

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

FELIPE BERTOLDO COLOMBO DE SOUZA

Gaveta
Um espaço de aprendizado dentro de seu bolso

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Érika Fernandes Cota

Porto Alegre
2020

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Reitor: Prof. Carlos André Bulhões Mendes

Vice-Reitora: Prof^a. Patricia Helena Lucas Pranke

Pró-Reitoria de Ensino (Graduação e Pós-Graduação): Prof^a Cíntia Inês Boll

Diretora do Instituto de Informática: Prof. Carla Maria Dal Sasso Freitas

Coordenador do Curso de Ciência da Computação: Prof. Sérgio Luis Cechin

Bibliotecária-chefe do Instituto de Informática: Beatriz Regina Bastos Haro

AGRADECIMENTOS

Agradeço à minha orientadora, Érika, por ter aceitado me orientar nesse trabalho. Agradeço aos professores do Instituto de Informática da UFRGS que, diariamente, além de transmitirem seus conhecimentos com excelência, nutrem a motivação de seguir adiante nos alunos. Agradeço à minha família que sempre esteve ao meu lado, mesmo com quilômetros de distância física, e também aos meus amigos, que formaram uma segunda família aqui em Porto Alegre.

RESUMO

Durante a graduação, muitas vezes os alunos buscam materiais além dos fornecidos pelas instituições de ensino, como artigos, listas de exercícios e até mesmo anotações de colegas. Com a popularização das redes sociais e dos mensageiros instantâneos, essas tecnologias tornam-se o meio de discussão e compartilhamento de conhecimento entre os alunos. No entanto, essa escolha pode nem sempre ser organizada e de fácil acesso: na prática, os alunos discutem sobre os mesmos assuntos nas redes sociais e nos mensageiros, causando uma descentralização nas informações e também, muitas vezes, replicação.

O objetivo desse trabalho é propor uma solução para essa descentralização, através do Gaveta, um produto que visa promover um ambiente focado nos estudos e na comunidade acadêmica. Para o desenvolvimento da solução, foram utilizados critérios de Engenharia de Software com o objetivo de organizar e otimizar o processo.

Palavras-chave: Social. Engenharia de Software. Compartilhamento de arquivos.

Gaveta: A learning space in your pocket

ABSTRACT

Students often look for materials in addition to those provided by educational institutions, such as articles, exercise lists and even notes made by colleagues. With the popularization of social networks and instant messengers, these technologies become the environment for discussion and knowledge sharing among students. However, this choice may not always be organized and easily accessible: in practice, students discuss the same subjects on social networks and messengers, causing a decentralization of information and also, often, replication.

The objective of this work is to propose a solution to this decentralization, through Gaveta, a product that aims to promote an environment focused on studies and the academic community. For the development of the solution, Software Engineering criteria were used in order to organize and optimize the process.

Keywords: Social. Software Engineering. File sharing.

LISTA DE FIGURAS

Figura 2.1.1 - Tela do Google Drive.....	11
Figura 2.1.2 - Tela do Dropbox.....	12
Figura 2.2 - Tela do Passei Direto.....	13
Figura 2.3 - Tela de um grupo da rede social Facebook.....	13
Figura 3.1.1 - Diagrama de um MVC.....	15
Figura 3.1.2 - MVC da Apple.....	16
Figura 3.1.3 - Padrão MVVM.....	17
Figura 3.1.4 - Exemplo de fluxo de telas com Coordenadores.....	18
Figura 4.2 - Exemplo de esboço da aplicação.....	22
Figura 4.3.1 - Exemplo de requerimento de dados.....	23
Figura 4.3.3 - Implementação de repositório de dados.....	24
Figura 4.3.4 - Camadas da aplicação.....	25
Figura 4.4.1 - Entidade de Instituição.....	26
Figura 4.4.2 - Entidade de usuário.....	27
Figura 4.4.3 - Entidade de disciplina.....	27
Figura 4.4.4 - Entidade de postagem.....	28
Figura 4.4.5 - Entidade de comentário.....	29
Figura 4.5 - Relacionamento entre as entidades do sistema.....	30
Figura 5.1.2 - Tela do Adobe XD.....	32
Figura 5.1.3 - Tela do Xcode com código em Swift (esq) e Simulador de iOS (dir).....	33
Figura 5.1.4 - Painel do Cloud Firestore.....	34
Figura 5.3 - Apresentação do Gaveta.....	36
Figura 5.3.1 - Cena de sugestão automática de nome de usuário.....	40
Figura 5.4 - Contratação do Gaveta.....	47
Figura 6.1 - Resultados da avaliação do fluxo de cadastro.....	48
Figura 6.2 - Resultados da avaliação do fluxo de seguir nova matéria.....	49
Figura 6.3 - Resultados da avaliação do fluxo de criar nova postagem.....	50
Figura 6.4 - Resultados da avaliação do fluxo de criar novo comentário.....	51
Figura 6.5 - Resultados da avaliação o fluxo de deixar de seguir matéria.....	52
Figura B1 - Formulário de avaliação.....	60

LISTA DE TABELAS

Tabela 4.1 - Cadastro de usuário	20
Tabela 5.2 - Etapas do projeto	34
Tabela A1 - Login de Usuário	55
Tabela A2 - Seguir disciplinas.....	55
Tabela A3 - Favoritar disciplinas.....	55
Tabela A4 - Visualizar lista de postagens de uma disciplina.....	56
Tabela A5 - Criar nova postagem em uma disciplina.....	56
Tabela A6 - Visualizar postagem de uma disciplina.....	57
Tabela A7 - Criar novo comentário em uma postagem de uma disciplina.....	57
Tabela A8 - Deixar de seguir uma disciplina	58
Tabela A9 - Remover uma disciplina da lista de favoritos.....	58

LISTA DE ABREVIATURAS E SIGLAS

MVC	Model View Controller
MVVM	Model - View - ViewModel
SDK	Software Development Kit
macOS	Mac Operation System
iOS	iPhone Operation System
MVP	Mínimo Produto Viável
NoSQL	Not Only Structured Query Language

SUMÁRIO

1 INTRODUÇÃO E MOTIVAÇÃO	9
2 SOLUÇÕES RELACIONADAS	11
2.1 Hospedagem de arquivos na nuvem	11
2.2 Passei Direto	12
2.3 Grupos em Redes Sociais.....	13
2.4 Stack Exchange e Stack Overflow	14
2.5 Mensageiros instantâneos.....	14
2.6 Considerações	14
3 FUNDAMENTAÇÃO TEÓRICA	15
3.1 Arquiteturas e padrões de desenvolvimento	15
3.2 Metodologia de trabalho.....	18
4 PROPOSTA DE ARQUITETURA	20
4.1 Funcionalidades.....	20
4.2 Esboço da aplicação	21
4.3 Camadas do sistema.....	22
4.3.1 Camada de Domínio	22
4.3.2 Camada de Apresentação	24
4.3.3 Camada de Dados	24
4.4 Entidades do sistema.....	25
4.4.1 Instituição.....	25
4.4.2 Usuário.....	26
4.4.3 Disciplina	27
4.4.4 Postagem.....	27
4.4.5 Comentário.....	28
4.5 Relacionamento entre as entidades.....	29
5 IMPLEMENTAÇÃO	31
5.1 Ferramentas Utilizadas.....	31
5.1.1 Git e GitHub.....	31
5.1.2 Adobe XD	31
5.1.3 Xcode e Swift.....	32
5.1.4 Firebase.....	33
5.2 Etapas do desenvolvimento	34
5.3 Experiência da aplicação	36
5.3.2 Cena de inserção de dados	37
5.3.3 Cena de inserção de foto de perfil	38
5.3.4 Cena de sugestão automática de nome de usuário	39
5.3.5 Cena inicial	40
5.3.6 Cena de seguir disciplinas.....	42
5.3.7 Cena de disciplina	43
5.3.8 Cena de postagem	44
5.3.9 Cenários de erro	45
5.4 Modelo de negócio	46
6 AVALIAÇÃO	48
6.1 Avaliação do fluxo de cadastro	48
6.2 Avaliação do fluxo de seguir nova matéria.....	49
6.3 Avaliação do fluxo de criar nova postagem.....	49
6.4 Avaliação do fluxo de criar novo comentário em postagem	50
6.5 Avaliação do fluxo de deixar de seguir matéria	51
6.6 Considerações das avaliações	52
7 CONCLUSÃO	53
REFERÊNCIAS	54
ANEXO A - HISTÓRIAS DE USUÁRIO	55
ANEXO B - FORMULÁRIO DE AVALIAÇÃO	60

1 INTRODUÇÃO E MOTIVAÇÃO

Com a popularização cada vez mais crescente das redes sociais, as pessoas fazem uso dessas plataformas para os mais diversos fins, como vendas, compartilhamento de propagandas, promoções, e até mesmo discussões do dia-a-dia, referentes ao trabalho, escola ou faculdade. No contexto acadêmico, vários grupos são criados em páginas na internet para que os alunos possam trocar informações de conhecimento, sendo elas desde listas de exercícios até mesmo anotações de aulas e arquivos pessoais.

No entanto, essas páginas são criadas, muitas vezes, pelos próprios usuários, o que pode ser problemático em alguns aspectos. Por ser algo descentralizado, é possível encontrar várias páginas com o mesmo assunto - e muitas vezes frequentada por grupos iguais de pessoas - o que pode contribuir para a desorganização das interações, uma vez que o mesmo tópico pode ser discutido em dois lugares diferentes. Isso pode ser prejudicial para as discussões, que poderiam render mais se todos estivessem discutindo no mesmo espaço.

Além disso, essa descentralização pode levar a uma replicação desnecessária de informações. Para melhor ilustrar, um exemplo: foi proposto para um aluno A do curso de Cálculo um exercício de estudo de difícil resolução, sobre o qual ele gostaria de debater com seus colegas. Ao perguntar para um colega B, este o forneceu um *link* com a resolução desse exercício feita por outro integrante da turma. Ao perguntar para um segundo colega, C, este também o forneceu um *link*, diferente do primeiro, com a mesma intenção. O aluno A se sentiu tranquilo pois teria duas fontes diferentes de referência para poder estudar mais tarde mas, para sua surpresa, apesar de os links serem de lugares diferentes, ambos levavam para o mesmo arquivo, apenas renomeado.

Outra questão sobre as páginas serem criadas pelos próprios usuários é sobre a sua manutenção. Se forem páginas que requerem administração de quem as criou, elas correm o risco de deixarem de ser alimentadas à medida que o usuário fique indisponível ou, no caso acadêmico, termine o curso. Além disso, muitas pessoas optam por não participarem de redes sociais por diversas razões, e não querem se ver obrigadas a participar de redes grandes para usar apenas a funcionalidade dos grupos.

Considerando essas questões, esse trabalho propõe um produto também social, mas específico e focado diretamente no meio acadêmico, promovendo um ambiente organizado que compreenda o contexto dos estudantes. O Gaveta é uma aplicação que visa fornecer ao aluno ambientes de discussão específicos para seu curso, em sua instituição de ensino, de modo a centralizar as discussões e facilitar a descoberta de informações.

Levando em consideração o grande número de usuários *mobile*, e buscando atingir a praticidade de acessar algo diretamente de um aplicativo, o Gaveta é desenvolvido como uma solução *mobile-first* e disponibilizado na plataforma iOS. O título Gaveta é pensado para lembrar de uma gaveta de verdade, como em uma escrivaninha. O espaço onde o estudante guarda o que é importante pra si, e de onde busca o que é importante pra si. Precisou de algo? Procura na Gaveta!

O texto está organizado da seguinte maneira: o Capítulo 2 mostra as soluções análogas ao Gaveta para compartilhamento de arquivos, mostrando as saídas que os estudantes têm encontrado para se comunicarem no ambiente acadêmico. O Capítulo 3 mostra a

fundamentação teórica que serviu de base para o desenvolvimento da aplicação, desde em termos técnicos de arquitetura quanto em termos organizacionais de metodologia de trabalho. O Capítulo 4 mostra a proposta de projeto, ou seja, a aplicação da fundamentação teórica do Capítulo 3 na problemática que motivou esse trabalho, mostrando as funcionalidades, camadas e entidades do sistema. O Capítulo 5 diz como foi a implementação do Gaveta, mostrando as ferramentas utilizadas, a divisão do trabalho em etapas, bem como como ficaram as telas da aplicação. O Capítulo 6 traz a avaliação com usuários, buscando medir o quanto satisfeitos ficaram os usuários ao usarem o Gaveta. O Capítulo 7 apresenta a conclusão e também os pontos de melhoria e trabalhos futuros a serem realizados.

2 SOLUÇÕES RELACIONADAS

Visando resolver os problemas discutidos no Capítulo 1, os alunos já lançam mão de algumas ferramentas para compartilhar conhecimento e arquivos. A seguir, serão discutidas três soluções tipicamente utilizadas no contexto acadêmico.

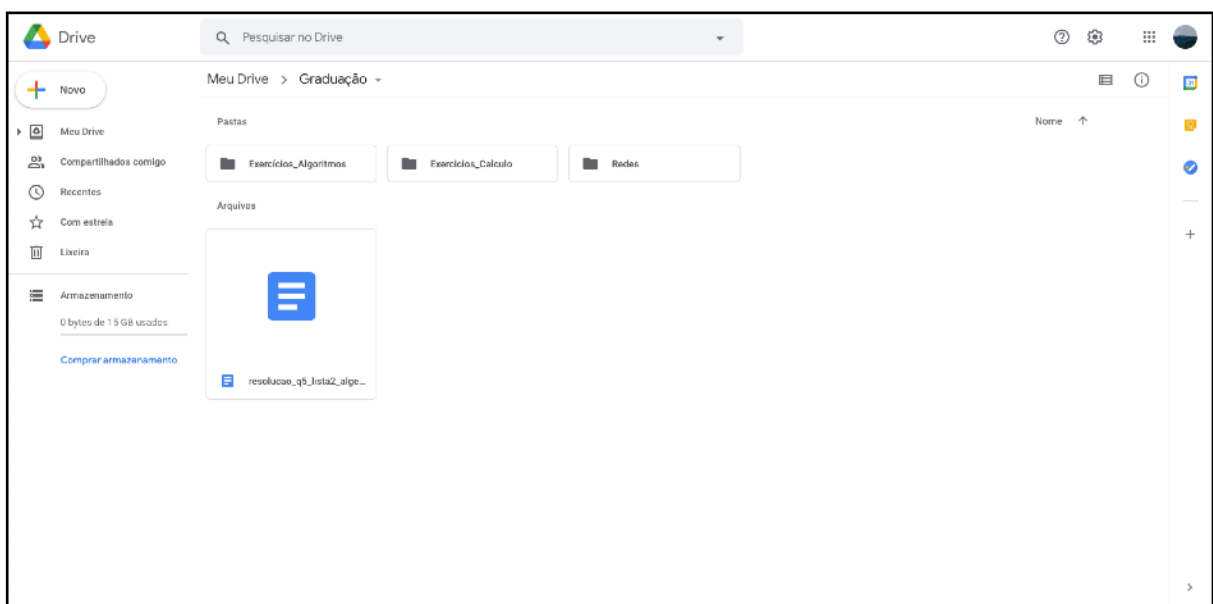
2.1 Hospedagem de arquivos na nuvem

Uma solução bastante usada para o compartilhamento de arquivos entre os alunos é o armazenamento na nuvem, como o Google Drive e o Dropbox. Essas plataformas possibilitam a criação de pastas, como ilustram as Figuras 2.2.1 e 2.2.2, que podem ser compartilhadas entre usuários, que podem apenas visualizar o conteúdo ou também contribuir com arquivos.

Essas pastas são acessadas, muitas vezes, por membros de uma mesma comunidade acadêmica, o que facilita a assertividade quando o usuário estiver buscando algum tópico específico do seu contexto. No entanto, a organização é criada coletivamente, o que pode levar à replicação de arquivos, por exemplo no seguinte cenário: um aluno A compartilha, em sua pasta, a uma lista de exercícios de Cálculo, através de um arquivo chamado “lista_01.pdf”; outro aluno B compartilha a mesma lista, mas em sua pasta e através de um arquivo chamado “exercicios_calculo.pdf”. Assim, existem dois arquivos com o mesmo conteúdo, porém em locais diferentes e sendo potencialmente acessados pelos mesmos alunos.

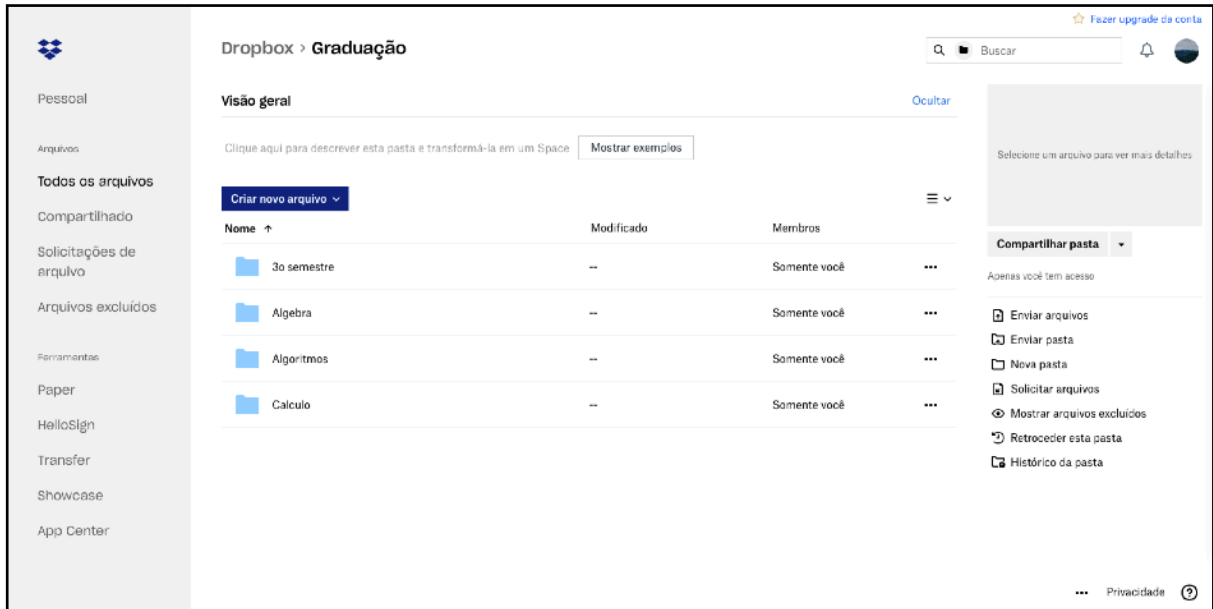
Nesse sentido, o Gaveta propõe reduzir o número de duplicações, através de um ambiente centralizado e organizado de acordo com cada curso.

Figura 2.1.1 - Tela do Google Drive



Fonte: o autor

Figura 2.1.2 - Tela do Dropbox

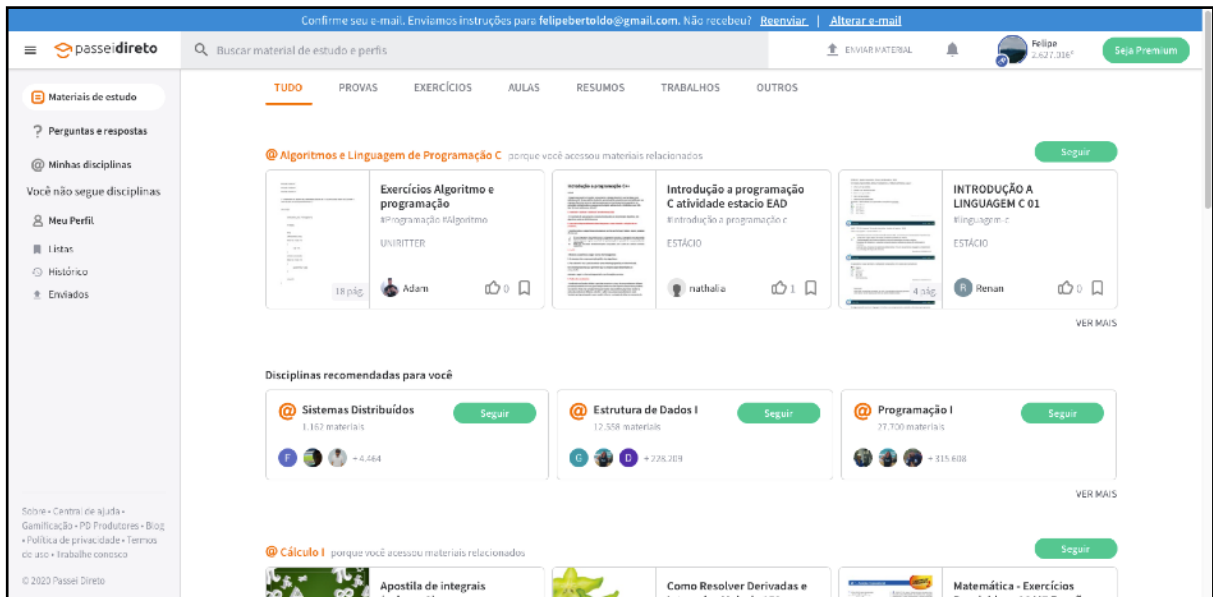


Fonte: o autor

2.2 Passei Direto

O Passei Direto é uma rede social focada no meio acadêmico, que aborda diversas disciplinas. Nela, os estudantes podem compartilhar materiais de diversos tópicos, com abrangência nacional. Diferente do que será proposto nesse trabalho, as disciplinas podem ser acessadas por alunos de diversas universidades, o que é bom por aumentar as possibilidades, mas pode gerar um contexto muito mais amplo do que o estudante está inserido e necessita. A Figura 2.2 mostra a tela inicial do Passei Direto.

Figura 2.2 - Tela do Passei Direto



Fonte: o autor

2.3 Grupos em Redes Sociais

Outra alternativa de compartilhamento de material e de discussão é através de grupos em redes sociais, como o Facebook, mostrado na Figura 2.3. Esses grupos podem ser tanto referentes a uma disciplina, ou então a um curso, geralmente englobando discussões diversas, não apenas relacionadas aos assuntos das matérias. Como são parte de uma rede social maior, de propósito geral, pode haver também replicação nesses grupos, por exemplo um grupo chamado “Alunos de Cálculo I” e outro “Alunos de Cálculo 1”, se referindo à mesma matéria.

Figura 2.3 - Tela de um grupo da rede social Facebook



Fonte: o autor

2.4 Stack Exchange e Stack Overflow

O *Stack Exchange* é uma rede de sites de perguntas e respostas sobre tópicos em diversos campos, cada site cobrindo um tópico específico. (EXCHANGE, 2020). Um de seus principais sites é o *Stack Overflow*, com foco em questões relativas à área de desenvolvimento de *software*. Essa solução também é bastante usada pelos estudantes, no entanto, muitas vezes para pesquisas de cunho mais prático, até mesmo em nível de código. Além disso, embora os assuntos dentro da plataforma sejam definidos por *tags*, eles são bem gerais, não pertencendo a uma noção de matérias do contexto acadêmico.

2.5 Mensageiros instantâneos

Outra alternativa para compartilhamento de informações e arquivos são grupos em mensageiros instantâneos, como *WhatsApp*, *Discord*, *Telegram* e *Slack*. No entanto, essas soluções se assemelham bastante à solução discutida na Seção 2.3, já que essas plataformas são de uso genérico e possuem a funcionalidade de criação de grupos, dos quais a semântica e a manutenção são dadas pelos próprios autores.

2.6 Considerações

Apesar de o Gaveta permitir o intercâmbio de arquivos e as soluções relacionadas serem, em sua maioria, focadas em armazenamento e compartilhamento de arquivos, o foco desse trabalho é fomentar um ambiente social focado em discussões acadêmicas mais técnicas. A função de compartilhar arquivos em si é apenas um facilitador para a usabilidade da aplicação, mas não seu foco principal.

Nessa proposta, pretendemos adaptar as interações comuns em redes sociais em um espaço que permite disseminação de conhecimento em um nível mais local, ou seja, como se os alunos estivessem na companhia de seus colegas de sala resolvendo problemas do dia-a-dia. Para isso, é importante manter a organização dos espaços no sistema: novos tópicos de discussões serão possíveis apenas dentro do contexto de determinada disciplina, estimulando ainda mais que o assunto abordado tecnicamente esteja dentro do contexto adequado.

De acordo com um relatório divulgado pela associação *GSMA* em 2017, o Brasil é o país com mais smartphones conectados à internet na América Latina. (AMPUDIA, 2020). Considerando esses dados, assumimos que o sistema também possuirá um grande, se não majoritário, público *mobile*, nos motivando a conceber uma aplicação primeiramente móvel.

A escolha pela plataforma iOS se dá pela preocupação com a experiência do usuário e facilidade no desenvolvimento. A Apple fornece aos desenvolvedores um guia completo de diretrizes de *design*, o que agiliza bastante o momento da concepção das interfaces de usuário do aplicativo, uma vez que seguir esse padrão exclui a necessidade de criar regras próprias de usabilidade e interação. Além disso, as aplicações iOS são executadas em um menor número de tipos de dispositivos, o que é um facilitador na preocupação de desenvolver para diferentes tamanhos de tela.

3 FUNDAMENTAÇÃO TEÓRICA

No momento da concepção do sistema, além de definir as principais funcionalidades, é preciso definir um processo de desenvolvimento que guiará todas as etapas que serão realizadas, visando alcançar um produto consistente, fácil de manter e fácil de evoluir no futuro. Para que atinjamos esse objetivo, são imprescindíveis alguns conceitos de Engenharia de Software, que oferecem a base necessária que será aplicada ao contexto da aplicação.

Os principais conceitos de Engenharia de Software usados nesse projeto serão discutidos a seguir.

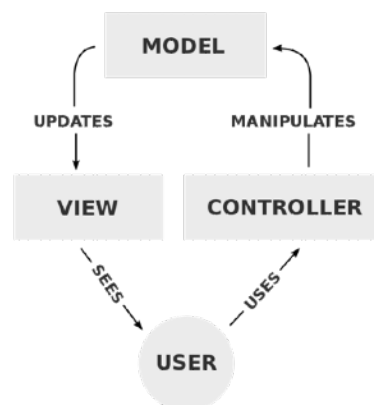
3.1 Arquiteturas e padrões de desenvolvimento

No processo de desenvolvimento de software, é importante que seja escolhida a arquitetura que mais se encaixe no contexto da aplicação, levando em conta manutenção, testabilidade e facilidade de compreensão.

Uma das arquiteturas mais comumente utilizadas no desenvolvimento de aplicações móveis é a MVC (*Model View Controller*). Essa arquitetura é composta por três elementos principais:

- **Model:** é a camada responsável pela lógica da aplicação, encarregada do comportamento dos dados do sistema.
- **View:** é a camada de fato visível ao usuário. É responsável por receber as interações do usuário e repassá-las às outras camadas, bem como receber as alterações de outras camadas e refleti-las ao usuário.
- **Controller:** é a camada que faz o papel de ponte entre o modelo e a visualização. Nela, os comandos dados pelo usuário na camada de visualização são manipulados e enviados para o modelo que, por sua vez, tem sua resposta materializada em atualizações na visualização.

Figura 3.1.1 - Diagrama de um MVC

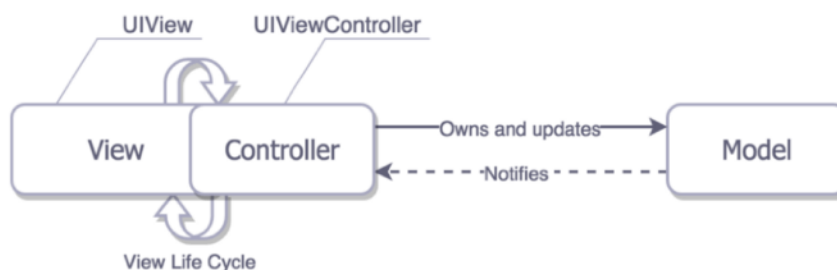


Fonte: (MVC, 2010)

Embora o MVC seja bastante utilizado, no caso de aplicações iOS, o mais comum é encontrar uma variante dessa arquitetura. Nela, há o componente *ViewController*, que é uma união da camada de visualização e de controle, na qual a *view* delega suas responsabilidades para o *controller*, criando um ciclo de vida entre a *view* e a *controller*.

Uma das maiores desvantagens dessa abordagem é que esse componente, a *ViewController*, admite muitas responsabilidades, culminando em um grande acoplamento entre a *view* e o *controller*. Nesse componente, são feitas chamadas de rede, configuração dos elementos visíveis na tela, recepção de eventos do usuário - e seu tratamento -, e até mesmo navegação, podendo gerar arquivos muito grandes e difíceis de serem reutilizados ou até mesmo mantidos. Essas *view controllers* muito extensas são popularmente chamadas de *Massive View Controllers*, indicando seu teor literalmente massivo, com grande quantidade de código em uma única classe e muitas responsabilidades em um único controlador.

Figura 3.1.2 - MVC da Apple



Fonte: ORLOV, 2015

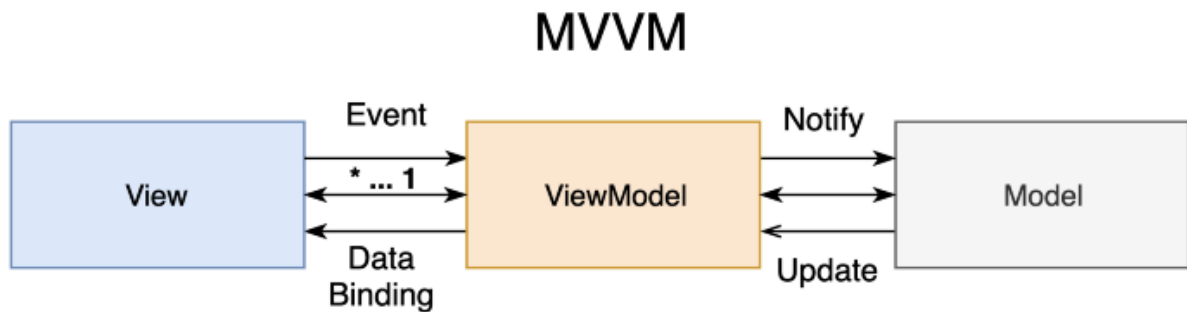
A existência dessas *Massive View Controllers* evidencia a necessidade de mais camadas na arquitetura, objetivando diminuir o acoplamento e a responsabilidade de cada uma delas, e também facilitar o entendimento do código.

Para isso, foi escolhido o padrão de design MVVM, que provê uma limpa separação de responsabilidades entre a parte de visualização e o domínio da aplicação (KUDINOV, 2019).

Nesse padrão, um novo elemento é adicionado, a *ViewModel*. Cada *View* possui uma *ViewModel* associada, da qual obtém os dados que serão mostrados na tela através do processo de *data binding*, e para a qual repassa as ações do usuário. A *view model* é, então, basicamente uma representação da *view* e de seu estado, independente de frameworks de visualização. (ORLOV, 2015). O processo de amarração (*binding*) consiste em a *view* observar propriedades da *view model* e atualizar a tela de acordo com os valores dessas propriedades. Dizer que as propriedades da *view model* são observáveis significa que quaisquer alterações que elas sofrerem serão notificadas a quem as estiver observando. Dessa forma, cada observador poderá tratar desse novo dado de acordo com seu contexto.

Cada *ViewModel*, por sua vez, possui um *Model*, e o atualiza de acordo com o processamento dos eventos recebidos da *View*, e espera notificações desse mesmo *Model* sobre atualizações de dados provenientes da lógica de negócio.

Figura 3.1.3 - Padrão MVVM

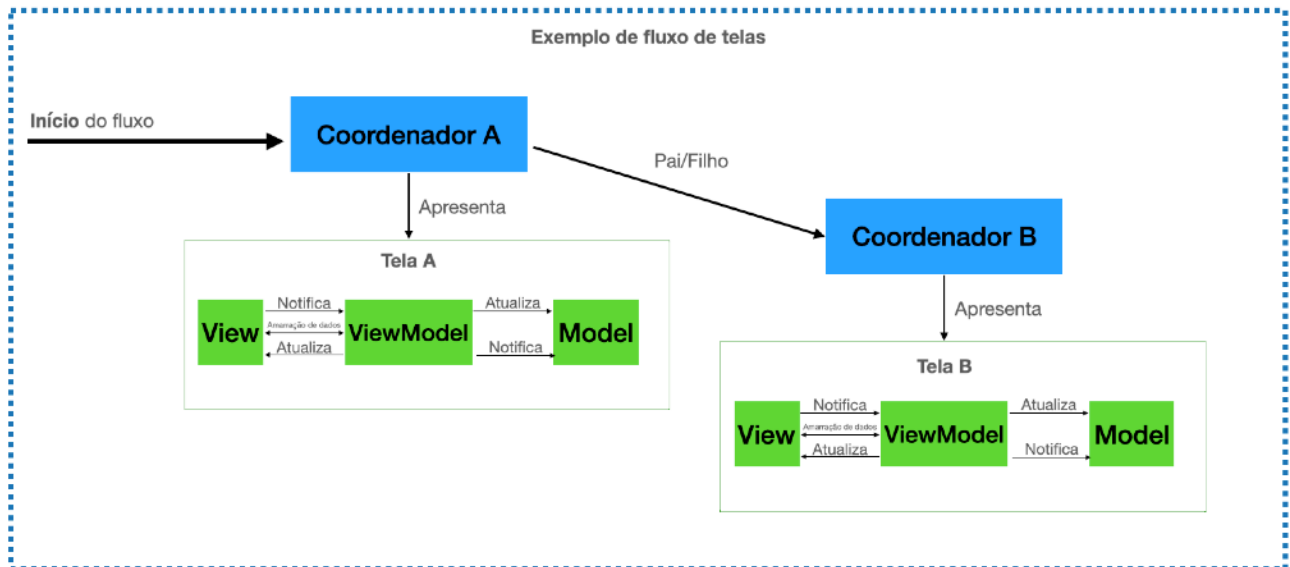


Fonte: KUDINOV, 2019

Até agora, resolvemos as responsabilidades da lógica de negócio e da visualização, através de uma interação bem definida entre essas camadas. No entanto, em uma aplicação composta por várias cenas, o MVVM deixa a desejar em termos de navegação. Como a utilização desses padrões visa, dentre outros aspectos, a separação clara de responsabilidades, é preciso de uma camada que se encarregue pela navegação entre as cenas. Para isso, utilizaremos *Coordinators*.

Um **Coordenador** (*coordinator*) é o elemento que será responsável pela apresentação das cenas em tela e pelo roteamento entre as telas das aplicações. Cada coordenador apresenta pelo menos uma cena inicial, e pode ter N coordenadores filhos, que levarão a outras cenas, e assim por diante, como exemplificado na Figura 3.1.4. Por exemplo: supondo uma cena de uma lista de compras, onde é possível selecionar uma das entradas e visualizar os detalhes desse item. O coordenador dessa lista a apresenta na tela, ou seja, instancia a *view model* e a *view* da cena de lista de compras e exibe essa cena em tela. Ao receber o evento do usuário de que um elemento da lista foi selecionado, a *view* o repassa para a *view model*, que o repassa para o coordenador. O coordenador é, então, responsável apenas por apresentar a tela de detalhe de item, seja diretamente ou através da instanciação de um novo coordenador, e assim por diante.

Figura 3.1.4 - Exemplo de fluxo de telas com Coordenadores



Fonte: o autor, adaptado de SANTA, 2017

3.2 Metodologia de trabalho

Existem diversas metodologias para o desenvolvimento de um *software*. Uma das mais comumente usadas é a Metodologia Ágil (ÁGIL, 2014), que engloba diversas técnicas e processos para serem aplicados no dia-a-dia de desenvolvimento. Uma dessas metodologias é o *Scrum*, uma metodologia ágil para gestão e planejamento de projetos de software. (ÁGIL, 2014).

O *Scrum* um *framework* tipicamente usado por equipes, composto por diversos personagens e cerimônias, com reuniões constantes de *feedback* e de acompanhamento diário. No entanto, como esse projeto foi realizado por um único desenvolvedor, não seriam necessários todos esses eventos, de modo que foram selecionados alguns aspectos que fariam sentido no processo de um desenvolvedor solo. Esses aspectos serão apresentados a seguir:

- **Product backlog:** o *product backlog* é uma lista de pendências que deverão ser resolvidas visando o bom funcionamento do produto final. Essas pendências são, geralmente, organizadas por ordem de prioridade e importância dentro da aplicação.
- **Sprint:** uma *sprint* é um ciclo de desenvolvimento, no qual são definidos itens do *backlog* que serão trabalhados. As *sprints* não têm duração uniforme, podendo durar desde uma semana até quatro semanas.
- **Sprint retrospective:** a retrospectiva de uma *sprint* é o momento no qual é avaliado o que foi desenvolvido no ciclo e no qual são identificados pontos de acerto e melhoria. É

também interessante para servir de orientação para próximas *sprints*, com o aprendizado de ciclos anteriores para a contínua melhoria do processo.

4 PROPOSTA DE ARQUITETURA

Considerando a solução discutida anteriormente, foi proposta uma lista de funcionalidades que devem compor o sistema. Essa lista de requisitos foi criada pensando em como seria o cotidiano de um usuário do Gaveta, a fim de atingir um grupo necessário de funções que permitissem uma experiência confortável na aplicação. Como a aplicação tem teor majoritariamente social, essas funcionalidades do produto inicial são baseadas em interações sociais, por exemplo na composição de postagens e comentários.

4.1 Funcionalidades

As funcionalidades foram propostas a partir de requisitos que atenderiam as demandas dos potenciais usuários, e definidas a partir de um conjunto mínimo de informações, que englobam os requisitos necessários no sistema para a execução dessa funcionalidade, o personagem envolvido em sua execução, o fluxo que representa a jornada do usuário durante a execução da funcionalidade e, opcionalmente, um fluxo de encerramento.

A seguir, a funcionalidade de cadastro de usuário é descrita a título de exemplo. As demais funcionalidades do Gaveta serão descritas no Anexo A.

• Cadastro de usuário

Como estudante, desejo me cadastrar no Gaveta, para poder acessar as matérias do meu curso.

Tabela 4.1 - Cadastro de usuário

Requisitos	Ter baixado o aplicativo Gaveta
Personagem	Estudante
Jornada	<ul style="list-style-type: none">- A partir da tela de entrada, o usuário deverá clicar no botão “Criar Nova Conta”.- O usuário deverá, então, inserir seu e-mail, seu nome e o código de sua instituição de ensino.- O sistema deverá sugerir um nome de usuário a partir do nome fornecido anteriormente, dando ao usuário a opção de aceitá-lo ou de inserir um nome de usuário personalizado.- O usuário poderá, opcionalmente, escolher uma foto de perfil.- Por fim, deverá escolher uma senha de, pelo menos, 8 caracteres.

Encerramento	- Se o cadastro ocorrer com sucesso, o usuário deverá ser levado à área logada. Caso não ocorra, o sistema deverá mostrar uma mensagem informando o insucesso da operação.
--------------	--

- **Login de usuário**

Como estudante, desejo me autenticar no Gaveta, a fim de acessar minha área logada.

- **Seguir disciplinas**

Como aluno, gostaria de seguir disciplinas, a fim de acessar o conteúdo das postagens.

- **Favoritar disciplinas**

Como aluno, gostaria de favoritar disciplinas, a fim de facilitar meu acesso às postagens.

- **Visualizar lista de postagens de uma disciplina**

Como aluno, gostaria de visualizar as postagens de uma disciplina, a fim de acessar o conteúdo da minha comunidade acadêmica no Gaveta.

- **Criar nova postagem em uma disciplina**

Como aluno, gostaria de criar uma nova postagem em uma disciplina, a fim de iniciar um novo tópico de discussão com meus colegas.

- **Visualizar postagem de uma disciplina**

Como aluno, gostaria de visualizar uma postagem em uma disciplina, a fim de acessar a discussão relacionada a ela.

- **Criar novo comentário em uma postagem de uma disciplina**

Como aluno, gostaria de criar um novo comentário em uma postagem, a fim de participar na discussão daquele tópico.

- **Deixar de seguir uma disciplina**

Como usuário, gostaria de deixar de seguir uma disciplina, a fim de não acompanhar mais tópicos que não me interessam mais.

- **Remover uma disciplina da lista de favoritos**

Como usuário, gostaria de remover uma disciplina da minha lista de favoritos, a fim de deixar em prestígio apenas as disciplinas em que tenho maior interesse.

4.2 Esboço da aplicação

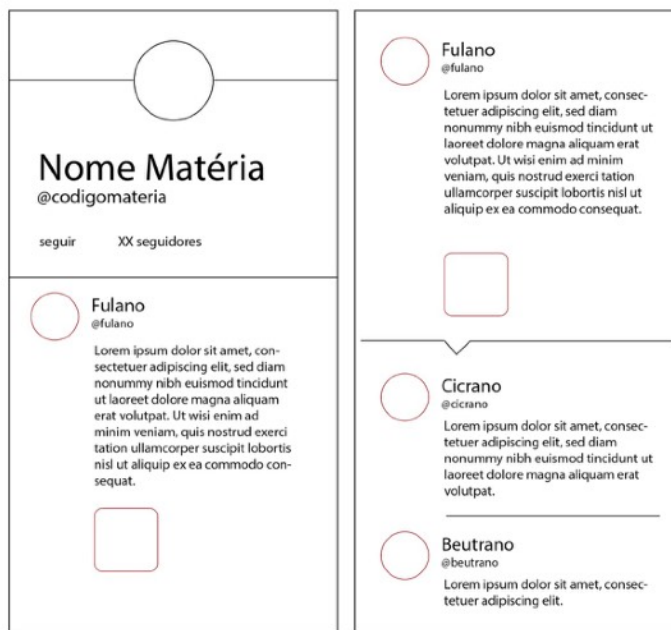
Após a definição das funcionalidades do sistema, foi realizado um primeiro esboço de como ele se materializaria em uma aplicação móvel. Para isso, foram usados *wireframes*, que

nada mais são do que “esqueletos” da aplicação, que geram apenas uma idéia de uma futura concepção. Não utilizam cores, imagens, nem textos reais.

Por serem uma etapa bastante inicial da concepção da aplicação, esses esboços servem de base, mas não representam fielmente o produto final, já que sofrem alterações e atualizações durante o processo de desenvolvimento.

A Figura 4.2 apresenta a primeira versão do que seria a tela de visualização de matéria, à esquerda, e de visualização de postagem, à direita. As versões reais dessas telas após a implementação serão mostradas no Capítulo 5.

Figura 4.2 - Exemplo de esboço da aplicação



Fonte: o autor

4.3 Camadas do sistema

A partir do padrão MVVM discutido no Capítulo 3, o sistema poderá ser visto a partir de três principais camadas, cada uma com sua própria responsabilidade na formação do projeto. Essas camadas não necessariamente dependem umas das outras e serão discutidas a seguir.

4.3.1 Camada de Domínio

A camada de domínio é a camada mais interna da aplicação, (KUDINOV, 2019) conceitualmente não dependendo de nenhuma outra camada ou framework externo para

funcionar. É nessa camada que está toda a lógica do negócio, onde são definidos os modelos, os casos de uso e também os requerimentos de dados.

Os **modelos** de domínio são a representação das entidades conceituais que o sistema possui. No Gaveta, por exemplo, alguns exemplos de modelos definidos são as disciplinas (*Subjects*), os usuários (*Users*) e outras entidades que serão mostradas na Seção 4.4.

Um **caso de uso** é uma lista de ações e passos de comunicação entre um personagem e um sistema que são necessários para atingir determinado objetivo. (MOLOCHKO, 2017). Dessa forma, cada uma das funcionalidades descritas na Seção 4.1 tem as informações suficientes para a execução de um caso de uso, e sua implementação está na camada de domínio do sistema.

Os **requerimentos de dados** são interfaces definidas no domínio que deverão ser implementadas pela camada de dados. Essas interfaces têm o objetivo de evidenciar os dados necessários para o correto funcionamento da camada de domínio, independente da origem real desses dados, como um servidor remoto ou um banco de dados local. Por exemplo: a camada de dados, que será discutida em na Seção 4.3.4, pode ter uma classe que implementa um desses requerimentos do domínio e busca os dados de algum servidor externo, ou então do banco de dados local, e isso não fará diferença para a visão do domínio, uma vez que é ele quem define os dados de entrada e o que ele espera de retorno.

Figura 4.3.1 - Exemplo de requerimento de dados

A imagem mostra um editor de código com um fundo escuro e uma interface de usuário com três botões de cor (vermelho, amarelo, verde) no canto superior esquerdo. O código exibido é em Swift e define um protocolo chamado 'UserRepository'.

```
public protocol UserRepository {
    func login(
        with email: String,
        password: String,
        completion: @escaping(Result<User,String>) -> Void
    )
}
```

Fonte: o autor

A Figura 4.3.1 mostra um exemplo de definição dos dados necessários para um *login* de usuário. A camada de domínio define que, para o *login*, entrará com os dados de *e-mail* e senha do usuário e espera, como resultado de sucesso, um objeto *User* (entidade de modelo) ou uma *string* com uma mensagem de erro. Ao domínio, fica transparente de onde vêm esses dados, dando a liberdade de várias fontes atenderem a esses requerimentos.

4.3.2 Camada de Apresentação

A camada de apresentação é a responsável pela coordenação e visualização dos dados que serão apresentados para o usuário, e é nela que estão localizados os *Coordinators*, as *Views*, e as *ViewModels*. A camada de apresentação depende apenas da camada de domínio, uma vez que as *ViewModels* executam um ou mais casos de uso (KUDINOV, 2019), que proverão os dados que serão apresentados em tela, obtidos pela *view* através do processo de amarração, explicado na Seção 3.1.

É nessa camada que são tratadas as interações do usuário com a aplicação, tornando-a responsável por navegar entre cenas e coordenar o fluxo de dados entre elas, mantendo consistência nas informações que serão visíveis ao usuário.

4.3.3 Camada de Dados

A camada de dados é a responsável por combinar uma ou mais fontes de dados e prover as informações requeridas pelo domínio. Nessa camada estão os **Repositórios de Dados**, que nada mais são do que as implementações das interfaces dos requerimentos de dados do domínio. É nessa etapa que são convertidos os objetos provenientes de, por exemplo, um servidor remoto em objetos do domínio.

Figura 4.3.3 - Implementação de repositório de dados



```
import Domain

final class MainUserRepository: UserRepository {

    /* Propriedades serão omitidas */

    func login(
        with email: String,
        password: String,
        completion: @escaping(Result<User, String> -> Void)
    ) {

        someNetworkingService.loginUser(email, password) { result in
            switch result {
            case .success(let remoteUserObject):
                completion(.success(remoteUserObject.toDomain()))
            case .failure(let error):
                completion(.failure(error.description))
            }
        }
    }
}
```

Fonte: o autor

A Figura 4.3.3 mostra a simulação de uma implementação do exemplo mostrado na Figura 4.3.1. Nele, a classe *MainUserRepository* implementa a função *login*, requerida pelo protocolo *UserRepository*, autenticando e buscando os dados do usuário de um servidor externo. No sucesso, o objeto recebido de resposta é mapeado no objeto de domínio, garantindo que todas as informações requeridas serão retornada, nada menos e nada mais.

Para melhor compreensão de como estão esquematizadas essas camadas, a Figura 4.3.4 mostra como funciona um fluxo de dados da aplicação.

Figura 4.3.4 - Camadas da aplicação



Fonte: o autor, adaptado de KUDINOV, 2019

4.4 Entidades do sistema

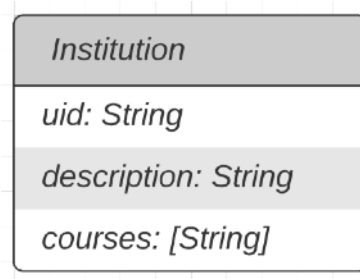
Nesta seção, serão descritas as entidades que compõem o sistema.

4.4.1 Instituição

A entidade de instituição é o que determina o espaço em que os usuários estarão inseridos. Cada usuário pertencerá a uma única instituição que, por sua vez, deverá ter uma ou mais disciplinas. Cada instituição possui, como mostrado na Figura 4.4.1:

- Um código, que a identifica univocamente no sistema
- Uma descrição, que permite uma visualização amigável de sua representação
- Uma lista de identificadores, que são referências às disciplinas que estão contidas nessa instituição

Figura 4.4.1 - Entidade de Instituição



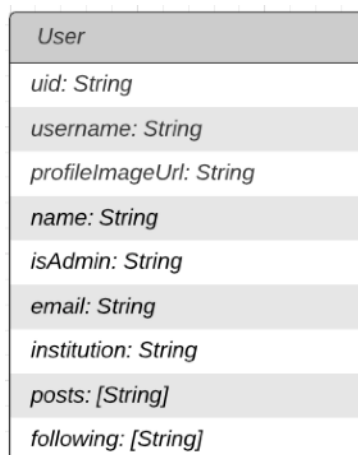
Fonte: o autor

4.4.2 Usuário

A entidade de usuário é a que contém as informações fornecidas pelo aluno no momento do cadastro, bem como as informações de disciplinas e postagens realizadas durante o uso da aplicação. Cada usuário possui, como mostrado na Figura 4.4.2:

- Um endereço de e-mail, fornecido no cadastro
- Uma lista de identificadores, que são referências às disciplinas que o usuário segue
- Uma lista de identificadores, que são referências às postagens feitas pelo usuário
- Um identificador da instituição à qual está associado
- Uma flag que indica se esse usuário é um administrador
- Um campo com o nome
- Um campo com seu identificador único, fornecido pelo servidor no momento da criação
- Um campo com o endereço da imagem de perfil
- Um campo com o nome

Figura 4.4.2 - Entidade de usuário



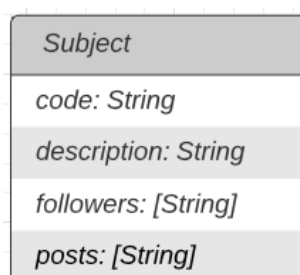
Fonte: o autor

4.4.3 Disciplina

A entidade da disciplina contém as informações de seguidores e de postagens. Vale ressaltar que essa entidade não referencia nenhuma instituição, mas sim é referenciada por uma ou mais instituições. Isso é interessante para alunos de diferentes cursos interagirem na mesma área da disciplina. Cada disciplina possui, como mostrado na Figura 4.4.3:

- Um código único, que a identifica no sistema
- Uma descrição, geralmente o título dessa disciplina
- Uma lista de identificadores, com referências aos usuários que a seguem
- Uma lista de identificadores, com referências às suas postagens

Figura 4.4.3 - Entidade de disciplina



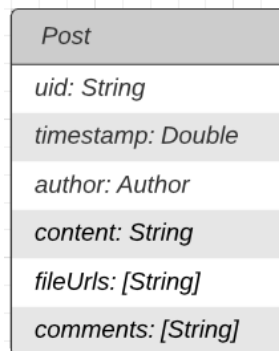
Fonte: o autor

4.4.4 Postagem

A entidade de uma postagem contém as informações de seu autor, seu conteúdo, seus anexos e também de sua lista de comentários. Cada postagem possui, como mostrado na Figura 4.4.4:

- Um código único, que a identifica no sistema
- Um campo com um objeto Autor, com os dados do autor da postagem
- Um campo de conteúdo, com a parte textual da postagem
- Uma lista de endereços referentes aos arquivos anexados à postagem
- Uma lista de identificadores, com referências aos seus comentários
- Uma *timestamp*, que define a data de sua criação

Figura 4.4.4 - Entidade de postagem



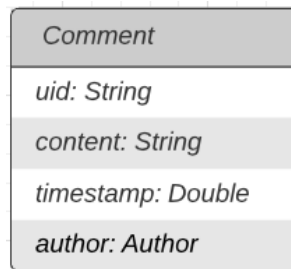
Fonte: o autor

4.4.5 Comentário

A entidade de um comentário pode ser vista como uma versão enxuta de uma postagem. Cada comentário possui, como mostrado na Figura 4.4.5:

- Um código único, que o identifica no sistema
- Um campo com um objeto Autor, com os dados do autor do comentário
- Um campo de conteúdo, com a parte textual do comentário
- Uma *timestamp*, que define a data de sua criação

Figura 4.4.5 - Entidade de comentário



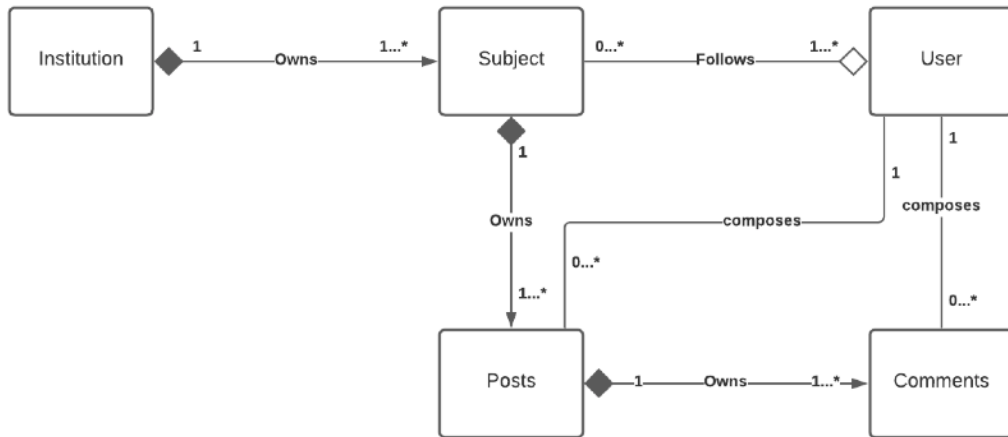
Fonte: o autor

4.5 Relacionamento entre as entidades

As entidades descritas acima relacionam-se entre si. Para compreender essas interações, serão definidas três relações, também mostradas na Figura 4.5:

- *Owns*: indica posse, por exemplo: uma disciplina (*subject*) possui (*owns*) uma lista de 0 a * postagens (*posts*).
- *Follows*: indica a relação entre usuários e disciplinas. Por exemplo: U usuários (*user*) seguem (*follows*) de 0 a * disciplinas (*subjects*).
- *Composes*: indica a relação de criação de uma nova instância de uma entidade, a partir de outra. Por exemplo: um usuário (*user*) compõe (*composes*) de 0 a * postagens (*posts*).

Figura 4.5 - Relacionamento entre as entidades do sistema



Fonte: o autor

5 IMPLEMENTAÇÃO

Neste capítulo, serão apresentadas as principais ferramentas utilizadas para a implementação do sistema Gaveta na plataforma iOS, bem como a experiência resultante do desenvolvimento.

5.1 Ferramentas Utilizadas

A seguir, serão descritas as principais ferramentas utilizadas para a implementação do Gaveta na plataforma iOS.

5.1.1 Git e GitHub

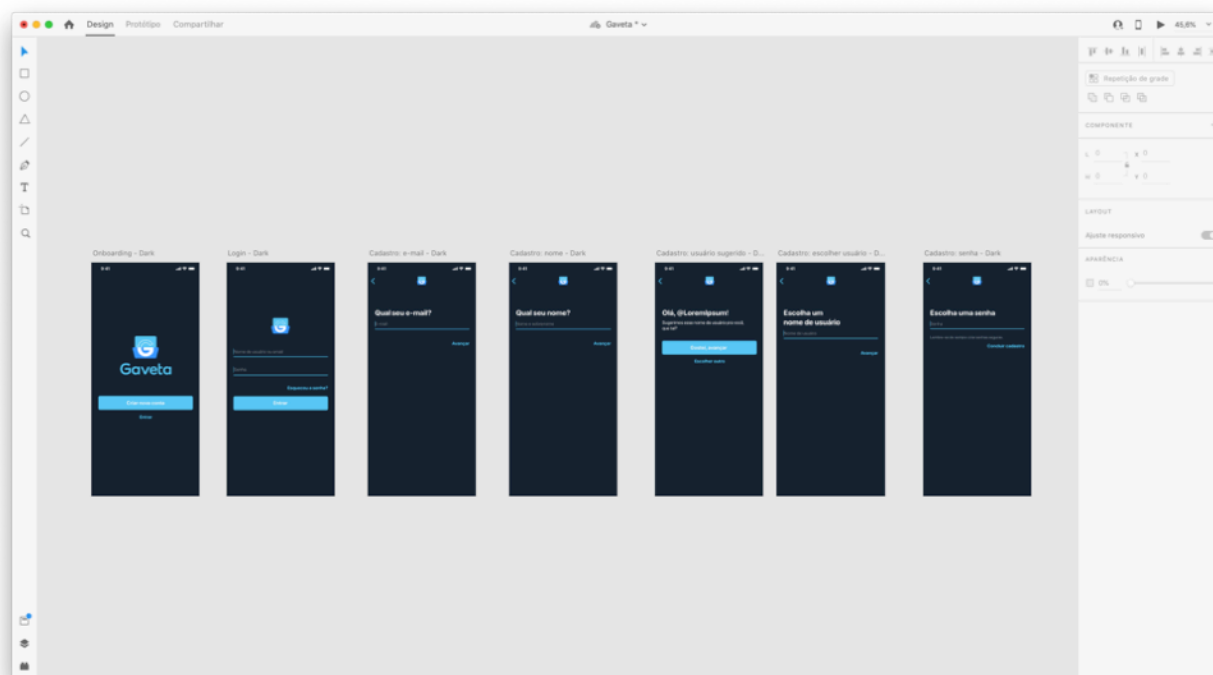
O Git é um sistema de versionamento proposto inicialmente no desenvolvimento do sistema Linux, hoje amplamente utilizado para códigos de projetos. Esse sistema provê ferramentas úteis para um fluxo de desenvolvimento de software, como o conceito de *branches* que permitem, por exemplo, que várias *features* de um sistema sejam desenvolvidas em paralelo e posteriormente integradas no fluxo principal da aplicação.

O GitHub é uma plataforma de hospedagem de código para controle de versão e colaboração. (GITHUB, 2020). Ele foi escolhido para o desenvolvimento do Gaveta por seu fácil uso e por permitir a criação de repositórios privados, funcionando também como uma cópia extra dos arquivos referentes ao projeto, que não ficam apenas salvo localmente no computador.

5.1.2 Adobe XD

O XD é uma ferramenta que faz parte do Creative Cloud da Adobe e permite que, gratuitamente, sejam feitos protótipos de telas de diversos tipos de aplicações, inclusive móveis. Ele foi utilizado em uma etapa intermediária entre o esboço da aplicação discutido anteriormente e a implementação propriamente dita, etapa essa de definição de cores, fontes, caixas de texto e outros elementos visuais da aplicação. O XD permite, também, que sejam feitas interações entre as telas do protótipo, simulando cliques e a lógica de navegação entre telas como se estivessem implementadas em um dispositivo.

Figura 5.1.2 - Tela do Adobe XD



Fonte: o autor

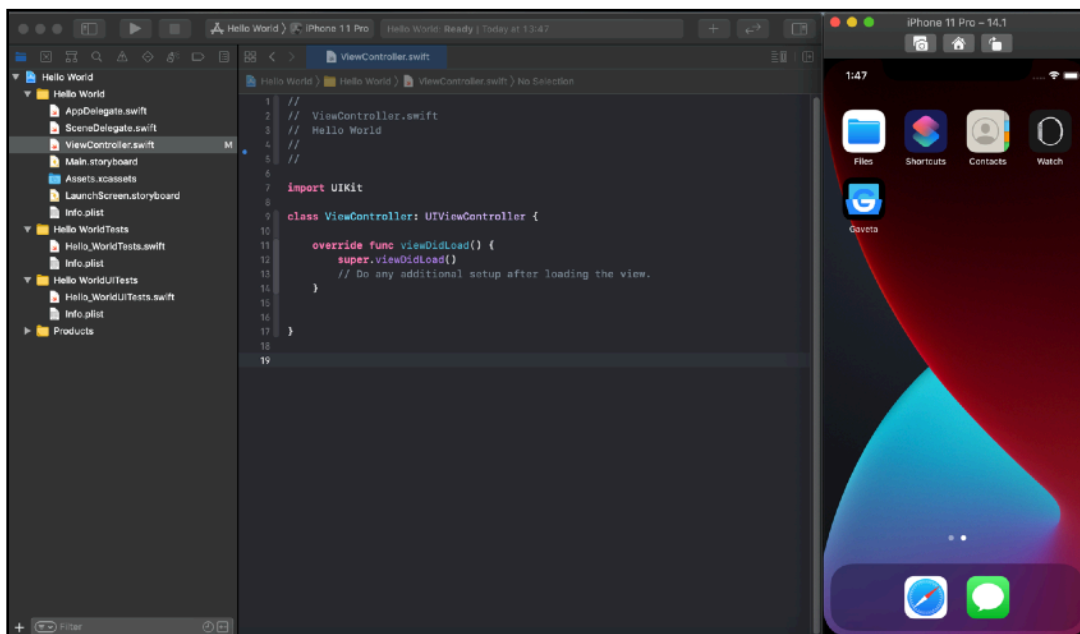
5.1.3 Xcode e Swift

Para a implementação da aplicação móvel para iOS do Gaveta, foi escolhida a linguagem Swift, da Apple. O Swift é uma linguagem de programação de propósito geral construída usando uma abordagem moderna a segurança, performance e padrões de design de software (INC, 2020).

Embora existam outras linguagens de programação que possibilitem escrever aplicações para iOS, o Swift é a linguagem nativa construída para tais aplicações. O código nativo traz algumas vantagens, como a integração com a IDE Xcode, que será discutida a seguir, maior controle sobre os componentes de interface, e velocidade de compilação.

O Xcode é a IDE da Apple feita para desenvolvimento de aplicações para macOS e iOS. Com ele, o desenvolvimento de aplicações iOS fica mais fácil e intuitivo, uma vez que podemos contar com integração direta com o compilador de Swift, com o *debugger* do LLVM, e também com recursos de *code-completion*, que aceleram ainda mais o desenvolvimento.

Figura 5.1.3 - Tela do Xcode com código em Swift (esq) e Simulador de iOS (dir)



Fonte: o autor

5.1.4 Firebase

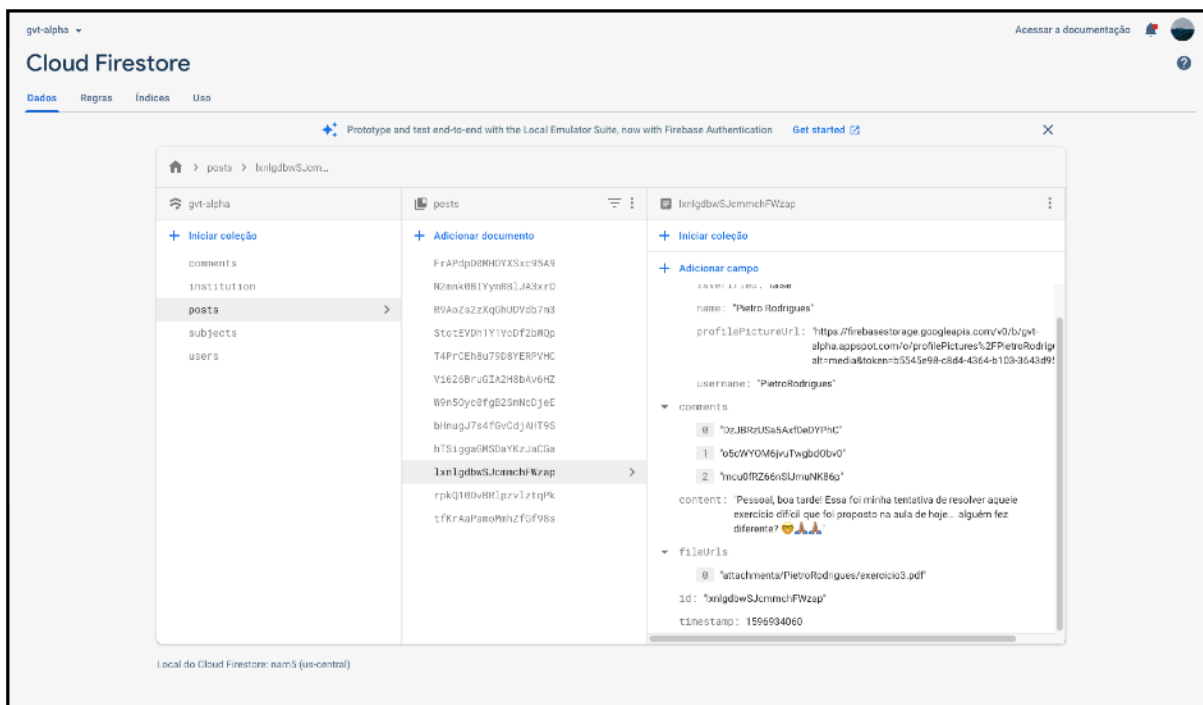
O Firebase é uma plataforma criada pela Google que visa facilitar o desenvolvimento de software *mobile* e *web*. Um dos seus produtos, utilizado no Gaveta, é o *Cloud Firestore*, que é um banco de dados NoSQL flexível e escalonável (LLC, 2020) hospedado na nuvem.

Com o *Cloud Firestore*, podemos armazenar dados em documentos JSON e armazenar esses documentos em Coleções, que nada mais são do que contêineres de documentos (LLC, 2020), que facilitam a organização e as consultas posteriormente. Esse produto possui um painel na web, através do qual podemos visualizar os dados e também alterá-los, e também possui uma SDK (*Software Development Kit*) que opera harmonicamente com um projeto iOS, permitindo consultas rápidas e fáceis de serem feitas.

É nesse banco de dados que os dados do Gaveta estão armazenados. Foram compostas cinco coleções, uma de **instituições**, uma de **usuários**, uma de **matérias**, uma de **postagens** e uma de **comentários**. Os documentos salvos em cada uma dessas coleções são objetos bem estruturados que dizem respeito às entidades do sistema.

Além do *Cloud Firestore*, também foi utilizado o produto *Cloud Storage*. Nele, foram salvos todos os arquivos enviados por usuários, sejam eles as fotografias do perfil de usuário, ou então os anexos enviados juntos às postagens. O *Cloud Storage* também possui uma SDK que opera com o projeto iOS, permitindo que facilmente sejam manipulados objetos no banco de dados junto com referências aos objetos no armazenamento.

Figura 5.1.4 - Painel do Cloud Firestore



Fonte: o autor

5.2 Etapas do desenvolvimento

A implementação do projeto foi dividida em 9 (nove) etapas, de diferentes durações, que representaram a construção do protótipo de uma maneira incremental. As atividades realizadas em cada uma dessas etapas são descritas na Tabela 5.2.

Essas etapas são análogas às *sprints* da Metodologia Ágil, no entanto, como esse projeto era composto principalmente por um único desenvolvedor, essas etapas tiveram diferentes durações, dada a diferença na complexidade da execução das tarefas que as englobam.

Tabela 5.2 - Etapas do projeto

<p>Etapa 0</p>	<ul style="list-style-type: none"> - Esboço do projeto, pesquisa - Definição de cores, e outras diretrizes de design no XD <p><i>Duração: 1 semana</i></p>
----------------	--

Etapa 1	<ul style="list-style-type: none"> - Criação do projeto base - Criação de uma camada comum com extensões úteis de imagens, cores, e códigos potencialmente reutilizáveis por outras camadas da aplicação <p><i>Duração: 2 semanas</i></p>
Etapa 2	<ul style="list-style-type: none"> - Criação dos modelos no servidor e configuração da camada de dados no projeto <p><i>Duração: 1 semana</i></p>
Etapa 3	<ul style="list-style-type: none"> - Criação das cenas necessárias para o cadastramento de usuário e login - Construção do fluxo de dados para o cadastramento e integração com o servidor <p><i>Duração: 2 semanas</i></p>
Etapa 4	<ul style="list-style-type: none"> - Criação da cena da página inicial - Configuração da lógica de apresentação da tela inicial <p><i>Duração: 3 semanas</i></p>
Etapa 5	<ul style="list-style-type: none"> - Criação da tela de seguir disciplinas - Integração de novas disciplinas com a tela inicial <p><i>Duração: 1 semana</i></p>
Etapa 6	<ul style="list-style-type: none"> - Criação das cenas de disciplina e de postagem - Listagem de postagens na disciplina e de comentários na postagem <p><i>Duração: 2 semanas</i></p>
Etapa 7	<ul style="list-style-type: none"> - Criação das cenas de composição, sem contemplar anexos - Submissão de novos posts e comentários para o servidor <p><i>Duração: 1 semana</i></p>
Etapa 8	<ul style="list-style-type: none"> - Implementação de envio de anexos em nova postagem - Implementação de exibição de anexos em visualização de postagem <p><i>Duração: 2 semanas</i></p>

Fonte: o autor

5.3 Experiência da aplicação

Nesta seção, serão apresentadas as principais telas do Gaveta. A Figura 5.3 mostra um usuário com a tela inicial do Gaveta aberta em seu *smartphone*.

Figura 5.3 - Apresentação do Gaveta



Fonte: o autor

5.3.1 Cena de entrada

A cena de entrada é o ponto inicial de quem executa o aplicativo. A ideia é que o *design* seja limpo e com apenas as informações necessárias. Nessa cena, o são visíveis dois botões:

- **Criar nova conta:** responsável por levar o usuário ao fluxo de cadastro
- **Entrar:** responsável por levar o usuário ao fluxo de *login*, uma vez já cadastrado

Figura 5.3.1 - Cena de Entrada



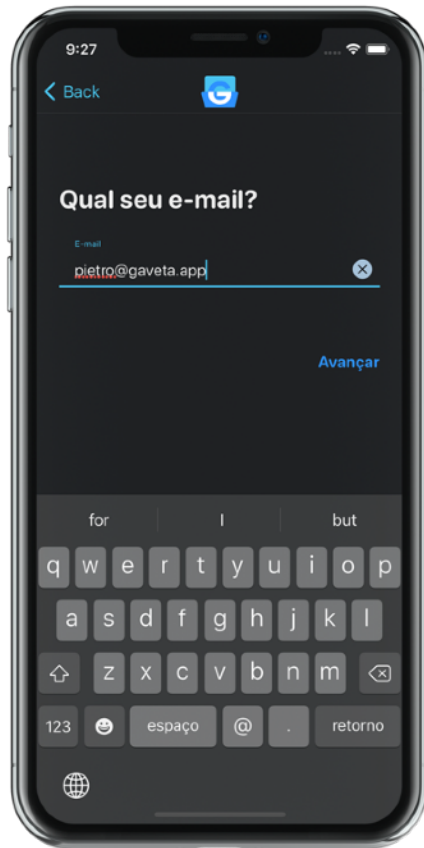
Fonte: o autor

5.3.2 Cena de inserção de dados

A cena de inserção de dados é pensada para ser responsável por receber os dados que o usuário digitar em todo o fluxo de cadastro e de *login* do usuário. Pensada em proporcionar uma experiência agradável ao usuário, essa cena espera receber um único dado a cada vez que é instanciada.

Além disso, o estilo do teclado pode variar conforme o tipo de dado a ser inserido, por exemplo, no caso de um e-mail, o teclado já aparece com o '@' e o '.' em primeiro plano, a fim de facilitar a digitação.

Figura 5.3.2 - Cena de inserção de dados



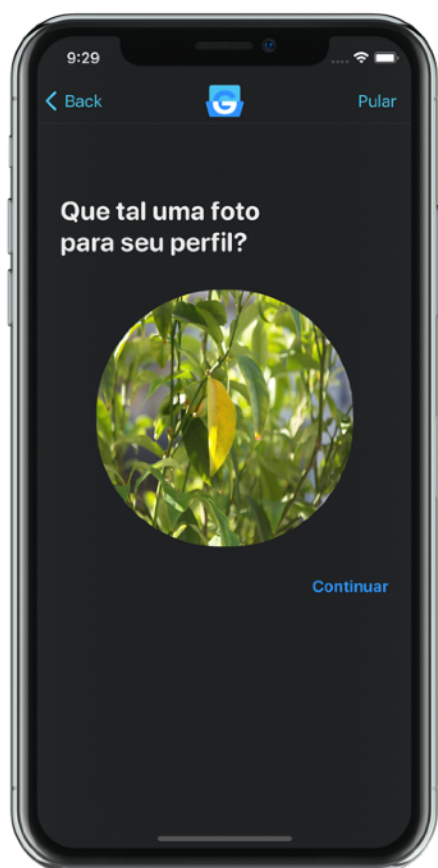
Fonte: o autor

5.3.3 Cena de inserção de foto de perfil

A cena de inserção de foto de perfil é utilizada quando o usuário quiser adicionar ou alterar a imagem que será associada ao seu perfil nas postagens. Por ser uma etapa opcional nos fluxos em que aparece, essa cena conta com a opção de ser descartada sem exigir a foto, através do botão “Pular” no canto superior direito.

É permitido adicionar fotos do álbum do celular, e opcionalmente recortá-las para melhor atender a necessidade do usuário.

Figura 5.3.3 - Cena de inserção de foto de perfil



Fonte: o autor

5.3.4 Cena de sugestão automática de nome de usuário

No processo de criação de conta, o usuário precisa definir um nome de usuário, através do qual será identificado. Para facilitar a experiência, o sistema automaticamente sugere um nome de usuário, primeiramente tentando a concatenação do nome e sobrenome (ex.: Bernardo Castro teria BernardoCastro) e, caso esse nome não esteja disponível para uso, o sistema gera um token que é anexado ao final dessa concatenação (ex.: BernardoCastroF8FG). Essa tela contém duas ações:

- **Gostei, avançar:** sinaliza que o usuário aceitou o nome que foi sugerido pelo sistema e quer avançar no fluxo
- **Escolher outro:** leva o usuário para uma tela de entrada de dados, na qual ele poderá entrar manualmente com um nome de usuário de sua preferência.

Figura 5.3.1 - Cena de sugestão automática de nome de usuário



Fonte: o autor

5.3.5 Cena inicial

A cena inicial do aplicativo é a primeira interação do usuário com a área logada. Nela, é possível acessar as matérias que segue, seguir (ou deixar de seguir) matérias, e acessar suas matérias favoritas.

O conceito de manter uma lista separada com matérias favoritas é para dar mais prestígio às tarefas nas quais há mais interesse em um determinado período, por exemplo as matérias que o aluno estiver cursando naquele semestre.

Figura 5.3.5 - Cena inicial

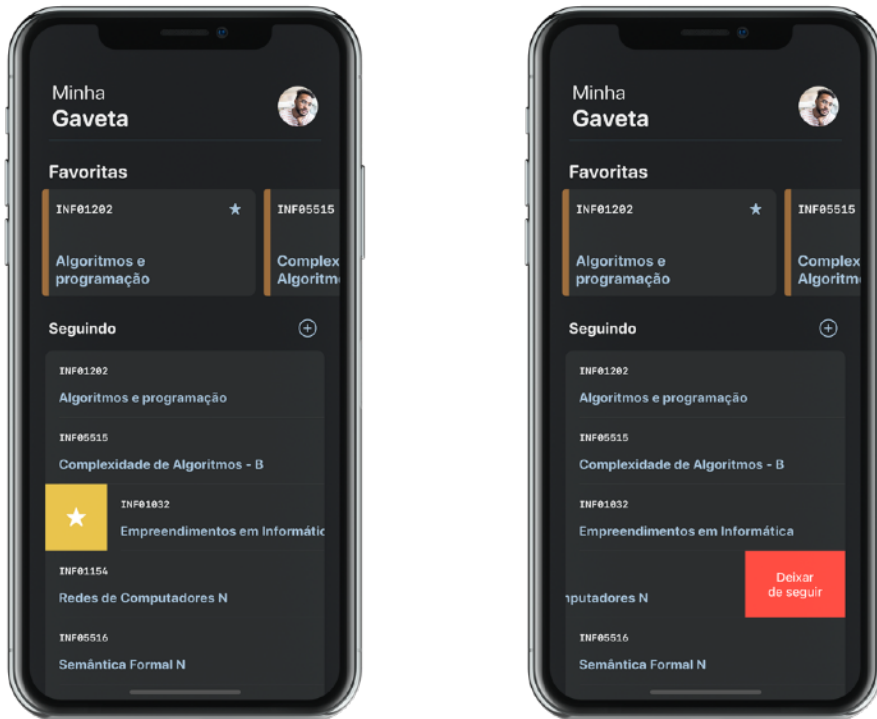


Fonte: o autor

Para seguir novas disciplinas, o usuário acessa a lista de disciplinas de sua instituição de ensino (mostrado em 4.2.5) através do botão '+' localizado à direita do título "Seguindo".

Uma vez que segue matérias, o usuário poderá deixar de segui-las apenas deslizando seu título da direita para a esquerda, ou então favorita-las, deslizando seu título da esquerda para a direita, como mostra a imagem 4.2.4.1.

Figura 5.3.5.1 - Favoritar e Deixar de Seguir Disciplinas



Fonte: o autor

5.3.6 Cena de seguir disciplinas

A cena de seguir disciplinas consiste em uma tabela com as disciplinas disponíveis na instituição de ensino do usuário. Nela, são mostrados o código e a descrição da disciplina, ordenados alfabeticamente para facilitar a organização. Não aparecem nessa lista as disciplinas que o usuário já segue, mas sim as disponíveis para seguir. Quando o usuário toca em uma dessas disciplinas, o sistema registra que ele agora a segue.

Figura 5.3.6 - Cena de Seguir Disciplinas



Fonte: o autor

5.3.7 Cena de disciplina

A cena de disciplina é composta pelas informações de uma disciplina (código e nome) e uma lista de postagens, ordenadas da mais recente para a mais antiga. Nessa tela, as postagens contêm as informações do autor (foto de perfil, nome e nome de usuário), um resumo do conteúdo textual da postagem, e opcionalmente o símbolo do clipe de papel, que indica quando há anexos associados à postagem.

No canto superior direito, há um botão de composição, a partir do qual o usuário será levado para a tela de compor nova postagem, onde poderá adicionar um conteúdo textual e, opcionalmente, anexos.

Figura 5.3.7 - Cena de Disciplina



Fonte: o autor

5.3.8 Cena de postagem

Quando uma postagem é selecionada a partir da lista de postagens de uma matéria, a cena de postagem é exibida. Nela, em destaque é apresentado o conteúdo original da postagem, com as informações do autor (foto de perfil, nome, nome de usuário), a data, o clipe de papel que indica se há postagens, o conteúdo textual da postagem e uma lista horizontal de anexos, representados por ícones ou miniaturas de seu conteúdo. Logo abaixo do conteúdo da postagem, há a lista de comentários associados a ela.

No canto superior direito, há o botão de composição, a partir do qual o usuário é levado para a cena de composição de comentário, onde poderá criar um conteúdo textual relacionado à postagem.

Quando um anexo é selecionado, é apresentada a experiência padrão do sistema de visualização de mídia.

Figura 5.3.8 - Cena de Postagem

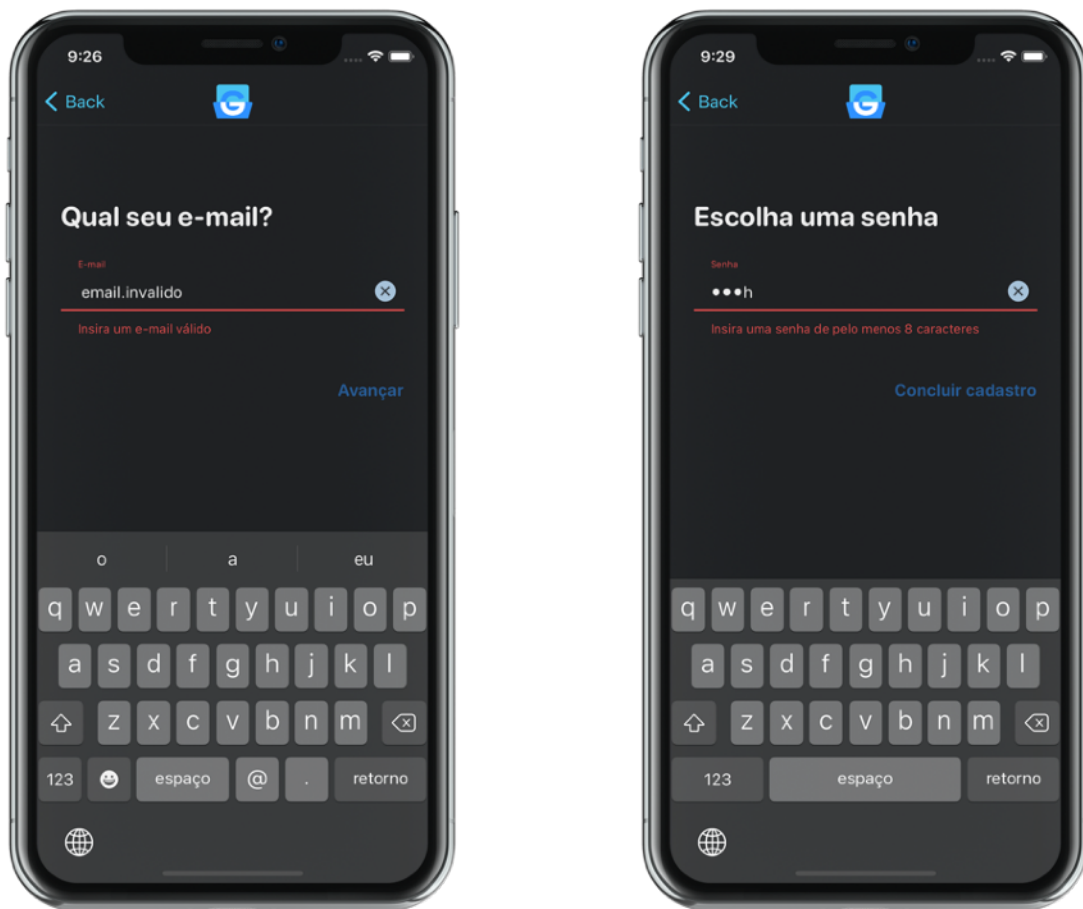


Fonte: o autor

5.3.9 Cenários de erro

Quando algo não ocorre como o esperado ou alguma entrada do usuário é inválida, ele é avisado através de *feedbacks* visuais na tela.

Figura 5.3.9 - Cenários de Erro



Fonte: o autor

5.4 Modelo de negócio

A solução foi pensada para proporcionar aos alunos um ambiente mais próximo ao seu contexto acadêmico real, com disciplinas pertinentes a seus cursos e administrada por alguma entidade, ao invés de um usuário final, para evitar o problema de continuidade citado no quarto parágrafo do Capítulo 1. Dessa forma, potenciais clientes do Gaveta seriam um Diretório Acadêmico ou uma Comissão de Graduação de uma universidade.

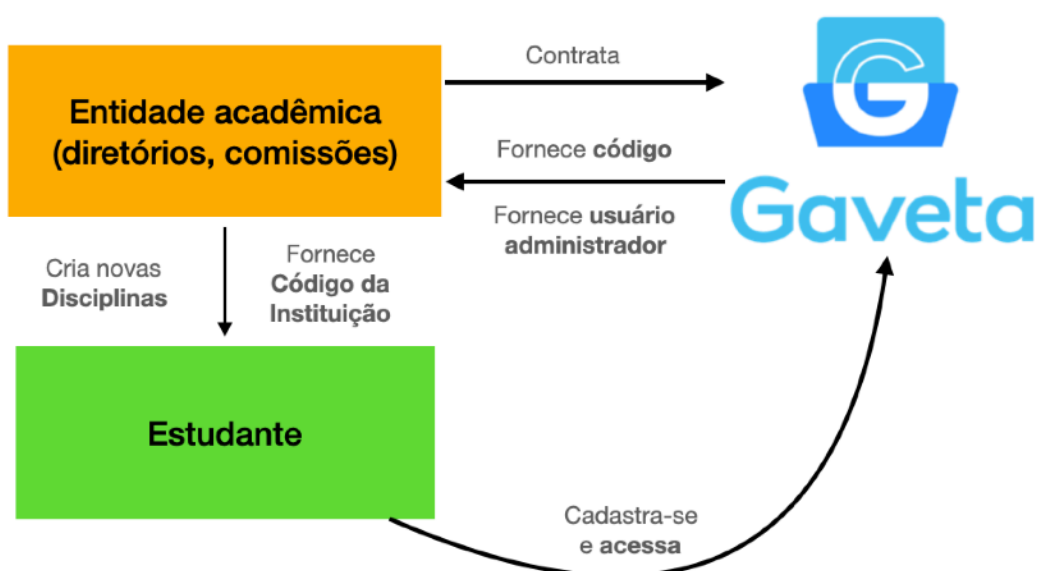
Essas entidades contratariam o serviço, registrando uma nova instituição de ensino. Esse registro gera um código único para a instituição. Por exemplo, o Diretório Acadêmico da Computação da UFRGS contrata o Gaveta e cria a instituição de ensino cujo código é UFRGS.Inf. A partir desse momento, ela recebe um usuário de administrador, capaz de criar novas disciplinas, e registra as disciplinas dos cursos de Ciência e Engenharia da Computação.

Os alunos desses cursos recebem da entidade contratante o código da instituição (no exemplo anterior, UFRGS.Inf) e o informa no momento do cadastro, a fim de ter visíveis as matérias que são de seu interesse. Os alunos não são usuários administradores, ou seja, não poderão criar novas disciplinas, apenas segui-las e criar novas postagens e comentários.

As entidades administradoras, além de poder criar disciplinas, têm o poder de remover postagens e comentários. Isso é bastante útil caso haja alguma postagem ou comentário com conteúdo sensível ou considerado ofensivo, bem como postagens que têm seus direitos autorais reclamados por outrem.

Nesse trabalho, o foco é o desenvolvimento do MVP (Mínimo Produto Viável) da aplicação, podendo ainda ser evoluído e ajustado em relação às suas regras e funcionalidades. A análise de mercado e precificação estão fora do escopo do trabalho, mas são questões que precisam ser estudadas para avaliar a viabilidade econômica da solução proposta. A Figura 5.4 esquematiza um possível fluxo de contratação do Gaveta.

Figura 5.4 - Contratação do Gaveta



Fonte: o autor

6 AVALIAÇÃO

Além das verificações típicas do processo de desenvolvimento de *software*, como testes unitários e testes de sistema, foi realizada uma validação inicial com usuários reais. O grupo de usuários que avaliou o Gaveta era composto por cinco pessoas, com idades entre 20 e 27 anos, dos quais dois tinham ensino superior completo e três ensino superior em andamento.

Uma vez gerada uma versão do Gaveta, ela foi instalada em um dispositivo físico e apresentada para esse grupo de usuários. Foram propostas cinco atividades que simulassem a jornada de um usuário em alguns fluxos de execução e, posteriormente, foram feitas proposições que deveriam ser endossadas ou refutadas pelos usuários, permitindo medir sua satisfação com o produto. Essas atividades foram definidas buscando atingir situações de uso mais frequentes do Gaveta, ou seja, as que seriam mais repetidas pelos usuários no dia-a-dia, como postagens e comentários. O fluxo de cadastro foi incluso nas avaliações porque é um fluxo imprescindível para acessar a aplicação, e caso esse fluxo não tenha uma boa avaliação, potenciais usuários podem perder o interesse em se cadastrar na aplicação. Além disso, os usuários não receberam nenhum tipo de treinamento, a fim de simular um usuário real que fizesse o *download* do aplicativo direto da loja e já o utilizasse.

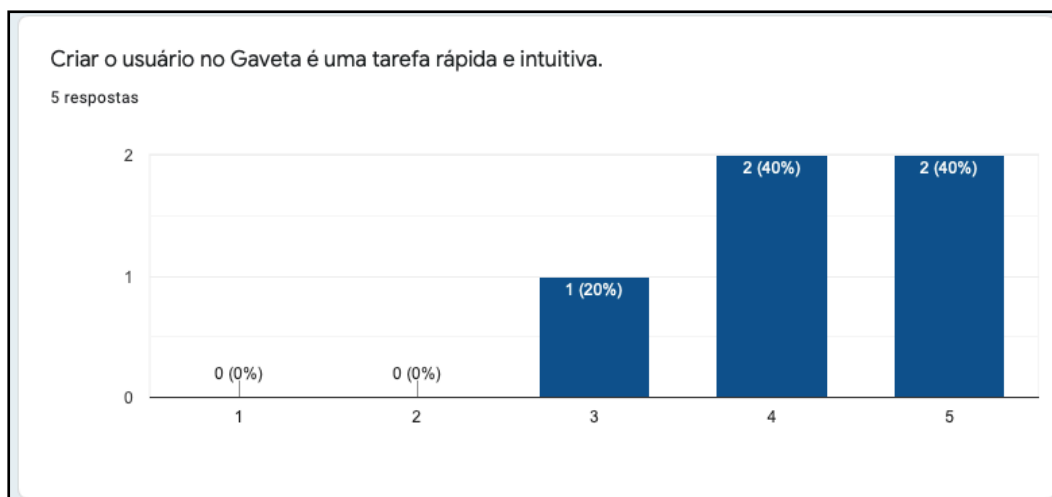
Cada uma dessas perguntas pôde ser respondida com uma nota de 1 a 5 onde 1 significa “Discordo Totalmente” e 5 “Concordo Totalmente”, e os resultados serão discutidos nesse capítulo. As perguntas foram feitas através de um formulário como mostra a Figura B1.

6.1 Avaliação do fluxo de cadastro

Nessa validação, foi entregue ao usuário o aplicativo em sua tela de entrada, como na Figura 5.3.1, e foi pedido para que ele se registrasse no sistema. Quando o fluxo chegou ao final, o usuário respondeu à seguinte proposição: “Criar o usuário no Gaveta é uma tarefa rápida e intuitiva.”

- 20% dos usuários responderam indiferente
- 40% dos usuários responderam que concordam
- 40% dos usuários responderam que concordam totalmente

Figura 6.1 - Resultados da avaliação do fluxo de cadastro



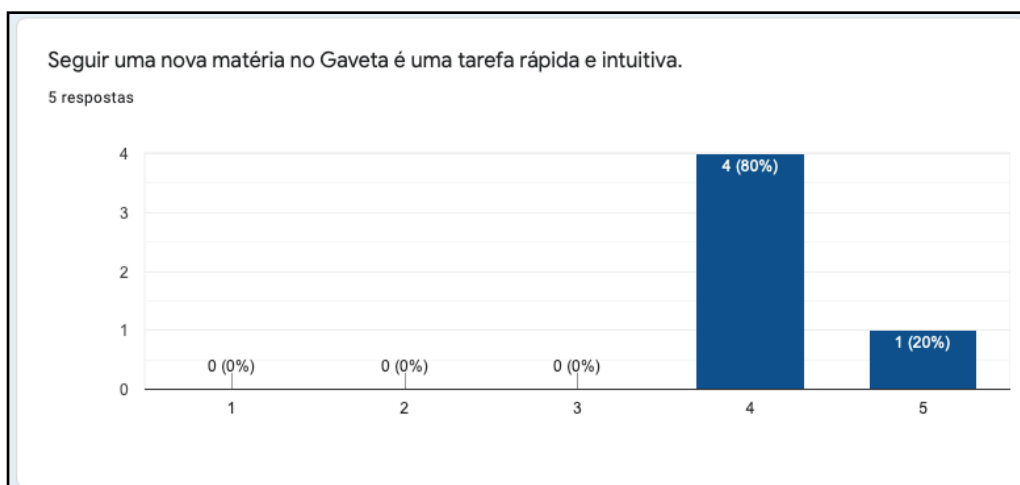
Fonte: o autor

6.2 Avaliação do fluxo de seguir nova matéria

Nessa validação, foi entregue ao usuário o aplicativo em sua tela inicial, como na Figura 5.3.5, mas em seu estado vazio, ou seja, sem nenhuma matéria sendo seguida. Então, foi proposto que o usuário seguisse uma nova matéria par acompanhá-la com o Gaveta. Ao fim do fluxo, foi feita a seguinte proposição: “Seguir uma nova matéria no Gaveta é uma tarefa rápida e intuitiva.”

- 80% dos usuários responderam que concordam
- 20% dos usuários responderam que concordam totalmente

Figura 6.2 - Resultados da avaliação do fluxo de seguir nova matéria



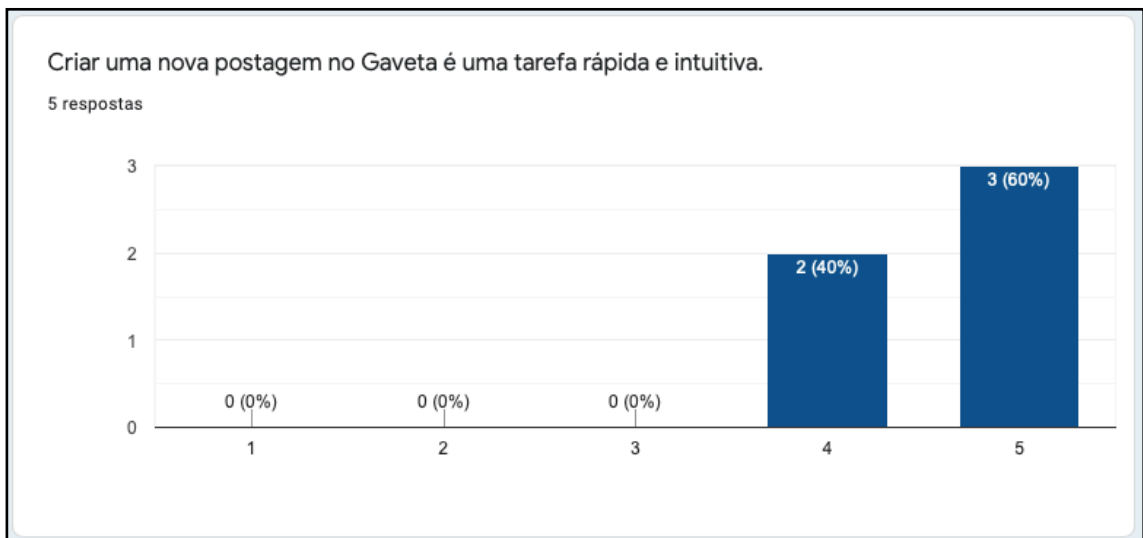
Fonte: o autor

6.3 Avaliação do fluxo de criar nova postagem

Nessa validação, foi entregue ao usuário o aplicativo em sua tela inicial, como na Figura 5.3.5, logado em um usuário que já seguia algumas matérias. Então, foi proposto que o usuário escolhesse uma matéria ao acaso e realizasse uma nova postagem nela. Ao final do fluxo, foi feita a seguinte proposição: “Criar uma nova postagem no Gaveta é uma tarefa rápida e intuitiva.”

- 40% dos usuários responderam que concordam
- 60% dos usuários responderam que concordam totalmente

Figura 6.3 - Resultados da avaliação do fluxo de criar nova postagem



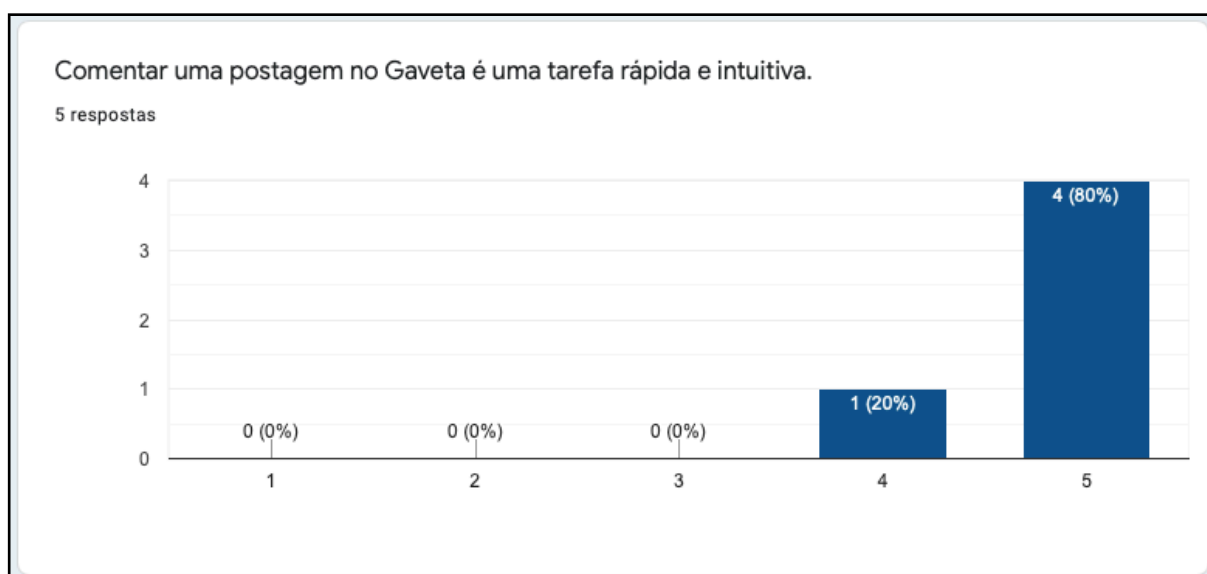
Fonte: o autor

6.4 Avaliação do fluxo de criar novo comentário em postagem

Nessa validação, novamente foi entregue ao usuário o aplicativo em sua tela inicial, como na Figura 5.3.5, logado em um usuário que já seguia algumas matérias. Então, foi proposto que o usuário escolhesse uma matéria, acessasse suas postagens, e escolhesse ao acaso uma postagem e realizasse um novo comentário nela. Ao final do fluxo, foi feita a seguinte proposição: “Comentar uma postagem no Gaveta é uma tarefa rápida e intuitiva.”

- 20% dos usuários responderam que concordam
- 80% dos usuários responderam que concordam totalmente

Figura 6.4 - Resultados da avaliação do fluxo de criar novo comentário



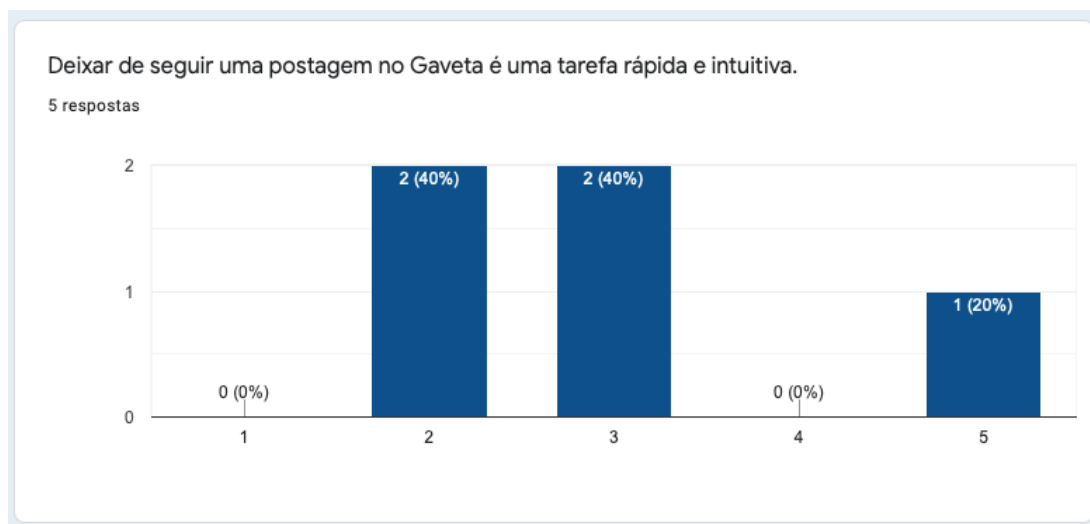
Fonte: o autor

6.5 Avaliação do fluxo de deixar de seguir matéria

Nessa validação, foi entregue ao usuário o aplicativo em sua tela inicial, como na Figura 5.3.5, logado em um usuário que já seguia algumas matérias. Então, foi proposto que o usuário escolhesse uma matéria ao acaso e deixasse de segui-la. Ao final da atividade, foi feita a seguinte proposição: “Deixar de seguir uma postagem no Gaveta é uma tarefa rápida e intuitiva.”

- 40% dos usuários responderam que discordam
- 40% dos usuários responderam indiferente
- 20% dos usuários responderam que concordam totalmente

Figura 6.5 - Resultados da avaliação o fluxo de deixar de seguir matéria



Fonte: o autor

6.6 Considerações das avaliações

Após as avaliações, os usuários tiveram a oportunidade de expressar sua opinião sobre o aplicativo e os fluxos que percorreram. Houve uma ponderação a respeito do fluxo de cadastro pois, segundo o usuário, apesar de o fluxo ser intuitivo, ele é longo e poderia ser mais objetivo. A sugestão foi de pedir, no fluxo de cadastro, apenas as informações estritamente necessárias para registrar o usuário e, então, permitir que as outras informações sejam alteradas quando o usuário quiser na área logada.

Outro ponto bastante comentado foi o fluxo de deixar de seguir matéria. Os usuários tiveram dificuldade em encontrar o gesto que realiza tal ação, como mostrado na Figura 5.3.5.1. Intuitivamente, ao serem solicitados a deixar de seguir uma disciplina, a maioria dos usuários abriu a página da disciplina e procurou algum botão por lá, relacionando essa ação a uma disciplina, e não à lista de disciplina. Apenas um único usuário conseguiu, na primeira tentativa, deixar de seguir uma disciplina, o que evidencia um ponto de melhoria na aplicação.

Os fluxos de seguir disciplinas, criar postagens e comentários foram bem avaliados pelos usuários. Os comentários foram de que eles eram fluxos intuitivos e bastante semelhante ao de outras aplicações que estão acostumados a usar, então não houve necessidade de ficar buscando na tela qual seria o botão correto para utilizar em cada caso. Um dos usuários sugeriu que os botões de composição, localizados no canto superior direito das telas mostradas nas Figuras 5.3.7 e 5.3.8 fossem substituídos por uma versão textual que indicasse, por exemplo, “Nova Postagem”, ao invés de apenas o ícone de composição padrão do sistema.

7 CONCLUSÃO

Esse trabalho propôs uma solução para o compartilhamento de conhecimento e arquivos entre os alunos do meio acadêmico. Essa solução se materializa através de uma aplicação móvel que fornece um ambiente organizado e focado em discussões acadêmicas, permitindo que os alunos vejam as matérias pertinentes ao seu curso e possa discutir com seus colegas de sala, como e estivessem em uma situação do cotidiano.

Foram apresentadas as técnicas de Engenharia de Software que serviram de base para a concepção da aplicação, desde a parte de arquitetura até a metodologia de desenvolvimento que norteou o dia-a-dia desse trabalho. Também foi apresentada a instanciação dessa base de conhecimento sobre o problema que motivou esse projeto, resultando em um protótipo funcional do produto Gaveta.

A solução foi estruturada em uma aplicação móvel administrada por uma entidade acadêmica, responsável por cadastrar novas disciplinas e fornecer aos alunos o código de acesso. Os usuários (alunos) têm, então, a possibilidade de seguir disciplinas, criar novas postagens de discussões - inclusive com anexos de mídia - e comentar nas postagens de seus colegas, a fim de promover um ambiente de discussão fácil e interativo. Essa solução diminui o problema da replicação de arquivos, uma vez que novos conteúdos serão postados no ambiente específico que lhes couberem, e também resolve o problema da manutenção por usuários, uma vez que os potenciais contratantes são entidades “permanentes” em uma instituição acadêmica.

Foi gerado um protótipo representando o MVP que contempla as funcionalidades necessárias para as interações entre os usuários, como seguir disciplinas, criar postagens e demais funcionalidades descritas no Anexo B. Esse protótipo possui algumas limitações, como a necessidade de uma área de busca a partir de *tags* e filtros, visando diminuir o tempo que um aluno levaria para encontrar determinado tópico de discussão.

A avaliação com usuários descrita no Capítulo 6 mostrou que o protótipo desenvolvido foi satisfatório, apesar de ter evidenciado a necessidade de ajustes, como no fluxo de “deixar de seguir matéria”. Essa avaliação foi bastante importante para a conclusão desse projeto, já que foi a oportunidade de validar tudo o que foi estudado e implementado durante os últimos meses. Além disso, a opinião dos usuários serviu como um norteador para melhorias e ajustes futuros.

Como sugestão de trabalhos futuros, seria interessante colocar em prática o *feedback* dos usuários, reduzindo as etapas do cadastro ou até mesmo prover a opção de cadastro através de provedores, como Google, Facebook e Apple. Além dos ajustes de usabilidade e interface, também seria interessante prover um mecanismo de filtros e pesquisa, que agilize ainda mais a experiência do usuário dentro do Gaveta. Também seria bastante útil um mecanismo de detecção de duplicatas através da análise do conteúdo dos arquivos. Por último, a sugestão de implementação em demais plataformas, como Android e Web, a fim de ampliar ainda mais o acesso dos alunos ao ambiente do Gaveta.

REFERÊNCIAS

AMPUDIA, R. **Brasil lidera número de smartphones conectados na América Latina**. 2017. Disponível em: <www1.folha.uol.com.br/mercado/2017/09/1917782-brasil-lidera-numero-de-smartphones-conectados-na-america-latina.shtml>. Acesso em: 20 out. 2020.

MVC. **Model–view–controller**. 2010. Disponível em: <<https://en.wikipedia.org/wiki/Model–view–controller>>. Acesso em 10 out. 2020

ORLOV, B. **iOS Architecture Patterns**. 2015. Disponível em <<https://medium.com/ios-os-x-development/ios-architecture-patterns-ecba4c38de52>>. Acesso em: 10 set. 2020.

KUDINOV, O. **Clean Architecture and MVVM on iOS**. 2019. Disponível em <<https://tech.olx.com/clean-architecture-and-mvvm-on-ios-c9d167d9f5b3>>. Acesso em 8 mai. 2020.

SANTA, M. **MVVM-C with Swift**. 2017. Disponível em <<https://marcosantadev.com/mvvmc-with-swift/>>. Acesso em 26 set. 2020.

ÁGIL, D. **Scrum**. 2014. Disponível em: <<https://www.desenvolvimentoagil.com.br/scrum/>>. Acesso em 13 nov. 2020.

MOLOCHKO, A. **Clean Architecture : Part 2 – The Clean Architecture**. 2017. Disponível em <<https://cosp.net/blog/software-architecture/clean-architecture-part-2-the-clean-architecture/>>. Acesso em 10 out. 2020.

GITHUB. **Hello world**. 2020. Disponível em: <<https://guides.github.com/activities/hello-world>>. Acesso em 18 set. 2020.

INC, A. **Swift.org and Open Source**. 2020. Disponível em: <<https://swift.org/about/#swiftorg-and-open-source>>. Acesso em 18 set. 2020.

LLC, G. **Cloud Firestore**. 2020. Disponível em: <<https://firebase.google.com/docs/firestore?hl=pt-br>>. Acesso em 18 set. 2020

EXCHANGE. **Stack Exchange**. 2020. Disponível em: <https://en.wikipedia.org/wiki/Stack_Exchange>. Acesso em 05 dez. 2020

ANEXO A - HISTÓRIAS DE USUÁRIO

Tabela A1 - Login de Usuário

Requisitos	Ter baixado o aplicativo Gaveta
Personagem	Usuário
Jornada	<ul style="list-style-type: none"> - A partir da tela de entrada, o usuário deverá clicar no botão “Entrar”. - O usuário deverá, então, inserir o e-mail que informou no momento do cadastro - O usuário deverá, então, inserir a sua senha
Encerramento	<ul style="list-style-type: none"> - Se a combinação de usuário e senha existir no banco de dados e estiver correta, o usuário deverá ser levado à área logada. Se a combinação falhar, o sistema deverá mostrar uma mensagem informando o insucesso da operação.

Fonte: o autor

Tabela A2 - Seguir disciplinas

Requisitos	O usuário precisa estar autenticado, através do fluxo de cadastro ou de login.
Personagem	Usuário
Jornada	<ul style="list-style-type: none"> - Na tela inicial, o usuário deverá visualizar um botão para seguir novas matérias - Ao acioná-lo, deverá ver uma tela com a lista de matérias disponíveis para a sua instituição de ensino.
Encerramento	<ul style="list-style-type: none"> - Ao selecionar uma matéria, ela deve ser incluída na lista de matérias que o usuário segue.

Fonte: o autor

Tabela A3 - Favoritar disciplinas

Requisitos	O usuário precisa estar autenticado, através do fluxo de cadastro ou de login.
Personagem	Usuário

Jornada	<ul style="list-style-type: none"> - Na tela inicial, o usuário deverá visualizar a lista de matérias que segue. - Deverá ter, para cada matéria da lista, a opção de torná-la favorita, através de um botão.
Encerramento	<ul style="list-style-type: none"> - Ao tocar neste botão, a disciplina deverá ser marcada como favorita, e inserida na lista de disciplinas favoritas

Fonte: o autor

Tabela A4 - Visualizar lista de postagens de uma disciplina

Requisitos	O usuário precisa estar autenticado, através do fluxo de cadastro ou de login.
Personagem	Usuário
Jornada	<ul style="list-style-type: none"> - Na tela inicial, o usuário deverá visualizar a lista de disciplinas que segue - Ao tocar na linha correspondente a uma disciplina, deverá ver outra tela com os dados da disciplina e com a lista de postagens associadas à ela. - As informações da disciplina deverão contemplar seu título e código. - Cada postagem da lista deverá ter as informações do autor (foto, nome e nome de usuário), o conteúdo da postagem em texto, a data de criação e deverá exibir um símbolo indicando que há anexos associados àquela postagem, caso existam.

Fonte: o autor

Tabela A5 - Criar nova postagem em uma disciplina

Requisitos	O usuário precisa estar autenticado, através do fluxo de cadastro ou de login.
Personagem	Usuário

Jornada	<ul style="list-style-type: none"> - Na tela de disciplina, o usuário deverá ver um botão para a inclusão de nova postagem. - Ao acioná-lo, deverá ver uma tela de composição, na qual poderá inserir o conteúdo textual de sua postagem e, opcionalmente, inserir arquivos. - Para a inserção de arquivos serão permitidos os formatos de documentos (como pdf, txt, doc, docx, pages) e também de mídia (como mp3, mpeg, jpg, bmp). - Caso o usuário opte por inserir arquivos, deverá ver uma mensagem informando a necessidade dos direitos autorais sobre aquele conteúdo. - O usuário deverá visualizar um botão "Enviar" e, ao acioná-lo, o sistema deverá executar o envio dessa postagem.
Encerramento	<ul style="list-style-type: none"> - Se o envio for bem-sucedido, o usuário deverá ser levado para a tela anterior, que contemplará sua nova postagem. Se o envio for mal-sucedido, deverá ser mostrada uma mensagem que indique o insucesso.

Fonte: o autor

Tabela A6 - Visualizar postagem de uma disciplina

Requisitos	O usuário precisa estar autenticado, através do fluxo de cadastro ou de login.
Personagem	Usuário
Jornada	<ul style="list-style-type: none"> - Na tela de detalhe de uma matéria, onde vê a lista de postagens, o usuário toca em uma postagem. - Deverá, então, ver uma tela de detalhe de postagem, que contemplará em destaque o conteúdo da postagem (informações do autor, conteúdo, e lista de anexos, quando houver) e uma lista de comentários relacionados àquela postagem

Fonte: o autor

Tabela A7 - Criar novo comentário em uma postagem de uma disciplina

Requisitos	O usuário precisa estar autenticado, através do fluxo de cadastro ou login e estar seguindo uma matéria. Essa matéria precisa ter ao menos uma postagem.
------------	--

Personagem	Usuário
Jornada	<ul style="list-style-type: none"> - Na tela de detalhe de postagem, o usuário deverá visualizar um botão para novo comentário - Ao acioná-lo, deverá mandar ver a tela de composição de comentário, na qual poderá inserir apenas conteúdo textual. Por definição de negócio, anexos não são contemplados em comentários. - A tela de composição deverá possuir um botão de "Enviar" para que o comentário seja submetido. - Quando o botão "Enviar" for acionado, o sistema deverá submeter o novo comentário.
Encerramento	<ul style="list-style-type: none"> - Se bem-sucedido, o usuário deverá ser levado para a tela anterior que já contemplará seu novo comentário. - Se mal-sucedido, o sistema deverá mostrar uma mensagem indicando o insucesso.

Fonte: o autor

Tabela A8 - Deixar de seguir uma disciplina

Requisitos	O usuário precisa estar autenticado, através do fluxo de cadastro ou login e estar seguindo uma matéria.
Personagem	Usuário
Jornada	<ul style="list-style-type: none"> - Na listagem de matérias na tela inicial, o usuário deverá ter a opção de deixar de seguir uma matéria. - Quando essa opção for acionada, a lista de matérias seguidas pelo usuário deverá ser atualizada, e essa matéria deverá ficar novamente disponível para ser seguida caso o usuário deseje.

Fonte: o autor

Tabela A9 - Remover uma disciplina da lista de favoritos


Requisitos	O usuário precisa estar autenticado, através do fluxo de cadastro ou login e ter marcado como favorita ao menos uma das matérias que está seguindo.
Personagem	Usuário

Jornada	<ul style="list-style-type: none">- Na listagem de favoritos na tela inicial, o usuário deverá ver um botão que desfça a ação de favorito.- Ao acionar o botão, a matéria deverá ser removida da lista de favoritas. A lista de matérias que o usuário segue não deverá ser afetada por essa ação.
---------	---

Fonte: o autor

ANEXO B - FORMULÁRIO DE AVALIAÇÃO

Figura B1 - Formulário de avaliação



Gaveta
Avaliação de usuário do aplicativo Gaveta. 2020.

Criar o usuário no Gaveta é uma tarefa rápida e intuitiva.

1 2 3 4 5

Discordo totalmente Concordo Totalmente

Seguir uma nova matéria no Gaveta é uma tarefa rápida e intuitiva.

1 2 3 4 5

Discordo totalmente Concordo totalmente

Criar uma nova postagem no Gaveta é uma tarefa rápida e intuitiva.

1 2 3 4 5

Discordo totalmente Concordo totalmente

Comentar uma postagem no Gaveta é uma tarefa rápida e intuitiva.

Fonte: o autor