

SRM - SOLUÇÃO PARA GESTÃO DE RELACIONAMENTO COM ESTUDANTE

Cícero Alvarenga Santos Neto
Graduando em Engenharia de Software – Uni-FACEF
ciceroalvarenga2017@hotmail.com

Donato Cardoso Ávila
Graduando em Engenharia de Software – Uni-FACEF
donatocardoso11@gmail.com

Carlos Eduardo de França Roland
Mestre em Desenvolvimento Regional - Uni-FACEF
roland@facef.br

Resumo

Com a crescente globalização e aumento na competitividade no mercado discutem-se novas formas que sejam úteis, práticas e rápidas para acesso a informações para auxiliar nas tomadas de decisões, surge a necessidade de gerenciar a comunicação e o relacionamento com clientes. Para atender a tais necessidades as organizações fazem uso de sistemas de Gerenciamento de Relacionamento com o Cliente, conhecidos como sistemas CRM acrônimo do termo inglês. Tais ferramentas centralizam, organizam e permitem a gestão de informações obtidas dos clientes. Tais necessidades são realidade também no mercado educacional, com a necessidade de se manter e gerenciar dados de alunos em instituições de ensino. Neste contexto, este artigo apresenta o projeto de desenvolvimento de um aplicativo que atenda às necessidades de acesso a informações pela comunidade acadêmica, buscando conquistar maior satisfação e fidelidade de seus estudantes. Tais sistemas têm sido pensados e propostos em diversos mercados e foram classificados como sistemas de Gerenciamento de Relacionamento com o Estudante que, em inglês é designado como Student Relationship Management (SRM). Para o desenvolvimento do projeto foram utilizadas ferramentas, métodos e boas práticas propostas pela Engenharia de Software tais como os diagramas BPMN e os da UML como o de Casos de Uso. Foram realizados os processos do Ciclo de Vida do Desenvolvimento de Software (CVDS) nas etapas de levantamento, documentação e análise de requisitos; projeto e prototipação de banco de dados e de interfaces; implementação de código e testes; além de aplicados conceitos de qualidade e segurança de software. O protótipo funcional implementado consegue oferecer acesso fácil e rápido aos comunicados de professores para os estudantes. Os resultados alcançados nos testes de uso do aplicativo ainda permitiram vislumbrar novas e melhores funcionalidades que poderão se tornar objetos de melhorias futuras do produto.

Palavras-chave: Aplicativo Mobile. BPMN. Diagramas UML. Engenharia de Software. Gestão de Relacionamento com o Estudante. SRM.

Abstract

With the increasing globalization and increased competitiveness in the market, new ways are discussed that are useful, practical and quick to access information to assist in decision making, from this arises the need to manage communication and relationship with customers. To meet these needs, organizations make use of Customer Relationship Management systems, known as CRM systems. Such tools centralize, organize and allow the management of information obtained from customers. Such needs are also a reality in the educational market, with the need to maintain and manage student data in educational institutions. In this context, this article presents the project to develop an application that meets the needs of access to information by the academic community, seeking to achieve greater satisfaction and loyalty from its students. Such systems have been designed and proposed in several markets and have been classified as Student Relationship Management systems (SRM). For the development of the project, tools, methods and good practices proposed by Software Engineering were used, such as the BPMN and UML diagrams such as Use Cases. The Software Development Life Cycle (SDLC) processes were carried out in the requirements survey, documentation and analysis steps; design and prototyping of databases and interfaces; code implementation and testing; in addition to applying software quality and security concepts. The functional prototype implemented can offer easy and fast access to university communication by professors for students. The results achieved in the tests of use of the application still allowed us to glimpse new and better functionalities that could be objects of future improvements of the product.

Keywords: BPMN. Mobile Application. Software Engineering. SRM. Student Relationship Management. UML Diagrams.

1 Introdução

Com o advento da globalização as empresas, para se manterem sustentáveis no mercado ágil e dinâmico, necessitam cada vez mais de sistemas de informação que ofereçam recursos eficazes para auxiliar nas tomadas de decisões do dia a dia (GHOLAMI, 2015). Com busca de conquistar vantagem competitiva, muito se tem discutido sobre teorias e práticas para gerenciar o relacionamento com clientes. Neste contexto, sistemas de informação classificados como de Gerenciamento de Relacionamento com o Cliente (do inglês *Customer Relationship Management* - CRM) têm se mostrado um dos melhores meios de se alcançar melhor posicionamento mercadológico.

Segundo Gomes, Fávero e Lucas (2015), para alcançar o sucesso uma Instituição de Ensino Superior (IES) precisa definir suas visão, missão e valores para apresentar adequadamente sua imagem no mercado. Tal imagem deve ser sustentada por uma boa relação de todos os envolvidos (alunos, professores e funcionários), uma vez que são a

razão da instituição existir. Assim é preciso gerenciar correta e adequadamente essas relações, para que os alunos mantenham o interesse em relação aos produtos e serviços oferecidos pela IES, aumentando a retenção dos estudantes nos cursos e fazendo com que retornem em novos momentos de aperfeiçoamento profissional e acadêmico. Para suportar este complexo processo há a necessidade de uso de ferramentas que facilitem o gerenciamento desses relacionamentos.

O CRM é uma ferramenta de marketing, composta por diferentes tipos de ações que são usadas para analisar a eficiência da comunicação da empresa com seus clientes. O objetivo do CRM é melhorar o relacionamento entre o cliente e a empresa (GHOLAMI, 2015).

Em uma IES os clientes são os alunos, sugerindo assim a adequação dos conceitos e processos do CRM a uma nova classe de sistema de informação denominada Gerenciamento de Relacionamento com o Estudante (do inglês *Student Relationship Management* - SRM). A partir do uso de um SRM, a IES poderá entender melhor as necessidades de seus estudantes e as demandas do mercado, oferecendo produtos mais adequados e com isso alcançar uma reputação melhor frente à acirrada concorrência. Tais sistemas aumentam a disponibilidade de informação gerencial oferecendo à IES melhores resultados operacionais.

O projeto aqui apresentado tem como objetivo descrever o processo de desenvolvimento de um aplicativo com funcionalidades de um SRM que permite o acesso a informações e comunicados direcionados aos estudantes da IES objeto do estudo de caso. São apresentados detalhes das especificações do projeto tais como as partes interessadas na solução e as suas necessidades que foram supridas pelo protótipo funcional implementado.

Os requisitos do sistema foram levantados, documentados e analisados para então ser projetado o aplicativo, com base nos conceitos, métodos e boas práticas propostos pela Engenharia de Software, elevando a qualidade do produto final e diminuindo o retrabalho na implementação do *software*. A partir da versão inicial desenvolvida percebeu-se a possibilidade de oferecer novas funcionalidades, que foram documentadas, podendo ser objeto de estudo para uma futura evolução do aplicativo.

2 Objetivo do Projeto

Vantagem competitiva é o principal motivo pelo qual os estudantes escolhem uma determinada instituição de ensino e não seus concorrentes. A vantagem competitiva pode ser vista em dois fatores determinantes no mercado educacional: o primeiro é o valor percebido pelos estudantes na oferta dos cursos e o segundo são os serviços únicos que a instituição apresenta para se diferenciar no mercado (COLLIS e RUKSTAD, 2008).

Uma vantagem competitiva se torna sustentável quando os concorrentes são incapazes de oferecê-la ou superá-la. Acesso à informação é imprescindível para se adequar à dinâmica do mercado, que é facilitado pelo uso de sistemas Gerenciadores

de Relacionamento com o Estudante que potencializa a tomada de decisões estratégicas para se alcançar melhores resultados econômicos e financeiros (MENDONÇA e TERRAZ, 2017).

Um sistema SRM permite a centralização do acesso aos dados operacionais coletados nos diversos departamentos que estruturam os processos administrativos em uma IES. A adoção de soluções de processamento e armazenamento em nuvem oferece aos usuários a facilidade do acesso rápido e em qualquer lugar proporcionando aumento de fidelização dos alunos, o que é mais barato do que conquistar novos e, com a automatização de processos, se economiza tempo aumentando a produtividade da administração escolar.

Também é possível otimizar a colaboração entre os diversos departamentos da instituição - marketing, financeiro, acadêmico além serviços de apoio educacional como parcerias para estágios e ofertas de cursos de extensão e de pós-graduação.

Buscou-se, neste projeto, aumentar o desempenho das práticas de gestão de relacionamento comuns em outras áreas de negócio para melhorar a relação entre a escola e seus alunos. Foi projetado e implementado um aplicativo para uso em dispositivos móveis para o agendamento de salas para a realização de aulas, especialmente em laboratórios, e para registro de eventos e atividades acadêmicas, para que os alunos possam acessar as informações em tempo real desde que tenham acesso à internet.

Após a criação desse aplicativo a comunicação entre os coordenadores, professores e estudantes poderá ser mais efetiva e dinâmica, pelo fato da informação estar disponível a todos que fazem uso do aplicativo, consultando inclusões, alterações e cancelamentos de agendamentos realizados por professores e coordenadores dos cursos da instituição.

3 Referencial Teórico

A Engenharia de Software (ES) propõe procedimentos, práticas e variadas ferramentas que permitem aos profissionais de Tecnologias Digitais de Informação e Comunicação (TDIC) desenvolverem software com alta qualidade e com custos adequados. A adoção dos métodos ES resultam em aumento do conhecimento do domínio do software, do seu desempenho e da capacidade de entrega do projeto (PRESSMAN, 2016). Possibilita, ainda, que os desenvolvedores construam softwares que sejam adaptados à realidade dos usuários finais. De acordo com Wazlawick (2013) conceituar e pôr em prática os processos de Engenharia de Software não é uma tarefa simples, porém necessária, onde antes de tudo se deve pensar que cada etapa é diferente da outra e as técnicas utilizadas dependem do tipo de abordagem que se decide tomar para desenvolver o software.

Assim, embora isso não seja consenso, considera-se, que as atividades de levantamento de requisitos, modelagem, design e codificação, por exemplo, não são

típicas de um engenheiro de software, embora, muitas vezes, ele seja habilitado a realizá-las. Sua tarefa consiste mais em observar, avaliar, orientar e alterar os processos produtivos quando necessário. Assim, a rigor, as atividades de análise, design, programação e teste de software, e mesmo as atividades de gerenciamento do processo de desenvolvimento, seriam realizadas por profissionais identificados como analistas, designers [...] (WAZLAWICK, 2013, online).

3.1 HTML, CSS, e React Native

A linguagem *HyperText Markup Language* (HTML), traduzida para o português como Linguagem de Marcação de Hipertexto, teve início por volta de 1990, após a criação do protocolo *HyperText Transfer Protocol* (HTTP), ambos criados por Tim Berners Lee. A linguagem define um protocolo de comunicação entre sistemas de informação, que permite a transferência de dados em redes de computadores (TORRES, 2018).

Além de todas estas tags e códigos HTML dentro de uma página, o criador pode escolher optar por formatações mais organizadas e separadas, como é o caso do CSS, formatações mais modernas e também optar por não perder tempo digitando várias linhas de código, como é o caso de Frameworks, que servem para facilitar e agilizar o trabalho com padrões para HTML, CSS e Javascript (TORRES, 2018, p. 99-101).

Com a evolução da programação web surgiu o conceito *Model View Controller* (MVC) como um padrão de arquitetura de software usado nas mais recentes linguagens de programação para desenvolver projetos escaláveis.

No desenvolvimento com React ao invés de utilizar o Javascript puro para a modelagem da interface de usuário, foi utilizado o JSX. De acordo com Pandit (2017), JSX é uma sintaxe que permite a citação em HTML para renderizar subcomponentes. A sintaxe HTML é processada em chamadas JavaScript do React.

Na documentação oficial, *React Native* é uma biblioteca em JavaScript criada pelo *Facebook*. É utilizada para desenvolver aplicativos para os sistemas operacionais de dispositivos móveis, como o *Android* e *IOS*. *React Native* converte o código JavaScript para a linguagem nativa do dispositivo, para Java no caso do *Android* por exemplo, com o intuito de resolver problemas que causavam considerável transtorno na manutenção de elementos que compõem a interface do usuário (KOSTRZEWA (2018); PANDIT (2017)). Assim foi desenvolvida a biblioteca com o objetivo de renderizar páginas HTML tornando aplicativos web mais rápidos, escaláveis e simples. Aplicações em React podem manipular e alterar dados sem a necessidade de carregar páginas.

3.2 Nodejs, ORM e Sequelize para Bancos de Dados PostgreSQL

Node.js é uma plataforma *open source* e *server-side* criada pelo Google, desenvolvida inicialmente em 2009 por Ryan Dahl. É usado principalmente para

desenvolver sistemas com tecnologias PHP, ASP, ASP.NET e Java. Trata-se de uma tecnologia assíncrona que trabalha em uma só *thread* de operação. Assim consegue atender um volume de requisições ao mesmo tempo, não bloqueando os processos a cada requisição como ocorre em outras tecnologias (PULUCENO, 2012).

Mas Node.js é muito mais do que um simples interpretador de Javascript. Ele foi construído utilizando o interpretador de Javascript mais rápido da atualidade. O V8 é o interpretador de Javascript que é utilizado no navegador Chrome, do Google, e foi escrito em C++ pela equipe de desenvolvimento do Google Chrome.

A execução de Javascript no V8 é extremamente rápida e executa muito bem em diversas circunstâncias. O Node.js ajustou o V8 para trabalhar melhor em outros contextos além do navegador, principalmente, fornecendo APIs alternativas que são otimizadas para casos de usos específicos (PULUCENO, 2012, p. 13).

Sequelize é um *framework Object-Relational Mapping* (ORM) para Mapeamento Objeto-Relacional com Node.JS. Ele utiliza o padrão *Promises*, que é uma implementação semântica para tratamento de funções assíncronas. Sua função é conectar, ler, gravar e manipular os dados presentes em um Banco de Dados, como por exemplo MSSQL Server, MySQL, PostgreSQL dentre outros (SEQUELIZE, 2020).

PostgreSQL é um Sistema Gerenciador de Banco de Dados (SGDB) que é o software que cuida do armazenamento, atualização e recuperação de dados computacionais, incluindo os relacionamentos, permitindo toda administração de forma centralizada e segura (POSTGRESQL, 2020).

3.3 Git e Github

Git é um sistema *open source* de controle de versão, inicialmente projetado e desenvolvido por Linus Torvalds em meados de 2005. Sua criação foi focada em desenvolvedores de software, mas pode ser usada para registrar o histórico de edições de qualquer tipo de arquivo. Utilizando o git pode-se manter todo o histórico de alterações no código de projetos e facilmente se retornar para qualquer ponto anterior para saber como o código estava naquela data (GIT, 2020).

Utilizando *branches* (ramificações) do git, que são formas de se ter uma determinada versão do código que sofreu alterações e recebendo *commits* de diferentes fontes e inclusive por diferentes desenvolvedores. Assim, trabalhos em equipe com um número indeterminado de pessoas se tornam mais organizados (GIT, 2020).

Github é uma plataforma de hospedagem de código-fonte que contém um controle de versão utilizando o git. Foi desenvolvido por Chris Wanstrath, J. Hyett, Tom Preston-Werner e Scott Chacon usando as linguagens Ruby e Rails, em fevereiro de 2008 (GITHUB, 2020).

4 Desenvolvimento do Projeto e Implementação do Aplicativo

4.1 Levantamento e Documentação de Requisitos

O levantamento de requisitos foi realizado por meio de coleta de dados com formulários eletrônicos respondidos por alunos e professores, bem como por entrevistas realizadas com responsáveis pelas áreas acadêmicas da instituição de ensino objeto do estudo de caso. Na etapa de análise e projeto foram desenvolvidos diagramas com a Linguagem de Modelagem Unificada (UML do inglês *Unified Modeling Language*), como os de Casos de Uso, de Classes e de Atividades, e realizada a modelagem dos processos de negócio com a ferramenta de Notação de Modelo de Processos de Negócios (BPMN do inglês *Business Process Model and Notation*). Para a fase de codificação foram utilizadas as ferramentas Node.js, React Native, Adobe XD para prototipação de interface e o gerenciador de banco de dados PostgreSQL.

A etapa de testes foi realizada em paralelo à etapa de codificação, sendo utilizados os testes unitários, de integração e de sistema. Foram utilizadas técnicas de conversação, observação, *brainstorming* e entrevistas, para que fosse possível analisar qual a melhor forma de oferecer valor aos alunos, seja com uma ferramenta de automatização de processos de contato ou de simplificação de processos. No Quadro 1 são apresentados os requisitos e os responsáveis por suas implementações.

4.2 Análise e Projeto: diagramas UML e BPMN

Existem diversas abordagens para a modelagem de processos de negócios, ou seja, formas de representação dos conceitos do modelo de negócio onde o objetivo principal é fornecer uma especificação do processo de modelagem de regras de negócio que seja compreensível pelos envolvidos desde analistas de negócio que criam os desenhos iniciais dos processos, passando pelos desenvolvedores que são os técnicos responsáveis pela implementação da automação que executará os processos, e finalmente para os responsáveis pela gestão do negócio (WHITE, 2004).

Para tais atividades de análise e projeto foram usadas as ferramentas UML e BPMN que criam uma ponte padronizada entre o processo de negócio e sua implementação. As Figuras 2 e 3 apresentam os diagramas criados.

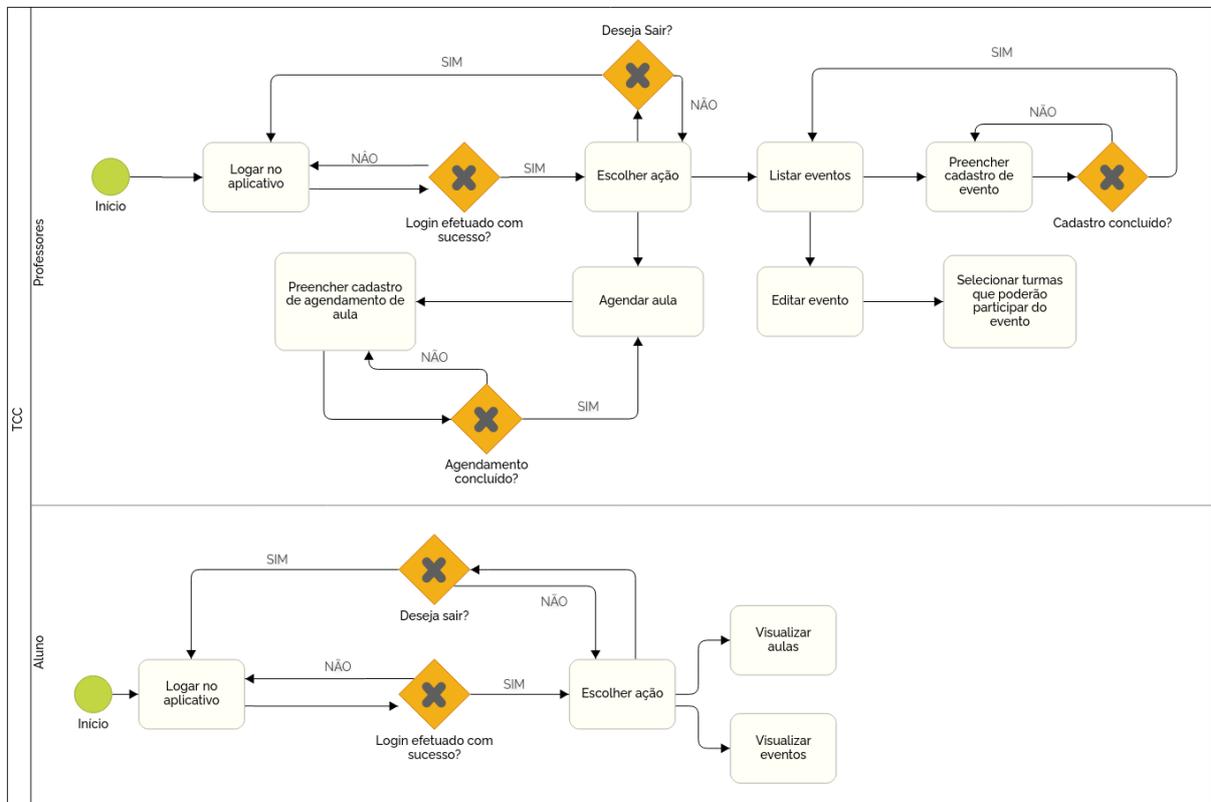
Quadro 1 - Requisitos e responsabilidades

Requisitos	Descrição	Responsável
01	Disponibilização dos horário das aulas dos professores	Cícero
02	Priorização de agendamento por parte de determinadas áreas “próprias” daquele laboratório	Cícero
03	Alertar o aluno cadastrado para quando houver notificações	Cícero
04	Gravação de LOG de usuário para todas as movimentações de mudança de	Donato

	status quando for cadastrado uma aula ou evento	
05	Alertar professores cadastrados quando houver laboratórios vagos ou ocupados	Cícero
06	Alertar professores cadastrados para cadastrar suas aulas diárias	Donato
07	Gravação no histórico das aulas cadastradas pelos professores para documentação	Cícero

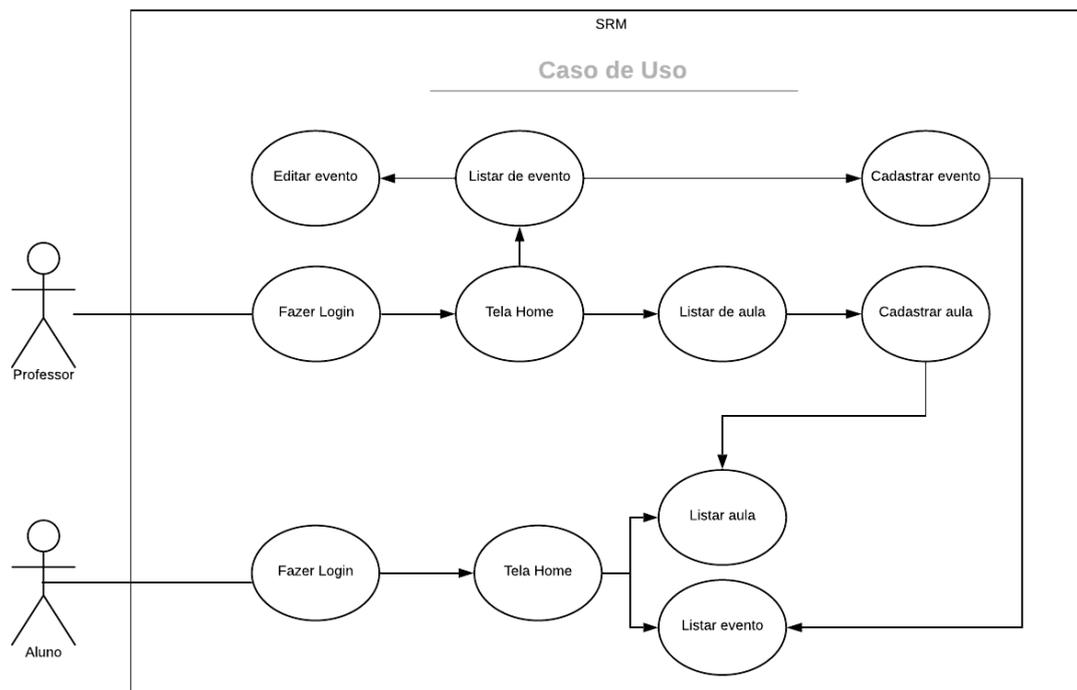
Fonte: os autores

Figura 2: BPMN



Fonte: os autores

Figura 3: Casos de Uso



Fonte: os autores

4.3 Diagrama de Classes

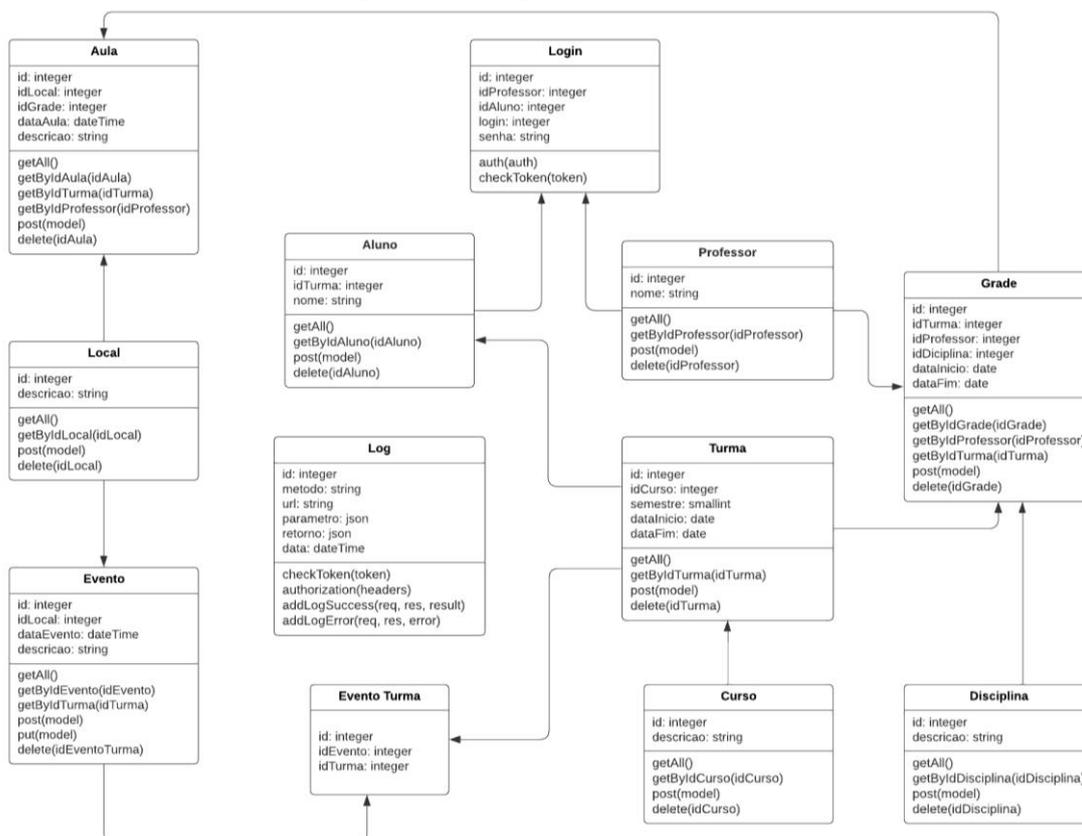
Para Silveira e Schmitz (1999) o Diagrama de Classes (DC), parte central da programação orientada a objetos, descreve os tipos de objetos do sistema e os relacionamentos estáticos que ocorrem entre eles. O DC é estático pois a estrutura descrita é sempre válida em qualquer instante do ciclo de vida de um projeto. Ele mostra os atributos e as operações de uma classe e as restrições associadas a ela. Na programação orientada a objetos, classes, objetos e seus relacionamentos são vistos como os elementos primários da modelagem. A Figura 4 mostra o DC do projeto.

4.4 Modelagem do Banco de Dados

Perkusich (1999, p. 253-267) afirma que:

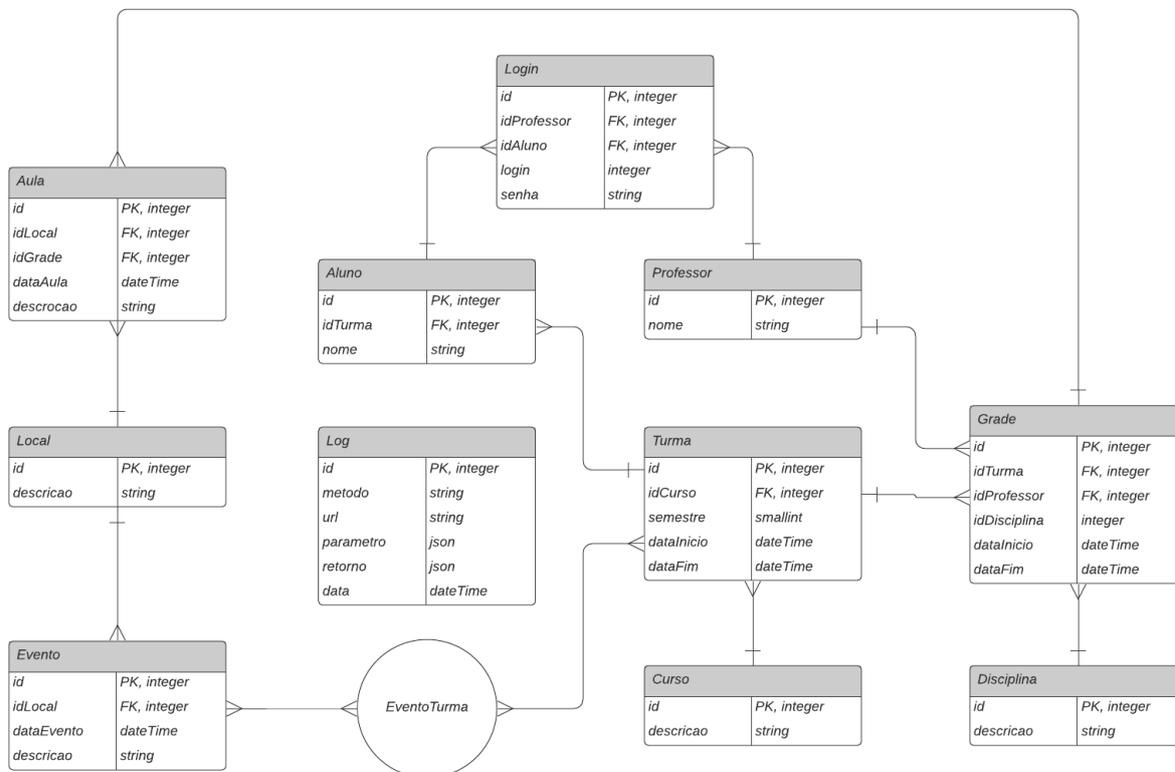
No contexto de sistemas de banco de dados, principalmente no caso de grandes sistemas, a integração é uma capacidade crítica e vital. Dados de diferentes aplicações devem ser integrados com o objetivo de possibilitar a coordenação de aplicações individuais em consonância com os objetivos corporativos.

Figura 4: Diagrama de Classes



Fonte: os autores

Desta forma, para satisfazer o requisito de integração no contexto de aplicações com grandes volumes de dados e restrições temporais, e a necessidade da utilização de ferramentas formais no desenvolvimento de sistemas complexos, é usada a modelagem de banco de dados, contemplando o requisito de integração e a necessidade de embasamento formal. A Figura 5 mostra o diagrama de modelagem do banco de dados do projeto.

Figura 5: Modelagem do Banco de Dados


Fonte: os autores

4.5 Prototipação das Telas

Conforme Pressman (2016) o usuário nem sempre sabe expressar qual sua necessidade, sendo assim o analista se depara com um problema: definir quais são os requisitos a serem atendidos no desenvolvimento do sistema. Uma técnica apropriada nesse caso é a prototipação, na qual protótipos de telas da interface são criados contendo as possíveis entradas e saídas do sistema. O processo de prototipação envolve atividades que devem ser realizadas para que se tenha sucesso na definição das funcionalidades que devem ser implementadas. São elas:

- A. **Comunicação:** o engenheiro de software e o cliente encontram-se e definem os objetivos gerais do software, identificam as necessidades conhecidas e delineiam áreas que necessitam de mais definições.
- B. **Plano Modelagem e Projeto Rápido:** uma iteração de prototipagem é planejada rapidamente e a modelagem (na forma de projeto rápido) ocorre.
- C. **Construção:** o projeto rápido leva à construção de um protótipo. Para elaborar o protótipo, o desenvolvedor tenta usar partes de programas existentes ou aplicar ferramentas (por exemplo, geradores de relatórios, gestores de janelas, etc.) que possibilitam programas executáveis serem gerados rapidamente.

D. **Implantação, Entrega e Feedback:** o protótipo é implantado e depois avaliado pelo cliente/usuário que fornece o *feedback* que é usado para refinar os requisitos e a implementação.

A interação destas atividades ocorre à medida que o protótipo é ajustado para satisfazer às necessidades do cliente, e, ao mesmo tempo, permite ao desenvolvedor entender melhor o que precisa ser feito. As Figuras 6 a 23 apresentam protótipos de alta resolução das telas da interface do aplicativo que foram criados durante o projeto.

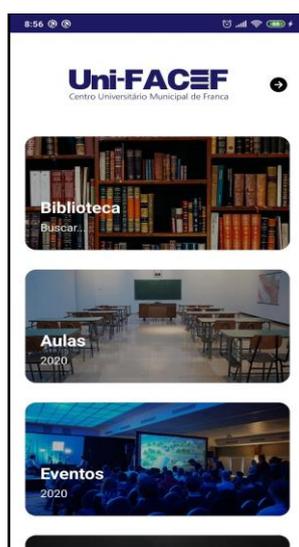
Na Figura 6 temos a tela de *login* do aplicativo, nela é pedido os dados já cadastrados do aluno ou professor. Na figura 7 temos a tela *home* do aluno, nela tem as principais funcionalidades que o aluno pode acessar. Na figura 8 temos a tela de aulas, que é onde o aluno visualiza as aulas cadastradas pelos professores.

Figura 6: Login



Fonte: os autores

Figura 7: Aluno - Tela Home



Fonte: os autores

Figura 8: Aluno - Aulas



Fonte: os autores

Figura 9 é a mesma tela da Figura 8 mostrando detalhes das aulas, como local, data e descrição da aula. Figura 10 é a tela de eventos, que é onde o aluno visualiza as aulas cadastradas pelos coordenadores e/ou professores. Figura 11 é a mesma tela da figura 10 mostrando os detalhes dos eventos, como local e data.

Figura 9: Aluno - Detalhe Aula

Fonte: os autores

Figura 10: Aluno - Lista de Eventos

Fonte: os autores

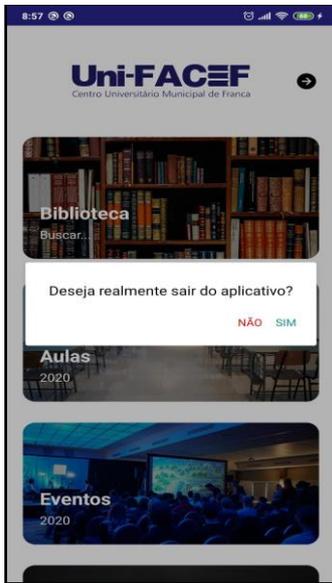
Figura 11: Aluno - Detalhe de Evento

Fonte: os autores

A Figura 12 demonstra a caixa de diálogo de confirmação de logout do aplicativo de acordo com a área do aluno. Na Figura 13 é apresentado a Tela Home da área do professor, onde ele pode escolher uma das opções clicando nas caixas. Já na Figura 14 podemos ver a Tela de Listagem de Aulas que é acessada pelos professores.

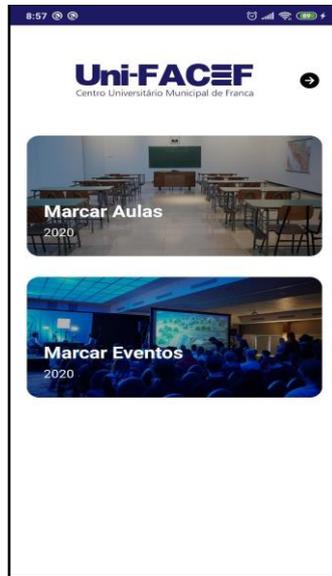
Na Figura 15 vemos a caixa de cadastro das aulas, que aparece ao clicar no botão “Adicionar Aula” da Figura 14, nessa caixa é solicitado a turma, local, data e descrição da aula. A Figura 16 é a mesma da Figura 14, porém mostrando os detalhes de uma aula. Já na Figura 17 é demonstrado a caixa de confirmação de exclusão de uma aula, que aparece ao clicar na descrição da aula.

Figura 12: Aluno - Sair



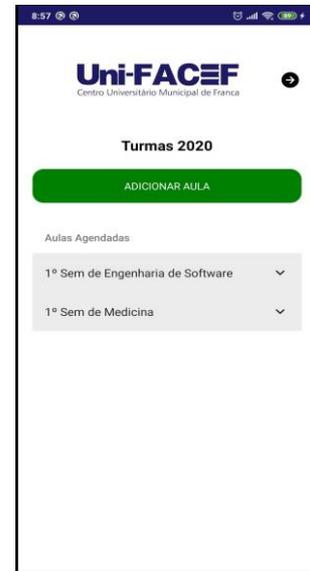
Fonte: os autores

Figura 13: Professor - Tela Home



Fonte: os autores

Figura 14: Professor - Lista de Aulas



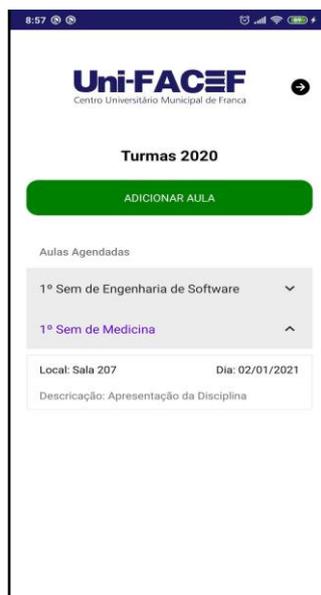
Fonte: os autores

Figura 15: Professor - Inclusão de Aula



Fonte: os autores

Figura 16: Professor - Detalhes de Aula



Fonte: os autores

Figura 17: Professor - Exclusão de Aula



Fonte: os autores

Na Figura 18 podemos ver a Tela de Listagem de Eventos que é acessada pelos professores. Na Figura 19 vemos a caixa de cadastro dos eventos, que aparece ao clicar no botão “Adicionar Evento” da Figura 18, nesta caixa é solicitado a descrição, local e data do evento. Já a Figura 20 é a mesma da Figura 18, porém mostrando os detalhes de um evento.

Figura 18: Professor - Lista de Eventos



Fonte: os autores

Figura 19: Professor - Inclusão de Eventos



Fonte: os autores

Figura 20: Professor - Detalhes do Evento



Fonte: os autores

Na Figura 21 é demonstrada a tela de inclusão de turmas no evento, que aparece ao clicar na descrição de um evento. Já na Figura 22 é apresentada a caixa de confirmação de inclusão de uma turma ao evento, que aparece ao clicar no botão “Adicionar Turma” da Figura 21. Na Figura 23 é demonstrado a caixa de confirmação de exclusão de uma turma do evento, aparece ao clicar na turma listada na Figura 21.

Figura 21: Professor - Edição do Evento



Fonte: os autores

Figura 22: Professor - Inclusão de Turma em Evento



Fonte: os autores

Figura 23: Professor - Exclusão de Turma do Evento



Fonte: os autores

A Figura 24 demonstra a caixa de diálogo de confirmação de logout do aplicativo de acordo com a área do professor.

Figura 24: Professor - Sair



Fonte: os autores

5 Considerações

Dentre as inúmeras necessidades de melhorar em instituições de ensino a comunicação com os alunos, professores e funcionários, destaca-se a necessidade de centralizar em um único canal os horários e salas das aulas, das provas e dos eventos

escolares. Isto posto, foi proposto o projeto e a implementação de um aplicativo para facilitar a comunicação entre esses atores, e proporcionar melhor experiência ao aluno durante seu período na IES com mais clareza e facilidade na disponibilização de informações acadêmicas.

Para o desenvolvimento do aplicativo foram utilizadas ferramentas e metodologias propostas pela Engenharia de Software como os diagramas BPMN e de Caso de Uso, bem como técnicas de levantamento e documentação de requisitos, testes e segurança de software, no intuito de facilitar as etapas do ciclo de vida de desenvolvimento de software. Dessa maneira foi possível aplicar os conceitos e padrões vistos nas disciplinas do curso para melhorar a qualidade dos processos de coleta de dados, análise e projeto do protótipo, previsão de eventuais falhas e suas correções, mapeamento de testes, e implementação de padrões de segurança do software.

Segundo análise realizada dos dados coletados com alunos, constatou-se a expectativa de que com o aplicativo a comunicação e o acesso às informações seria mais fácil e ágil, melhorando o relacionamento entre os membros da instituição.

No âmbito de enriquecimento profissional dos autores, destacou-se a ampliação de conhecimentos teóricos e práticos acerca dos sistemas de informação e dos processos do ciclo de vida do desenvolvimento de software, bem como a satisfação no atingimento dos objetivos estabelecidos para o projeto de TCC.

Referências

COLLIS, D.; RUKSTAD, M. Você sabe dizer qual é sua estratégia. *Harvard Business Review*, p. 40-99, 2008.

GHOLAMI, Hamed et al. A CRM strategic leadership towards sustainable development in student relationship management: SD in higher education. *Procedia Manufacturing*, v. 2, p. 51-60, 2015.

GIT. *About*. 2020. Disponível em: <<https://git-scm.com/about>>. Acesso em: 29 out.2020.

GITHUB. *About*. 2020. Disponível em: <<https://github.com/about>>. Acesso em: 29 out. 2020.

GOMES, Mariza; FÁVERO, Natália Colantonio; LUCAS, Carlos Alberto. Gerenciamento do Relacionamento com o Estudante no ensino superior. *Revista Eletrônica de Sistemas de Informação e de Gestão Tecnológica*, v. 4, n. 1, 2015.

KOSTRZEWA, Denis. *Is React.js the Best JavaScript Framework in 2018?*. 2018. Disponível em: <<https://hackernoon.com/is-react-js-the-best-javascript-framework-in-2018-264a0eb373c8>>. Acesso em: 09 mai.2020.

MENDONÇA, D. A.; TERRAZ, L. A. A. Vantagens de Implementação do CRM na Indústria: uma análise multicase. 2017. Disponível em: <https://www.researchgate.net/publication/317643666_Vantagens_da_implementacao_do_CRM_na_industria_uma_analise_multicase>. Acesso em: 9 mai.2020.

PANDIT, Nitin. *What Is ReactJS and Why Should We Use It?*. 2017. Disponível em: <<https://www.c-sharpcorner.com/article/what-and-why-reactjs>>. Acesso em: 10 mai.2020.

PERKUSICH, Maria Ligia B.; DE FQV TURNELL, Maria; PERKUSICH, Angelo. Modelagem de Bancos de Dados em Tempo-real. In: *SBBD*. 1999. p. 253-267.

POSTGRESQL. *About*. 2020. Disponível em: <<https://www.postgresql.org/about/>>. Acesso em: 29 out.2020.

PRESSMAN, Roger; MAXIM, Bruce. *Engenharia de Software-8ª Edição*. McGraw Hill Brasil, p. 14, 2016.

PULUCENO, Thiago Vieira. *Estudo de caso sobre uma API REST em Node.js*. 2012. p. 13.

SEQUELIZE. *Api Reference*. 2020. Disponível em: <<https://sequelize.org/master/>>. Acesso em: 29 out.2020.

SILVEIRA, D.; SCHMITZ, Eber A. *FAST CASE-Uma Ferramenta CASE para o Desenvolvimento Visual de Sistemas Orientados a Objetos*. 1999. PhD Thesis. tese de mestrado, IM/NCE/UFRJ.

TORRES, Victor Monteiro. HTML e seus Componentes. *Revista Ada Lovelace*, 2018, 2: 99-101.

WAZLAWICK, Raul. *Engenharia de Software - Conceitos e Práticas*. Rio de Janeiro: Grupo GEN, 2013. 9788595156173. Disponível em: <<https://integrada.minhabiblioteca.com.br/#/books/9788595156173/>>. Acesso em: 16 set.2020.

WHITE, Stephen A. *Introduction to BPMN*, 2004. *IBM Corporation*. Disponível em: <<http://www.bptrends.com>>. Acesso em: 10 mai.2020.