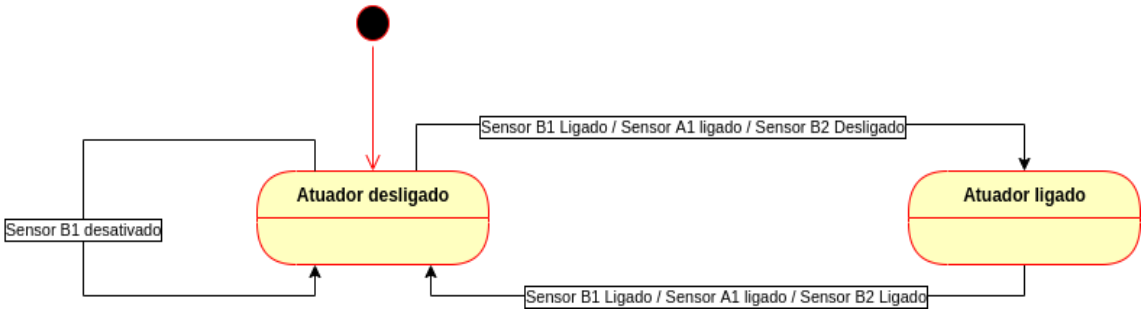


Dessa forma, utilizou-se o ambiente de execução Node.js para a construção de serviços em nuvem que contemplem tais características. A escolha do Node.js como ambiente de execução dá-se por sua natureza assíncrona voltada para trabalhar com atividades intensivas de Entrada/Saída (*Intensive I/O task*) utilizando os recursos computacionais de forma mais eficiente.

5.3 MODELAGEM COMPORTAMENTAL

A Figura 4 ilustra o diagrama de estados que modela o comportamento dos sensores que monitoram os tanques de água. O sistema só irá bombear água se esta estiver disponível no reservatório inferior (se reservatório inferior não estiver vazio) e se o reservatório superior estiver disponível para ser alimentado (se reservatório superior não estiver cheio).

Figura 4: Diagrama de estados do comportamento do sistema



5.4 MODELAGEM DE DADOS

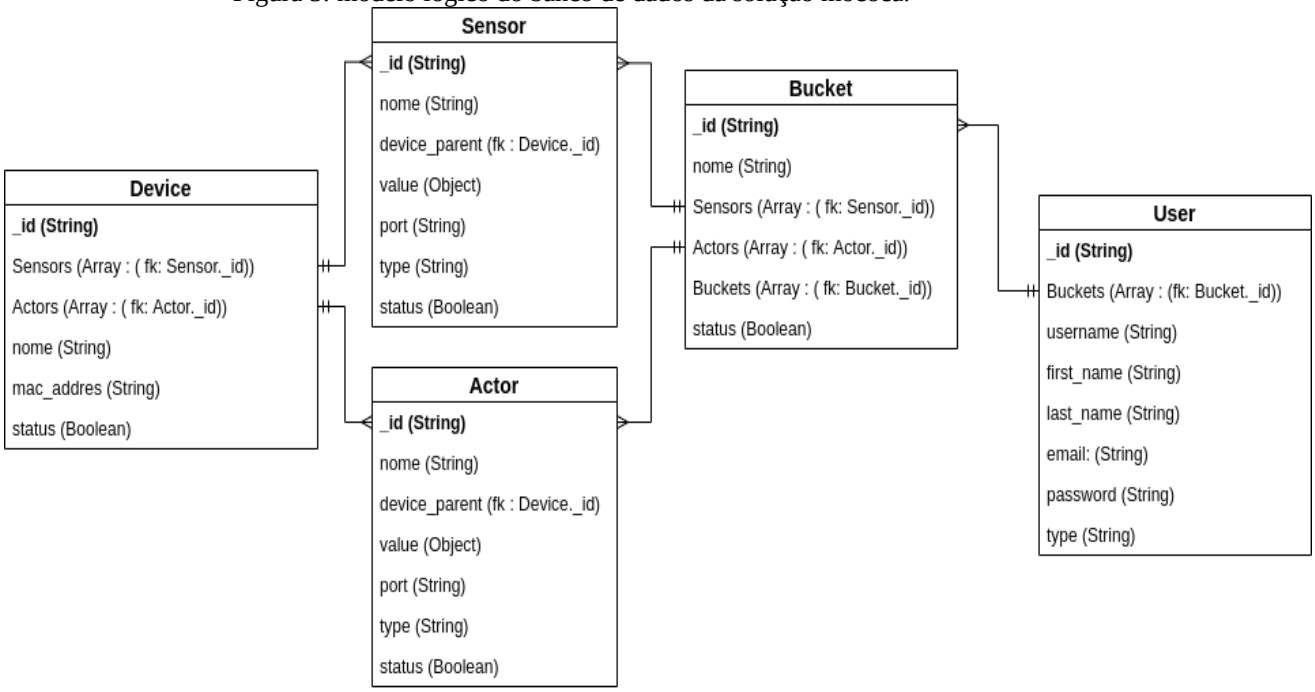
Quanto ao armazenamento de dados, a solução utiliza como base o banco de dados NoSQL MongoDB, que tem como principal característica ser um banco de dados orientado a documentos, onde cada unidade de dados equivale a um documento do tipo JSON (*Javascript String Object Notation*) que possui certa flexibilidade em quesitos de modelagem.

Além de características de performance e de ferramentas nativas que provém capacidade de implementar o MongoDB como um banco de dados distribuído, o mesmo possui suporte nativo a *clusterização* é um sistema de organização baseado em coleções (*collections)*, onde uma coleção representa um agrupamento de documentos que utilizam indexação, sendo esta podendo ser usada para identificar a coleção a qual um documento pertence.

Por não possuir sistema de *script* e modelagem específica o MongoDB define que toda coleção possui os métodos de inserção atualização, remoção e busca (*insert*, *update*, *delete* e *find*) além de outros como contagem (*count*) e *upsert* que seria uma inserção / atualização caso o dado já exista.

Com o intuito de atender não somente essa, mas futuramente outras soluções IoT definiu-se o seguinte modelo lógico para o banco de dados.

Figura 5: modelo lógico do banco de dados da solução mococa.



A Figura 5 descreve cinco entidades que são mantidas no sistema, são elas:

- **Device**, representa um dispositivo, como por exemplo o Arduino, que faz o processamento dos dados no sistema físico. Contém sensores e atuadores conectados a si, um nome, um status e uma identificação de endereço físico;
- **Sensor,** recebe um estímulo físico causado pelo nível d'água no reservatório no momento, e a partir desse, emite um sinal a um dispositivo. Contém um nome, um dispositivo pai conectado, um valor, estado, porta de conexão com o dispositivo e descrição do tipo de sensor;
- **Actor**, atuador, responde a um comando enviado por um dispositivo, realizando mecanicamente a função de ativar ou desativar a bomba hidráulica. Assim como o sensor, tem um nome, um dispositivo pai conectado, um valor, uma porta de conexão com o dispositivo e um estado, além da descrição do tipo de atuador;

- **Bucket**, reservatório de água. Cada reservatório contém sensores, atuadores, outros reservatórios e um estado;
- **User**, representa o usuário que interage com o sistema através da interface web.

7 CONCLUSÕES E TRABALHOS FUTUROS

Neste projeto foi construído um sistema de monitoramento e controle de reservatório de água para residências. Durante o desenvolvimento foram desenvolvidos diferentes modelos de análise e projeto tanto para a parte física quanto para o software do projeto. Esses modelos serão utilizados no projeto em andamento para construção de outros produtos na abordagem DDM.

O produto desenvolvido foi testado em laboratório e se mostrou viável para ser replicado em ambiente real. Foi possível identificar as etapas envolvidas no processo de construção de produtos IoT, tais como o projeto da parte física, o projeto do software, a construção e validação destas partes. Observou-se a necessidade de construir modelos que não são comuns ao desenvolvimento de software convencional, isto é, que estão diretamente relacionados ao domínio de IoT. Por exemplo, foram projetados diagramas simulando a interação do Arduino e a aplicação cliente, com o objetivo de melhorar o entendimento acerca da comunicação entre os atores envolvidos no produto.

O conhecimento adquirido neste trabalho está sendo utilizado em uma nova etapa da pesquisa para construção de produtos IoT utilizando a abordagem DDM.

APOIO FINANCEIRO

FAPESB, CNPQ e UNIFACS através do PIBIC.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] H.-D. Ma, "Internet of Things: Objectives and Scienti¯c Challenges," Journal of Computer Science and Technology, pp. 919-924, 11 2011.

[2] B. Debasis e S. Jaydip, "Internet of Things: Applications and Challenges," Springer Science+Business Media. Wireless Pers Commun, pp. 49-69, 2011.

[3] C. Davide, B. Paolo, K. Prabhakaran, P. Claudio, P. Ferry e P. A. Cultrona, "Industrial application development exploiting IoT vision and model driven programming," em 18th International Conference on Intelligence in Next Generation Networks, 2015.

[4] C. Federico, C. Ivica, D. R. Davide, M. Ivano, P. Patrizio e S. Romina, "Model-Driven Engineering for Mission-Critical IoT Systems," IEEE Software, pp. 46-53, 2017.

[5] C. Federico e S. Romina, "MDE4IoT: Supporting the Internet of Things with Model-Driven Engineering.," 10th International Symposium on Intelligent Distributed, pp. 67-76, 2016.

[6] M. Brambilla, J. Cabot e M. Wimmer, Model-Driven Software Engineering in Practice.,Morgan & Claypool Publishers, 2012.

[7] T. Stahl e M. Volter, Model-Driven Software Development., Wiley, 2010.

[8] S. e. a. Mellor, MDA Distilled, EUA : Addisson-Wesley, 2004.

[9] SOMMERVILLE, Ian. Engenharia de Software. 9a ed., São Paulo: Pearson, 2011.

[10] Embarcados (ames) ,Arduino UNO (2019), Disponivel em: https://www.embarcados.com.br/arduino-uno/ , acessado em 10/05/2019.

[11] Sarfraz, A.; Nol, J. Semantic Enhanced Service Proxy Framework. Em IEEE/ACM Int'l Conference on Green Computing, China, 2010.

[12] Darko, A. ; Aparna Saisree, T.; Arne, B.; Hoan, Q.; Achille, Z. Deliverable 4.3.b Service Discovery and Orchestration Second Release. 2017.

[13] Arne, B. ; Stefan, S. ; Corina Kim,S. ; Abdelmajid, K.; Sebastian, K. Enabling IoT Ecosystems through Platform Interoperability. IEEE Software, pp. 54-61, 2017.

[14] Davide,C. ; Paolo, B.; Prabhakaran, K.; Claudio, P. ; Ferry, P.; Cultrona, P. A. Industrial application development exploiting IoT vision and model driven programming. Em 18th International Conference on Intelligence in Next Generation Networks, 2015.

[15] Guinard, D. Mashing Up Your Web-Enabled Home. Em Current Trends in Web Engineering - 10th International Conference on Web Engineering, Viena, 2010.

[16] Federico,C.; Ivica,C. ; Davide, D. R. ; Ivano, M.; Patrizio, P.; Romina, S. Model-Driven Engineering for Mission-Critical IoT Systems. IEEE Software, pp. 46-53, 2017.

[17] HyunJae, L.; Eunjin, J.; Donghyun, K. ; Jinmyeong, K.; Soonhoi, H. A novel service-oriented platform for the internet of things. Em 7th International Conference on the Internet of Things, Austria, 2017.

[18] Federico, C.; Romina, S. MDE4IoT: Supporting the Internet of Things with Model-Driven Engineering. 10th International Symposium on Intelligent Distributed, pp. 67-76, 2016.