Lucas Pavan Garieri
Pedro Henrique Rodrigues de Viveiros
Victor de Almeida Santana

Desenvolvimento de uma solução para monitoramento de presença e engajamento para o Cursinho Popular da Escola Politécnica da Universidade de São Paulo

São Paulo, SP

2024

Lucas Pavan Garieri
Pedro Henrique Rodrigues de Viveiros
Victor de Almeida Santana

Desenvolvimento de uma solução para monitoramento de presença e engajamento para o Cursinho Popular da Escola Politécnica da Universidade de São Paulo

Trabalho de conclusão de curso apresentado ao Departamento de Engenharia de Computação e Sistemas Digitais da Escola Politécnica da Universidade de São Paulo para obtenção do Título de Engenheiro.

Universidade de São Paulo – USP

Escola Politécnica

Departamento de Engenharia de Computação e Sistemas Digitais (PCS)

Orientador: Prof. Dr. Fábio Levy Siqueira

São Paulo, SP

2024

Catalogação-na-publicação

Garieri, Lucas
Desenvolvimento de uma solução para monitoramento de presença e engajamento para o Cursinho Popular da Escola Politécnica da Universidade de São Paulo / L. Garieri, P. Viveiros, V. Santana -- São Paulo, 2024.
75 p.

Trabalho de Formatura - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1.Sistemas Computacionais 2.Software 3.Aplicativos móveis I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t. III.Viveiros, Pedro IV.Santana, Victor

O futuro pertence àqueles que acreditam na beleza de seus sonhos.

Agradecimentos

Os agradecimentos principais são direcionados aos professores do curso de Engenharia de Computação, pelo conhecimento ministrado e passado com tanta dedicação. Ao nosso orientador, Prof. Dr. Fábio Levy Siqueira, pelo direcionamento, pelo acompanhamento e pelas intervenções, sem as quais este trabalho não seria possível. Ao Cursinho Popular da Escola Politécnica da USP (CP), por aceitar ser parte deste projeto e por inspirar, não apenas nós, mas centenas de alunos com sua missão. Ao Enzo Tassini do CP, que ficou responsável por acompanhar este projeto, juntamente ao cursinho, e realizou um trabalho excepcional. Aos nosso amigos, que passaram por todas as dificuldades juntamente a nós e tornaram nossos dias melhores e a todos que contribuíram para o desenvolvimento deste trabalho.

Lucas Pavan Garieri

Ao meu pai, Alexandre, que me inspira todos os dias a ser melhor e sem o qual eu não teria chegado até aqui. À minha mãe, Daniela, que me criou e me mostrou o que é ser forte e resiliente e continua me mostrando o caminho, mesmo depois de anos. Aos meus avós, Neusa e Alcino, que me mostram todos os dias o que é amor e a quem eu devo tudo que tenho. E, por fim, aos meus avós, Sílvia e Carlos, que me ensinaram tudo que sei e me mostraram, desde pequeno, como seguir meus sonhos.

Pedro Henrique Rodrigues de Viveiros

Aos meus pais, Alex e Eliana, por sempre apoiarem em tudo que dizia respeito a minha formação como engenheiro. Aos meus colegas de gestão do CEE, que me proporcionaram momentos e aprendizados para a vida. Aos meus colegas da Inspira, por me ensinarem e alegrarem todos os dias. E por fim, mas não menos importante, ao Henrique, que sempre pude contar como fonte de calmaria e suporte.

Victor de Almeida Santana

Aos meus pais, Daniel e Vânia, por nunca terem medido esforços para me criarem com a melhor estrutura que eu poderia ter. Ao meu irmão, Vinícius, por ser o melhor amigo e o maior presente que Deus me deu. À minha avó, Dalvinisia que descansa no Senhor, mas que em vida sempre me impulsionou a ser um homem melhor. E sobretudo, ao meu Deus e Senhor Jesus Cristo, sem o qual eu não teria nem sonhado em chegar até aqui.

Resumo

O Cursinho Popular da Escola Politécnica da USP (CP) busca, por meio de aulas gratuitas de conteúdos dos vestibulares, auxiliar estudantes de baixa renda a entrarem em universidades públicas de qualidade.

Buscando conhecer melhor a realidade dos alunos impactados, o Cursinho da Poli tem tomado iniciativas para monitorar o seu engajamento, através da coleta de dados relativos à presença dos alunos nas aulas. Além disso, o cursinho também visa minimizar a evasão e oferecer auxílios de permanência para os alunos. Porém, por utilizarem listas de assinaturas, a compilação desses dados é um processo demorado e sujeito a atrasos e erros.

Assim, o objetivo deste trabalho é desenvolver um sistema em software que automatiza tanto a coleta de presença como a compilação de dados. Para isso, são usados métodos de Design Thinking para idealização da solução e Métodos Ágeis para o desenvolvimento. Desse modo, os diretores da entidade poderão obter tais métricas em tempo real e dedicarem-se a outras atividades mais prioritárias.

Como resultado, o sistema demonstra ser capaz de substituir as listas de presença, reduzindo o tempo necessário para a sua contabilização e eliminando atrasos no processo. Além disso, ele proporcia aos alunos maior acessibilidade às métricas de frequência, reforçando seu potencial para minimizar a evasão e apoiar a gestão do CP. Ainda, professores e alunos demonstraram satisfação com a solução, avaliando positivamente a experiência, com notas médias superiores a 4,0, numa escala de 1 a 5, para aspectos como intuitividade, interface e curva de aprendizado.

Palavras-chave: software; presença; educação; design thinking; cursinho popular.

Abstract

The Polytechnic School of USP's Pre-College Preparatory Course (Cursinho Popular da Poli in Portuguese, or CP) aims to help low-income students enter public universities with high-quality education by offering free classes on the contents of the entrance exams.

In pursuit of a deeper understanding of students' realities, CP prioritizes monitoring overall class engagement and gathering attendance data through traditional paper attendance lists. It also aims to minimize class evasion and provide assistance to students. However, relying on these conventional methods makes compiling this data labor-intensive and subject to delays and mistakes.

Therefore, this work aims to produce a software system that automates attendance collection and data compilation. We used Design Thinking and Design Sprint to idealize the solution and Agile Methods for development. Therefore, the organization's directors can obtain these metrics in real-time and devote themselves to other higher-priority activities.

As a result, the system proves to be capable of replacing traditional attendance lists, reducing the time required for the process and eliminating delays. Furthermore, it provides students with greater accessibility to attendance metrics, reinforcing its potential to minimize dropout rates and support the CP's management. Additionally, both teachers and students expressed satisfaction with the solution, positively evaluating their experience, with average scores exceeding 4.0 out of 5.0 in aspects such as intuitiveness and interface design.

Keywords: attendance; software; education; design thinking; pre-college preparatory course.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

CP	Cursinho Popular da Escola Politécnica da Universidade de São Paulo
TDIC	Tecnologias Digitais da Informação e Comunicação
USP	Universidade de São Paulo
POV	Point of View
HMW	How Might We?
MVT	Model-View-Template
MVP	Mínimo Produto Viável
AWS	Amazon Web Services

Sumário

1 Introdução

O uso de Tecnologias Digitais de Informação e Comunicação (TDIC) na área da educação vem crescendo anualmente, gerando a possibilidade de aplicá-las para facilitar o processo de aprendizagem. Mesmo que a pandemia de Covid-19 tenha aumentado abruptamente durante os anos de 2020 e 2021 o número de instituições de ensino que utilizam alguma plataforma online, um tipo de TDIC, esse número ainda foi maior em 2022 quando comparado a 2019 (CETIC, 2022). Além disso, seu uso para automatizar fluxos dentro das instituições de ensino pode facilitar o cumprimento de atividades de maneira mais rápida, segura e intuitiva.

1.1 Motivação

O presente trabalho buscou entender o cenário do Cursinho Popular da Poli (CP) e propor uma plataforma digital para automatizar o processo de presença dos alunos. O CP é um curso preparatório para vestibulares que visa fornecer uma educação de qualidade para estudantes de baixa renda que não têm acesso facilitado a ela. Assim, são oferecidas aulas noturnas ministradas por estudantes universitários da Escola Politécnica da Universidade de São Paulo, tanto presenciais quanto *online*. Para garantir que os alunos estejam acompanhando o curso, sua presença é aferida a fim de minimizar a evasão e prover auxílios de permanência.

No entanto, o processo é feito a partir de uma lista de chamada assinada que precisa ser passada manualmente para o sistema. Segundo o coordenador discente do cursinho, esse fluxo atualmente gera um atraso de quase 2 meses entre a assinatura da lista pelos alunos e a inserção desses dados no sistema. Além disso, esse atraso também gera a necessidade de armazenar as listas por um grande período de tempo, aumentando a possibilidade de perda e produzindo material a ser descartado posteriormente.

Juntamente a isso, a presença das aulas *online* não é aferida por dificuldade de integração dos sistemas utilizados, fazendo com que o cursinho não tenha esse controle. Por fim, nenhum dos modelos de ensino apresenta método eficaz para que os alunos acompanhem a própria presença, sendo necessário pedir tais dados a um coordenador sempre que necessário.

1.2 Objetivos

O objetivo do projeto é a criação de um ambiente de controle de presença durante as aulas do Cursinho da Poli. Dessa forma, busca-se automatizar o processo, deixá-lo mais seguro e consistente, e fornecer um ambiente em que o cursinho possa gerenciar os dados dos alunos matriculados de maneira centralizada.

Para isso é necessário que o ambiente verifique a presença dos alunos de maneira automática, retirando do CP a responsabilidade de passar manualmente a lista de chamada para o sistema. Além disso, o ambiente deve apresentar dados, baseados na taxa de presença e participação, que lhes permitam analisar a taxa de evasão e seus fatores, facilitando o contato com os alunos e a adesão às atividades.

Dessa forma, é mostrado um cenário que poderia se beneficiar do uso de TDIC para automatização do processo de presença, diminuindo tanto o atraso quanto o descarte de material. Além disso, existe a possibilidade de criar um maior contato com o aluno, permitindo que ele acompanhe sua presença sem que haja necessidade da intervenção do coordenador.

1.3 Justificativa

A proposta deste trabalho é modernizar a monitoração de presença no âmbito escolar. No entanto, em razão de sua abordagem atrelada a uma organização social, o projeto apresenta especificidades que o diferencia de aplicações similares. Diversos sistemas de monitoramento já existentes no mercado costumam demandar recursos ou infraestrutura custosas, tornando-as inviáveis para organizações de cunho social. Considerando o contexto do Cursinho Popular da Escola Politécnica da USP (CP), o sistema aborda duas principais problemáticas: a necessidade de minimizar os processos manuais para o registro de presença e a dificuldade de mitigar a evasão escolar, que impacta diretamente o desempenho acadêmico e a gestão pedagógica.

Por meio das duas interfaces propostas (a *web* e a *mobile*), visa-se facilitar o registro de presença de maneira descentralizada, tirando a responsabilidade da gestão do CP de compilar os dados manualmente. Essa metodologia elimina a necessidade de dispositivos específicos como biometria ou leitores de códigos, comumente encontrados em sistemas físicos de monitoramento, mantendo a simplicidade de operação e escalabilidade, além de abarcar as diferentes turmas (presencial e *online*) do cursinho.

Deste modo, ao oferecer uma solução economicamente acessível e tecnicamente eficaz a uma comunidade educacional de grande impacto social, este projeto se diferencia de soluções comerciais já existentes. Pode-se considerar também o ganho a partir da modernização de processos administrativos em organizações educacionais com recursos

limitados, promovendo a inclusão tecnológica nesse meio.

Vale destacar que será possível entender melhor, na prática, como é um projeto que possui um cliente real, suas características e desafios e a importância da criação de soluções personalizadas aos objetivos e experiência do usuário. Dessa forma, também será abordado como a aplicação de conceitos de metodologias como *Design Thinking* (LEWRICK; LINK; LEIFER, 2018) e *Design Sprint* (KNAPP; ZERATSKY; KOWITZ, 2016) contribui para o entendimento dos problemas e das necessidades do usuário.

Por fim, devido ao teor social do Cursinho da Poli a seu objetivo de levar a educação de forma gratuita a pessoas de baixa renda, o projeto ainda contribui com o acesso à educação e à formação de estudantes de baixa renda. Ao melhorar os serviços do CP e possibilitar que a evasão de alunos seja detectada sem atrasos e que o cursinho possa intervir e auxiliá-los no momento certo, o projeto acaba por alavancar sua eficiência.

1.4 Organização do Trabalho

O restante deste trabalho é organizado da seguinte forma: o Capítulo 2 aborda aspectos teóricos e conceituais utilizados para guiar a metodologia aplicada no projeto, tendo em vista os objetivos explicitados. Também é dada uma breve explicação sobre eles.

O Capítulo 3 aborda a metodologia aplicada no projeto, passando pelas ferramentas utilizadas, atividades desempenhadas, requisitos levantados e como foram adaptados os conceitos apresentados no segundo capítulo.

Já o Capítulo 4 apresenta o desenvolvimento do projeto, passando pelos principais fluxos dos sistemas *web* e *mobile*. Também são especificadas as tecnologias escolhidas e a arquitetura do sistema, além da escolha de infraestrutura.

O Capítulo 5 trata do plano de testes elaborado para o projeto e quais foram os resultados obtidos, discorrendo, também, sobre sua análise e sobre a adoção do sistema por parte do Cursinho da Poli.

No Capítulo 6 são apresentados os testes realizados junto ao cursinho com o intuito de colocar o sistema a prova, levantar dados de utilização e coletar *feedback* dos usuários, a fim de mapear novas funcionalidades e rever o funcionamento de outras já estabelecidas.

Por fim, o Capítulo 7 apresenta as considerações finais do presente trabalho, mostrando como os objetivos foram alcançados e quais as principais contribuições. Também são citados possíveis caminhos para a continuação do trabalho.

2 Aspectos Conceituais

Para criar a solução, este trabalho usou uma adaptação, feita pelo grupo, das metodologias *Design Thinking* (LEWRICK; LINK; LEIFER, 2018) e *Design Sprint* (KNAPP; ZERATSKY; KOWITZ, 2016). Ambas são usadas para guiar a escolha e o *design* da solução proposta a partir da identificação dos problemas e das necessidades do usuário. Neste capítulo são apresentadas essas duas abordagens, juntamente com as arquiteturas que guiaram o planejamento e desenvolvimento da solução proposta.

2.1 Métodos de Ideação e Design da Solução

Existem diversos métodos de ideação e *design* de soluções. Neste trabalho, optou-se por usar o *Design Thinking* e o *Design Sprint*.

2.1.1 Design Thinking

O *Design Thinking* consiste em uma série de conceitos e ferramentas usados para identificar problemas e criar ideias centradas na necessidade do usuário. Seu intuito é disponibilizar essas ferramentas, juntamente com condições e recursos que possibilitem ao desenvolvedor entender melhor as necessidades e problemas do usuário e propor possíveis soluções (LEWRICK; LINK; LEIFER, 2018). Logo, fica a critério do profissional adaptá-lo e aplicá-lo da forma que julgar adequada para cada problema.

Este método consiste em cinco etapas principais, com objetivos e recursos diferentes, para ajudar o profissional a identificar as necessidades do usuário. São elas as fases de Empatia, Definição, Ideação, Prototipação e Testagem (LEWRICK; LINK; LEIFER, 2018).

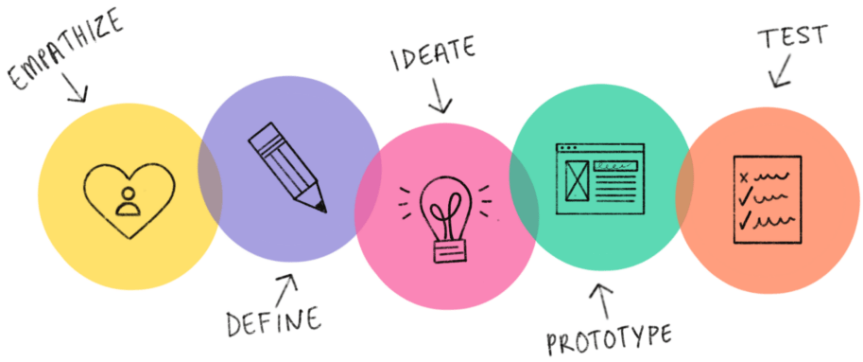

Figura 1 – Etapas do Design Thinking (AHMED, 2023)

2.1.1.1 Empatia

A primeira etapa do processo visa entender melhor quem é o usuário e quais as reais necessidades e objetivos dele. Assim, a finalidade é criar empatia entre o desenvolvedor e o cliente, de forma que a modelagem da solução seja direcionada por informações mais concretas e não apenas por suposições.

2.1.1.2 Definição

Já a etapa de Definição consiste em usar as informações levantadas durante a fase de Empatia para guiar a formulação do problema. O *Design Thinking* defende que, para ter uma boa definição da solução, é necessário ter uma ideia clara do problema a ser resolvido. Dessa forma, é possível entender quais são os desafios enfrentados pelo usuário, quais são suas reais causas e como resolvê-las.

2.1.1.3 Ideação

A etapa de Ideação envolve a definição e a primeira proposta da solução. Assim, dados os problemas levantados na fase anterior, o objetivo é levantar ideias que sejam capazes de resolvê-los. A partir disso, uma solução dada como mais adequada pelo grupo é escolhida e esboçada de forma mais profunda para facilitar o seu entendimento.

2.1.1.4 Prototipação

Já a etapa de Prototipação tem o objetivo de transformar a ideia selecionada em um protótipo testável para determinar sua viabilidade. Dado isso, o intuito é desenvolver um *Minimum Viable Product*, ou MVP, que consiste em um produto simples que comporta as funcionalidades mínimas e vitais para que a ideia seja testada pelo usuário.

2.1.1.5 Testagem

Por fim, a etapa de Testagem refere-se à testagem do MVP pelo usuário. Nessa fase, o objetivo é entender como o usuário interage com o protótipo e coletar mais dados a respeito da solução. Após isso, o grupo pode decidir por seguir com o desenvolvimento da solução ou por reiterar o processo de *Design Thinking*, a depender do resultado.

2.1.1.6 Ferramentas Selecionadas

Cada uma das etapas do *Design Thinking* possui diferentes ferramentas que podem ser usadas. Logo, o grupo selecionou algumas das que mais se encaixavam ao contexto e à proposta do projeto para serem aplicadas. A seguir, são citadas as ferramentas escolhidas.

- *User Interviews*: é um método de pesquisa que consiste em ter contato com grupos de usuários para levantar informações sobre quem eles são, seus objetivos, dores e necessidades, facilitando, posteriormente, a tomada de decisões (FOUNDATION, 2016c).

- *Personas*: são pessoas fictícias criadas, na fase de empatia, para representar os diferentes grupos de usuário mapeados a partir das entrevistas. Elas apresentam informações como características, gostos, desejos, problemas e necessidades de cada usuário, guiando o profissional (FOUNDATION, 2016b).

- *User Journey Map*: são representações visuais de toda a jornada que um usuário percorre para atingir determinado objetivo. Dessa forma, para cada persona criada são mapeadas as principais fases da sua jornada, com quais canais ela tem contato e quais as suas ações, pensamentos, sentimentos e objetivos em cada uma delas (UNIVERSITY, 2024).

- *Point of View* (POV): são afirmações em formato implementável e executável usadas para guiar o processo de definição do problema (DAM; SIANG, 2020). A partir das necessidades do usuário levantadas na etapa de empatia, são formuladas frases, que descrevem quem são usuários, quais as suas necessidades e qual dado ou informação coletada justifica essa afirmação.

- *How Might We* (HMW): são perguntas formuladas, para guiar a etapa de ideação método, no formato "Como poderíamos...?" seguida dos diferentes POV formulados e focam em incentivar a formulação de ideias para resolver o problema (FOUNDATION, 2016a).

2.1.2 Design Sprint

O *Design Sprint* (KNAPP; ZERATSKY; KOWITZ, 2016) é uma aplicação do *Design Thinking*, criada pela Google, com o intuito de responder questões críticas de negócio em um curto período de tempo, mais especificamente cinco dias corridos. O método se mostra útil por restringir e direcionar as ferramentas a serem utilizadas e apresentar instruções claras e concisas do que deve ser feito em cada etapa.

2.1.2.1 Preparação

Consiste na organização do processo como um todo, isto é, agendamento de reuniões, atribuição de papéis, reserva de locais e compra de materiais. Nessa fase é feita a escolha do papel do Facilitador do processo, a pessoa responsável por controlar o tempo e o andamento da *sprint*, e do Decisor, responsável por tomar decisões em última instância.

2.1.2.2 Segunda-feira

Mesclando as duas primeiras fases do *Design Thinking*, o primeiro dia visa definir a meta a ser atingida de acordo com o problema do cliente. Inicialmente, é feito um mapeamento da jornada do usuário e, em seguida, passa-se a diversas reuniões com especialistas para entender como os clientes enxergam o problema tratado, como o sistema funciona atualmente e o que já foi feito para atacar a problemática. Após as entrevistas, é aplicada a técnica de HWM, incluindo as perguntas no mapa do usuário, e uma meta é escolhida com base na importância do cliente e do mapa.

2.1.2.3 Terça-feira

Este dia é focado na elaboração de soluções para os problemas do dia anterior. Inicia-se com a pesquisa e o *benchmark* de soluções já existentes para os problemas levantados, anotando ideias relevantes. Em seguida, essas anotações são usadas para guiar uma sessão de *Brainstorming*. Por fim, a partir das ideias levantadas, é feito um esboço mais elaborado e descritivo da solução, pensando em apresentar para os demais participantes no dia seguinte.

2.1.2.4 Quarta-feira

Dia dedicado para análise das soluções e escolha de quais devem seguir para a fase de prototipação. Primeiramente, os esboços elaborados são dispostos e uma crítica rápida é feita visando tirar dúvidas e levantar objeções ou ideias novas. Então, três ideias são escolhidas pelo Decisor para serem prototipadas e testadas e, assim, cria-se um *storyboard* com a solução detalhada, incluindo desenhos e textos, visando facilitar a prototipação no dia seguinte.

2.1.2.5 Quinta-feira

Dia dedicado inteiramente à prototipação da solução. Inicialmente, é feita uma divisão de tarefas para que grupos menores trabalhem numa parte do sistema, focando na praticidade ao invés da beleza e fidelidade à solução final. Ao final, espera-se testar o protótipo já unificado e procurar e consertar erros.

2.1.2.6 Sexta-feira

Dia de realização das entrevistas e organização de *insights*. É dada uma contextualização ao entrevistado, seguida pela introdução do protótipo e, ao final, pelo resumo, possíveis dúvidas, sugestões ou observações do entrevistado. É esperado que sejam feitas cinco entrevistas ao todo. Ao final do dia, as entrevistas são classificadas em positivas, negativas ou neutras; a meta estabelecida é revisada e decide-se pela realização de outra *sprint*, caso necessário, ou por seguir com o desenvolvimento do produto.

2.2 Práticas e Arquiteturas em Engeharia de Software

A aplicação do *Design Thinking* e do *Design Sprint* permite uma ideia clara da solução a ser implementada e dos requisitos necessários. Isso, somado ao protótipo desenvolvido e ao resultado da fase de testagem, permite que decisões arquiteturais possam ser tomadas com foco nas necessidades do usuário (LEWRICK; LINK; LEIFER, 2018). Assim, optou-se pelas arquiteturas *Clean Architecture*, *Flux Architecture* e MVT (*Model-View-Template*) pela sua compatibilidade e sua complementaridade em tratar possíveis problemas relacionados ao seu uso individual.

2.2.1 Clean Architecture

Concebida por Robert C. Martin, a Arquitetura Limpa ou *Clean Architecture* é um modelo de organização de *software* focado na divisão de responsabilidades através do emprego de camadas, tendo por foco garantir independência de frameworks e bibliotecas externas; testabilidade; independência do banco de dados e da interface escolhida (MARTIN, 2017).

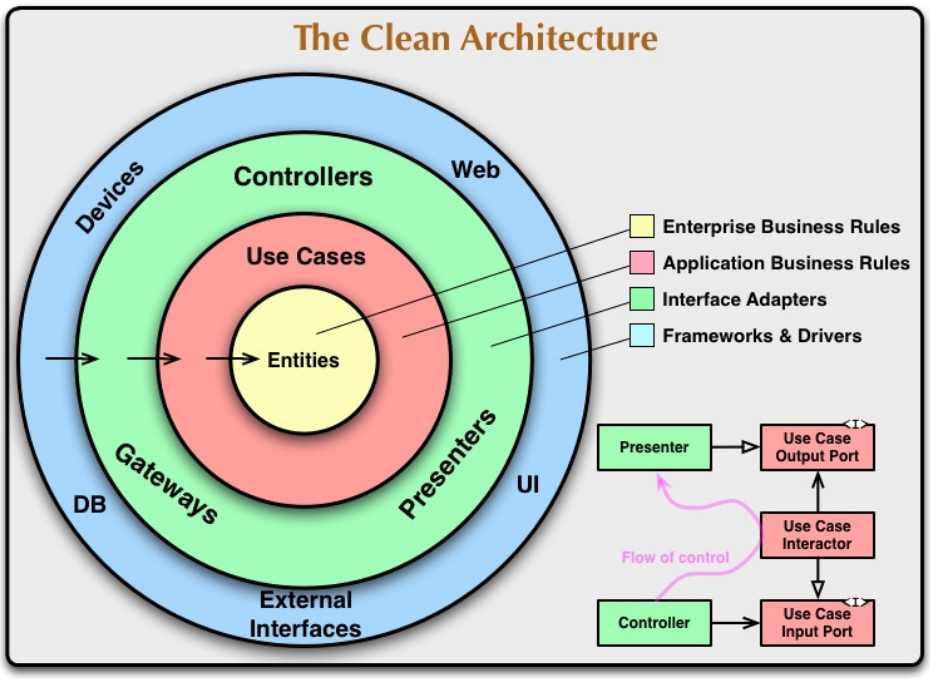

Figura 2 – *Clean Architecture*

A *Clean Architecture* é organizada segundo a Regra de Dependência, na qual define-se que uma camada nunca depende das camadas mais exteriores a ela, apenas das camadas mais internas.

- *Entities*: trata-se da camada mais interna e fundamental do modelo. É onde as regras de negócio são definidas, através da implementação das entidades de domínio. É

responsável por definir as regras e a lógica de mais alto nível da aplicação.

- *Use Case*: camada imediatamente acima das entidades, na qual são definidos os casos de uso da aplicação, responsáveis por orquestrar a maneira como os dados são tratados a partir e para as entidades.
- *Interface Adapters*: trata-se da camada responsável por formatar os dados de modo que seja possível tratá-los devidamente nos casos de uso e nas entidades.
- *Frameworks and Drivers*: camada mais externa da aplicação, onde se encontram o banco de dados, eventuais APIs com as quais a aplicação se comunica, interface de usuário, dispositivos, etc.

2.2.2 Arquitetura MVT

A arquitetura MVT (*Model-View-Template*) é derivada da arquitetura MVC (*Model-View-Controller*), sendo atrelada ao *framework Django*. Seu objetivo é permitir o desenvolvimento de interfaces de usuário, separando a visualização dos dados, a interação do usuário e a interação com o banco de dados a partir da divisão em diferentes camadas (HOLVILLA; MYLLYAHO; PORRES, 2010). Dessa forma, o uso da arquitetura MVT complementa a *Clean architecture* ao especificar como a camada *Frameworks and Drivers* deve ser organizada internamente.

A arquitetura MVT define três componentes principais, sendo eles:

- *Model*: responsável pela estrutura lógica do projeto, funcionando como um intermediário para manipular dados entre o banco de dados e a *View*. Nesta classe são definidos os modelos a serem utilizados no projeto, sendo que, geralmente, cada modelo corresponde a uma tabela no banco de dados utilizado.
- *View*: responsável pela formatação dos dados para que possam ser devidamente dispostos na camada de *template*. Nela ocorre o processamento das requisições e a definição das informações a serem recebidas pelo template.
- *Template*: responsável pela apresentação visual dos dados, define como a interface será renderizada.

2.2.3 Flux Architecture

Proposta pelo Facebook em 2014, *Flux Architecture* é uma abordagem para gerenciar os estados e o fluxo de dados em aplicações *front-end*. Seu principal objetivo é organizar a atualização de estados da aplicação, de forma a garantir maior previsibilidade e escalabilidade, a partir da implementação de um fluxo de dados unidirecional (FACEBOOK, 2014). Assim, a arquitetura se propõe a evitar problemas causados por fluxos de

dados muito complexos ou bidirecionais, normalmente atrelados à dependência de camadas presente na arquitetura MVT.

A arquitetura *flux* define quatro componentes principais, sendo eles:

- *Actions* (ações): são objetos e eventos simples que descrevem algo que deve ocorrer na aplicação. Normalmente contêm um tipo (*type*) e dados associados, referente à atualização de estados da aplicação.
- *Dispatcher* (despachante): é a estrutura responsável por gerenciar o fluxo de dados em uma aplicação *flux*. Seu objetivo é receber as ações e distribuí-las para os estados adequados dentro da aplicação.
- *Stores* (armazenamento): são estruturas que contêm o estado e definem a lógica da aplicação. Elas recebem as ações corretas, a partir do despachante, e atualizam seu estado com base na lógica definida. Seu objetivo é controlar e notificar diferentes componentes da aplicação frente à mudança de estados.
- *Views* (visualizações): são os componentes que consomem os estados de cada armazenamento e reagem à sua mudança. São responsáveis por renderizar a interface do usuário.

O fluxo de dados proposto pela arquitetura *flux* é unidirecional e segue o caminho mostrado na Figura 3. Nele, as ações são disparadas pelas interações do usuário com as *views* e despachadas corretamente para a *store* correspondente, pelo *dispatcher*, causando uma atualização nos estados internos da aplicação e nas *views* que os consomem. Isso garante que a aplicação tenha uma maior previsibilidade, modularidade e escalabilidade, ao mesmo tempo que previne possíveis problemas causados pela arquitetura MVT (FACEBOOK, 2014).

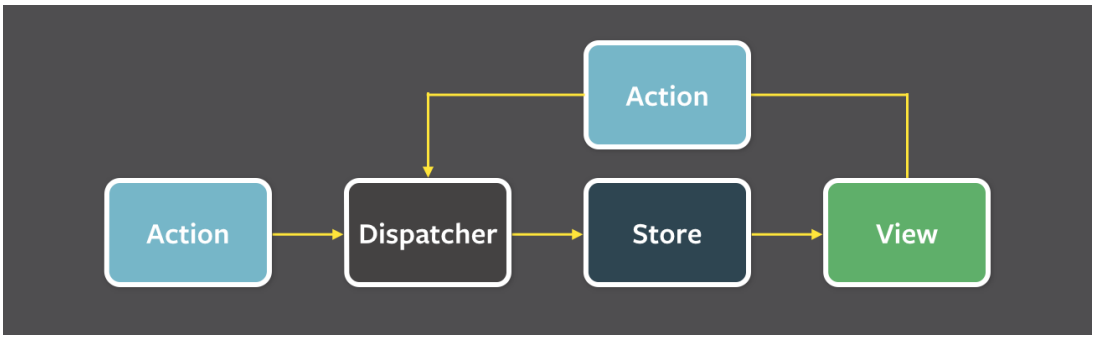

Figura 3 – *Fluxo de dados da Flux Architecture*

2.3 Scrum

O *Scrum* é uma metodologia ágil de gerenciamento de projetos cujo objetivo é auxiliar no desenvolvimento de produtos complexos de forma iterativa. Assim como outras metodologias ágeis, ele foi desenvolvido pensando em conceder flexibilidade e adaptabilidade a mudanças e imprevistos, permitindo que parâmetros do projeto sejam adaptados rapidamente (SCHWABER; SUTHERLAND, 2020).

Dentre as principais características da metodologia, podemos citar a divisão do desenvolvimento em *sprints* curtas e bem definidas, que prevêem, ao seu final, entregas incrementais. Além disso, têm-se a divisão de papéis importantes, como *Scrum Master* e *Product Owner*, e eventos importantes para cada *sprint*, como:

- ***Sprint Planning:*** planejamento do que será desenvolvido e entregue em cada *sprint*;
- ***Daily:*** reuniões diárias de curta duração para acompanhar o progresso do desenvolvimento e compartilhar possíveis impedimentos.
- ***Sprint Review:*** apresentação do que foi desenvolvido ao final da *sprint*.
- ***Sprint Retrospective:*** levantamento e análise do que funcionou, o que será mantido e o que pode ser melhorado para a próxima *sprint*.

Dessa forma, o *Scrum* permite que o desenvolvimento seja organizado em *sprints* com entregas definidas, permitindo uma entrega incremental de valor.

2.3.1 Histórias de Usuário

Histórias de Usuário representam uma descrição curta e simples de uma funcionalidade a partir do ponto de vista do usuário que a deseja (COHN, 2024). São uma forma de representar os requisitos para o *backlog* e ajudar na descrição mais detalhada de tarefas, podendo, assim, ser usadas para complementar o *Scrum*. Usualmente, seguem uma estrutura fixa como mostrado na Figura 4.

Como [tipo de usuário], **eu gostaria de** [fazer algo] **para que** [finalidade ou motivo].

Figura 4 – *Estrutura de uma história de usuário*

Histórias de usuário devem ser detalhadas a fim de guiar o desenvolvimento da funcionalidade. Para isso, elas devem ter uma descrição e apresentar critérios de aceite, que são especificações do comportamento mínimo do sistema, dadas várias situações, que permitem dar a história em questão como terminada.

2.4 Considerações do Capítulo

O presente capítulo apresentou as metodologias utilizadas como base para o processo de desenho da solução, sendo elas o *Design Thinking* (LEWRICK; LINK; LEIFER, 2018) e o *Design Sprint* (KNAPP; ZERATSKY; KOWITZ, 2016). Ambas apresentam divisão em etapas, porém possuem algumas diferenças quanto a ferramentas disponíveis, atividades a serem realizadas e duração. Além disso, também explicita as diferentes arquiteturas usadas no projeto, e como elas se complementam, e as ferramentas e metodologias de desenvolvimento utilizadas.

3 Método do trabalho

Após um estudo mais aprofundado das metodologias apresentadas na Seção 2.1, foram escolhidas as ferramentas que melhor se adequam ao projeto, criando-se uma adaptação do *Design Thinking* e do *Design Sprint*. Dessa forma, seguiu-se a sequência de atividades e instruções proposta pelo *Design Sprint*, juntamente com sua atribuição de papéis, porém de forma mais espaçada e, quando possível, assíncrona.

No entanto, devido ao pouco contato prévio com o Cursinho da Poli e ao baixo conhecimento do problema enfrentado, optou-se por dar mais ênfase à sua definição. Além disso, devido à falta de disponibilidade de tempo do grupo em realizar as atividades de acordo com o cronograma proposto pelo *Design Sprint*, foram incorporadas a divisão de fases do *Design Thinking* e algumas ferramentas disponibilizadas por essa metodologia, como *Personas* e POV, explicadas na Seção 2.1.1.6.

Com isso em mente, foi montado o cronograma apresentado na Tabela 1.

Tabela 1 – Cronograma do projeto para 2024

	JAN	FEV	MAR	ABR	MAI	JUN	JUL	AGO	SET	OUT	NOV	DEZ
ADAPTAÇÃO DA METODOLOGIA	X											
PRIMEIRA ITERAÇÃO DESIGN SPRINT ADAPTADO	X	X	X	X								
DESENVOLVIMENTO					X	X	X	X	X	X		
TESTAGEM COM O CP											X	X
AVALIAÇÃO E COLETA DE DADOS												X

3.1 Aplicação do Método

A aplicação do método descrito teve por objetivo um melhor entendimento dos usuários e do problema, a ideação e testagem de diferentes soluções e o levantamento de requisitos relacionados ao principal problema a ser resolvido pelo sistema: a coleta, controle e gerenciamento de presença dos alunos do cursinho.

3.1.1 Preparação

A fase de preparação abarcou as discussões iniciais referentes à definição das ferramentas utilizadas no desenvolvimento do projeto, assim como os momentos em que seriam necessárias intervenções por parte dos demais *stakeholders* do projeto, sendo eles, o Cursinho Popular e o professor orientador. Além disso, foi feita a atribuição de papéis, prevista pelo *Design Sprint*, entre os integrantes do grupo.

3.1.2 Empatia

Nesta fase, buscou-se entender e simular o problema da presença enfrentado pelo Cursinho Popular da Poli. Primeiramente, realizaram-se entrevistas para mapear as *personas* dos professores do CP e entender a coleta e lançamento da presença dos alunos. Esta jornada era feita, para turmas presenciais, por meio de uma lista assinada pelos alunos, que era manualmente passada para uma planilha Excel, gerando atrasos, erros e a necessidade de armazenar a lista. Já para turmas online, a dificuldade em padronizar as informações pessoais dos alunos fazia com que a presença fosse ignorada sem o seu conhecimento.

Posteriormente, esses dados foram usados para simular o lançamento de uma lista de assinaturas para uma planilha do Google Planilhas, a fim de gerar maior empatia e entendimento do problema. Foram realizados sete testes, resultando em um tempo médio de 1 hora, 5 minutos e 40 segundos para o lançamento da presença de um mês. Além disso, a quantidade de erros média, ou seja, alunos que tiveram sua presença preenchida incorretamente na planilha, foi de 3,8 erros por teste.

3.1.3 Definição

Nesta fase, utilizou-se as ferramentas POV e *How Might We* para identificar as dores e os problemas enfrentados pelo CP e pelos seus alunos. Isso resultou nas seguintes necessidades, divididas entre os dois principais usuários mapeados durante a fase de empatia:

- **Quem:** Professores do Cursinho Popular da Poli.
- **Precisa:**
  - que a presença seja inserida diretamente no sistema;
  - simplificar a etapa de preparação do processo de presença;
  - integrar a presença online facilmente ao sistema;
  - substituir a maneira como a presença é apurada.

- **Por que?:**
  - o método atual é ineficiente e demorado, gerando atraso;
  - envolve atividades que fogem do horário do CP, como imprimir a lista e prepará-la para a data correta;
  - a presença online não está sendo monitorada devido ao desencontro dos dados dos alunos;
  - gera armazenamento indesejado de papel, que pode acabar se perdendo no processo e que, posteriormente, é descartado.
- **Quem:** O aluno do CP.
- **Precisa:** de uma maneira de acompanhar sua presença.
- **Por que?:** muitos precisam da vaga no cursinho e dos auxílios devido à baixa renda e estes são determinados com base na presença.

3.1.4 Ideação

Durante esta fase, foi realizada uma sessão de *Brainstorming* baseada nas necessidades levantadas. Entendeu-se que a solução consiste em dois sistemas, que permitam a devida divisão de responsabilidades de modo que um atenda ao professor, enquanto o outro atende o aluno.

Assim, foi decidido por um sistema *web* para o controle e gerenciamento feito pelo CP e pelos professores, que permite a criação de aulas, marcação de presença e visualização de métricas. Somado a isso, o desenvolvimento de um sistema *mobile*, possibilitando aos alunos marcarem e consultarem suas presenças.

Quanto ao método de marcação da presença, foi optado pelo cadastro de uma palavra-chave, pelo professor ou pelo cursinho, que deve ser inserida pelo aluno no momento em que a aula em questão estiver ocorrendo.

3.2 Prototipação e Testagem

Após a definição da solução e levantamento dos principais requisitos, foram desenvolvidos protótipos de baixa fidelidade (FOUNDATION, 2019) com o intuito de passar as ideias ao CP e coletar mais informações. Para isso, criou-se dois protótipos, um para cada sistema, usando a ferramenta de design e prototipação Figma.

3.2.1 Aplicação Web

O protótipo da aplicação *web*, apresentado na Figura 5, teve por foco emular atividades como a abertura da presença de uma aula, marcação manual da presença de um aluno sem acesso ao sistema e verificação das métricas de frequência dos alunos e turmas.

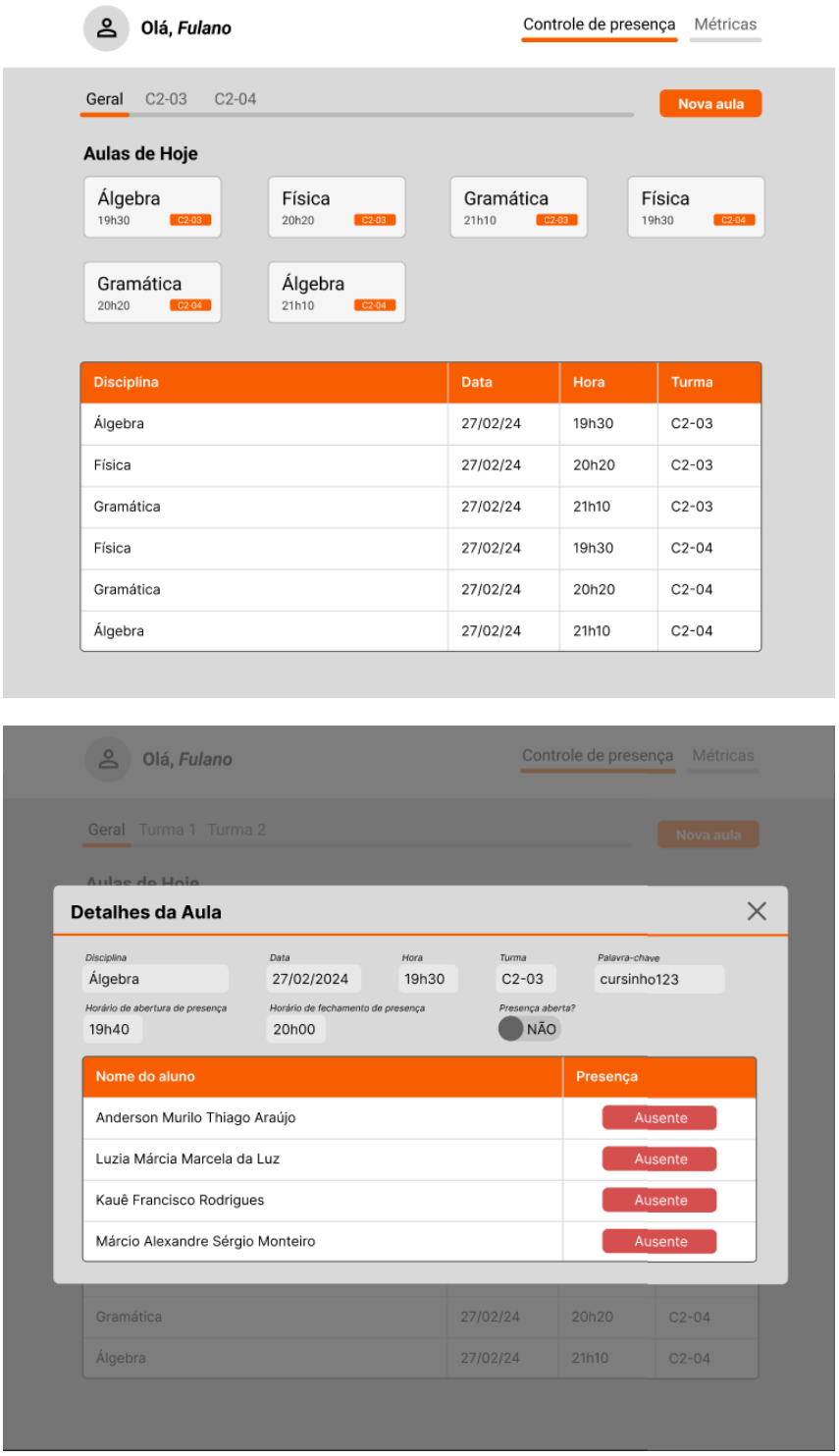

Figura 5 – *Interfaces do protótipo do sistema web*

3.2.2 Aplicação Mobile

Já o protótipo da aplicação *tmobile*, apresentado na Figura 6, teve como foco emular o perfil do aluno e atividades como a marcação da presença de uma aula e consulta de suas métricas.

Figura 6 – *Interface do protótipo do sistema mobile*

3.2.3 Teste e Resultados

Foram, então, marcadas entrevistas com os professores e alguns alunos do CP para a realização de testes de usabilidade dos protótipos. Devido à dificuldade em marcar as entrevistas graças à disponibilidade dos entrevistados, realizaram-se três testes com os professores e três com os alunos. Neles, foram passadas diferentes atividades, relacionadas às funcionalidades do sistema, para que o entrevistado realizasse, permitindo que ele explorasse o protótipo livremente enquanto era encorajado a verbalizar suas ações e pensamentos.

Com isso, foi possível entender como o usuário interagiu com o protótipo e quais foram suas ideias, comportamentos, impressões e sugestões sobre o formato do sistema. Assim, algumas mudanças foram feitas na ideia inicial, levando em consideração os dados levantados, e seguiu-se para a etapa de desenvolvimento.

3.3 Planejamento e Desenvolvimento

Para a etapa de desenvolvimento do sistema, foi usada a metodologia *Scrum* para organizar e gerenciar o projeto, introduzida na Seção 2.3. Dessa forma, tendo em vista a imprevisibilidade que um projeto de um ano com um cliente real, como o Cursinho da Poli, apresenta, o *Scrum* permite que alterações ou atrasos sejam superados de forma rápida. Além disso, permite uma visão mais clara do progresso devido às entregas incrementais, facilitando o acompanhamento do cronograma do projeto, apresentado no Capítulo 3.

Assim, o desenvolvimento foi dividido em *sprints* de duas semanas, com *plannings* organizadas no início do período, na qual as histórias de usuário eram detalhadas, divididas em formato de atividades e priorizadas. Além disso, ao final, o grupo realizava uma *review* e uma *retrospective*, assim como previsto pela metodologia. O fim da *sprint* era aproveitado, também, para agendar reuniões de acompanhamento com os *stakeholders* do projeto. Por fim, devido ao desencontro de horários e evolução mais lenta, o grupo preferiu adaptar a *daily* para uma reunião semanal com o mesmo objetivo.

3.4 Considerações do Capítulo

O método de trabalho adotado une o conhecimento do cliente, sua realidade e suas necessidades, através de uma adaptação das metodologias *Design Thinking* (LEWRICK; LINK; LEIFER, 2018) e *Design Sprint* (KNAPP; ZERATSKY; KOWITZ, 2016). Dessa forma, foi possível entender profundamente quem é o cliente, quais são os problemas e qual é a solução mais adequada. Além disso, dadas as características de duração do projeto e sua imprevisibilidade graças à presença de um cliente real, escolheu-se usar histórias do usuário e *Scrum* para o planejamento do desenvolvimento, resultando em um projeto com maior adaptabilidade, previsibilidade e desenvolvido de maneira incremental de acordo com a priorização das suas funcionalidades (SCHWABER; SUTHERLAND, 2020).

4 Especificação de Requisitos

Após o levantamento dos problemas e necessidades dos usuários, possíveis soluções e resultados dos testes, mostrados no Capítulo 3, foram usadas histórias do usuário para representar os requisitos dos sistemas, apresentadas na Seção 2.3.1. Como a solução foi dividida em duas plataformas distintas, uma *web* voltada aos representantes do Cursinho Popular da Poli (CP) e outra *mobile* para os seus alunos, o levantamento de requisitos foi feito pensando nessa separação.

4.1 Aplicação Web

O sistema *web* tem como principal objetivo controlar e monitorar a presença dos alunos do CP e possibilitar a consulta de métricas referentes às suas faltas. Para isso, foram produzidas histórias de usuário divididas entre três principais épicos e categorias, sendo elas cadastros no sistema, referente à criação e gerenciamento de entidades, controle de presença, que engloba toda a parte de edição, abertura e marcação de presença, e métricas. Além disso, foram produzidas histórias para a representação dos requisitos não funcionais da aplicação.

Tal divisão é apresentada nas tabelas 2, 3, 4 e 5, juntamente com as histórias que fazem parte dela e com os critérios de aceite produzidos para cada uma:

Tabela 2 – Requisitos relacionados a cadastros no sistema e consultas.

I. Cadastro no Sistema	
Histórias de Usuário	**Critérios de Aceite**
Como coordenador do CP, **eu gostaria de** realizar o cadastro dos alunos **a fim de** permitir que eles tenham acesso ao sistema.	• Sistema não deve permitir cadastro de emails inválidos ou já cadastrados; • Sistema deve permitir que o cadastro seja feito a partir de uma planilha, com as informações dos alunos, anexada ao sistema.
Como coordenador do CP, **eu gostaria de** redefinir minha senha **a fim de** recuperar o acesso em caso de comprometimento ou perda da senha.	• Sistema não deve enviar link para email inválido ou não cadastrado; • Sistema deve enviar link por email para redefinição da senha.

Continuação da Tabela 2

Histórias de Usuário	**Critérios de Aceite**
Como coordenador do CP, **eu gostaria de** criar, editar e excluir turmas **a fim de** organizar as aulas e alunos de cada uma	• O sistema não deve permitir criação de turmas com mesmo nome; • O sistema não deve permitir a criação com nome ou modalidade em branco; • O sistema deve pedir confirmação antes da exclusão.
Como coordenador do CP, **eu gostaria de** consultar as turmas **a fim de** consultar os alunos.	• As turmas devem ser mostradas em ordem alfabética.
Como coordenador do CP, **eu gostaria de** cadastrar, editar e excluir disciplinas **a fim de** inserir as aulas de cada uma no sistema e criar uma grade horária.	• O sistema não deve permitir criação de disciplinas com mesmo nome; • O sistema não deve permitir a criação de disciplinas com nome ou turma em branco; • O sistema deve pedir confirmação antes da exclusão.
Como coordenador do CP, **eu gostaria de** consultar as disciplinas **a fim de** alterar seu dia e horário.	• As disciplinas devem ser mostradas em ordem alfabética.
Como coordenador do CP, **eu gostaria de** agendar, editar e excluir aulas **a fim de** possibilitar aos alunos marcarem a presença via aplicativo.	• O sistema não deve permitir a criação de aulas com data, horário, turma e disciplina em branco; • O sistema deve permitir a criação de aulas semanais; • O sistema deve pedir confirmação antes da exclusão.
Como coordenador do CP, **eu gostaria de** consultar as aulas **a fim de** abrir a marcação de presença ou conferir a lista de presença.	• As aulas devem ser mostradas em ordem crescente de data; • Deve ser possível filtrar por turmas, disciplinas e data.

Fim da Tabela

Tabela 3 – Requisitos relacionados ao controle de presença dos alunos.

II. Controle de Presença	
Histórias de Usuário	**Critérios de Aceite**
Como professor do CP, **eu gostaria de** abrir a presença de uma aula **a fim de** permitir que os alunos marquem a presença em um tempo determinado.	• O sistema deve permitir abertura de presença mesmo em horários fora do marcado na aula.
Como professor do CP, **eu gostaria de** marcar a presença de um aluno manualmente **a fim de** prevenir casos em que o aluno não consiga marcar sua própria presença.	• O sistema deve permitir que o estado de um aluno seja alternado entre presente a ausente; • O sistema só deve permitir a troca caso a presença esteja aberta.
Como professor do CP, **eu gostaria de** alterar a palavra-chave da aula **a fim de** garantir segurança e personalizar a palavra ao meu gosto.	• O sistema não deve permitir que a palavra chave esteja em branco; • Caso nenhuma palavra chave seja escolhida, o sistema deve assumir um valor como padrão.
Como professor do CP, **eu gostaria de** consultar a presença de uma aula **a fim de** ver quem estava presente.	• Alunos em ordem alfabética; • Deve ser possível filtrar por nome; • O sistema não deve mostrar alunos que não fazem a aula em questão.

Tabela 4 – Requisitos relacionados às métricas de presença dos alunos.

III. Métricas	
Histórias de Usuário	**Critérios de Aceite**
Como coordenador do CP, **eu gostaria de** obter as métricas de presença dos alunos **a fim de** verificar a adesão e possíveis evasões.	• O sistema deve permitir filtragem por nome, turma e disciplina; • Devem ser mostrados dados da presença total e parcial do aluno; • Deve ser mostrado o histórico de presença do aluno.
Como coordenador do CP, **eu gostaria de** obter as métricas de presença das turmas **a fim de** entender se há algum padrão de faltas ligado às turmas.	• O sistema deve permitir filtragem por nome e disciplina; • Devem ser mostrados dados da presença média de cada turma considerando os seus alunos; • Deve ser mostrado o histórico de presença da turma.
Como coordenador do CP, **eu gostaria de** obter as métricas de presença das disciplina **a fim de** intervir em uma matéria caso esta tenha alta taxa de faltas.	• O sistema deve permitir filtragem por nome e turma; • Devem ser mostrados dados da presença média de cada disciplina considerando todos alunos; • Deve ser mostrado o histórico de presença da disciplina por turma.

Tabela 5 – Requisitos não funcionais.

IV. Requisitos Não Funcionais	
Histórias de Usuário	**Critérios de Aceite**
Como coordenador do CP, **eu gostaria que** o sistema estivesse disponível em qualquer dispositivo **a fim de** permitir o acesso de diferentes pessoas em diferentes lugares.	• O sistema deve ser hospedado em um provedor de serviços em nuvem; • O sistema deve ser acessível pela web.
Como coordenador do CP, **eu gostaria que** o sistema tivesse disponibilidade 24h **a fim de** permitir o acesso de diferentes pessoas em diferentes lugares.	• O sistema deve estar disponível 24h.
Como coordenador do CP, **eu gostaria que** apenas pessoas autorizadas tivessem acesso ao sistema **a fim de** proteger dados sensíveis.	• O sistema deve possuir autenticação e controle de administrador por parte do CP.
Como coordenador do CP, **eu gostaria que** o sistema fosse escalável **a fim de** possibilitar a adição de novas turmas, disciplinas e funcionalidades no futuro.	• O sistema deve prever a criação de novas entidades; • O deploy deve ser feito em um provedor que facilite a sua atualização; • O sistema deve ser modular e ter entidades e casos de uso definidos
Como coordenador do CP, **eu gostaria que** o sistema tivesse um custo baixo **uma vez que** o cursinho tem um cunho social e oferece seus serviços gratuitamente.	• O deploy deve ser feito em um provedor cujo custo seja proporcional ao uso; • O custo não deve ultrapassar $100 mensais.

4.2 Aplicação Mobile

O principal objetivo do sistema *mobile* é permitir aos alunos marcar sua própria presença, bem como consultar suas faltas na aula com o intuito de manter seus benefícios. Para isso, foram produzidas histórias de usuário tanto para os requisitos funcionais quanto para os não funcionais da aplicação. As tabelas 6 e 7 apresentam as histórias produzidas para descrever tais requisitos, juntamente com seus critérios de aceite:

Tabela 6 – Requisitos relacionados ao sistema mobile.

I. Requisitos Funcionais	
Histórias de Usuário	**Critérios de Aceite**
Como aluno do CP, **eu gostaria de** ter meu próprio login **a fim de** garantir que meus dados sejam vistos apenas por mim como aluno.	• O sistema só deve permitir o acesso para email e senha cadastrados previamente.
Como aluno do CP, **eu gostaria de** redefinir minha senha **a fim de** recuperar o acesso em caso de comprometimento ou perda da senha.	• Sistema não deve enviar link para email inválido ou não cadastrado; • Sistema deve enviar link por email para redefinição da senha;
Como aluno do CP, **eu gostaria de** marcar minha própria presença **a fim de** manter meus benefícios e prover acompanhamento pessoal ao CP.	• O sistema não deve permitir a inserção da palavra-chave caso a presença esteja fechada; • O sistema não deve marcar a presença caso a palavra-chave esteja incorreta; • O sistema deve permitir a marcação de presença apenas uma vez;
Como aluno do CP, **eu gostaria de** acompanhar as minhas faltas **a fim de** ter um controle melhor dos meus benefícios e adesão.	• O sistema deve mostrar a presença para cada disciplina. • O sistema deve mostrar disciplinas em que o aluno está matriculado, mas sem dado de presença, como 0
Como aluno do CP, **eu gostaria de** ver minha grade horária **a fim de** saber se aulas foram canceladas ou trocadas.	• O sistema não deve mostrar disciplinas nas quais o aluno não está cadastrado; • O sistema deve mostrar a grade horária da semana; • O sistema deve permitir consultar informações de cada disciplina.

Tabela 7 – Requisitos não funcionais relacionados ao sistema mobile.

II. Requisitos Não Funcionais	
Histórias de Usuário	**Critérios de Aceite**
Como aluno do CP, **eu gostaria que** o sistema estivesse disponível facilmente **a fim de** poder consultar minhas aulas onde quer que eu esteja.	• O sistema deve ser acessível por um aplicativo mobile.
Como aluno do CP, **eu gostaria que** o sistema tivesse disponibilidade 24h **a fim de** poder consultar minha grade quando quiser.	• O sistema deve estar disponível 24h.
Como aluno do CP, **eu gostaria que** o sistema fosse fácil de usar **a fim de** evitar curvas de aprendizagem.	• O sistema deve possuir interface intuitiva e de fácil uso.
Como aluno do CP, **eu gostaria que** o sistema fosse acessível mesmo com velocidade baixa de internet **a fim de** garantir meu acesso sem atrasos.	• O sistema deve ter um baixo consumo de rede; • O sistema deve ter um tempo de resposta e atualização de, no máximo, 5 segundos;

5 Desenvolvimento do Trabalho

Para o sistema *web*, voltado aos administradores do Cursinho Popular da Poli (CP), foram utilizados dois *frameworks*: *Django REST Framework* para o *back-end*, tendo como linguagem principal o *Python*, e *React* para o *front-end*, usando o *Typescript*. A escolha se baseou na familiaridade do grupo com a tecnologia, de modo a acelerar o desenvolvimento, e pela arquitetura do *Django*, que é baseada na arquitetura MVT (Django Software Foundation, 2024), apresentada na Seção 2.2.2.

Além disso, para gerenciar os estados da aplicação *front-end web*, utilizou-se *Redux*, uma biblioteca de *JavaScript* que, atualmente, é uma das principais formas de aplicar a arquitetura *flux* em aplicações *React* (FACEBOOK, 2014), apresentada na Seção 2.2.3.

Já para o sistema *mobile*, foi escolhido *React Native* como *framework*, cuja linguagem utilizada também foi o *Typescript*. A escolha se baseou na possibilidade da sua utilização para desenvolver para ambos os sistemas *Android* e *iOS*. Além disso, é um *framework* similar ao *React* tradicional, tornando a adaptação do grupo mais fácil. O *back-end* será compartilhado com o sistema *web*.

Para o banco de dados, escolheu-se *PostgreSQL* por ser um banco de dados relacional conhecido e com ampla documentação disponível, de modo que os serviços de hospedagem em nuvem costumam adotá-lo como uma das opções disponíveis.

Todos os códigos-fonte foram armazenados em repositórios no *GitHub* [1].

5.1 Arquitetura do Sistema

Para guiar o desenvolvimento dos sistemas *web* e *mobile*, foi definida a arquitetura a ser seguida pelo projeto. Devido à escolha do *Django* como ferramenta principal para a estruturação do *back-end*, a arquitetura do sistema é a MVT (*Model-View-Template*), seguida pelo próprio *framework*.

Para o sistema *web*, foi usada a arquitetura *flux* para contornar alguns dos problemas de complexidade e requisição bidirecional que são atrelados à arquitetura MVT. Assim, o usuário interage apenas com as *views* da arquitetura, composta pelos componentes *React* da aplicação, e, quando alguma interação dispara uma ação que altere o estado da aplicação, esta é despachada corretamente para a *store* responsável por aquele estado.

As ações que são disparadas pelo usuário podem conter lógica que interage com o *back-end* do sistema. Dessa forma, todas as requisições feitas pelo sistema e suas respostas são tratadas pelas ações, de forma assíncrona, antes de serem despachadas. Isso faz com que os estados da aplicação sejam atualizados apenas quando a requisição retorna um sucesso e minimiza a quantidade de requisições necessárias. Após isso, a aplicação atualiza os componentes necessários para refletir a mudança de estado na interface e para apresentar a mudança ao usuário. A arquitetura geral do sistema pode ser vista na Figura 7.

[1] (VIVEIROS; GARIERI; SANTANA, 2024)

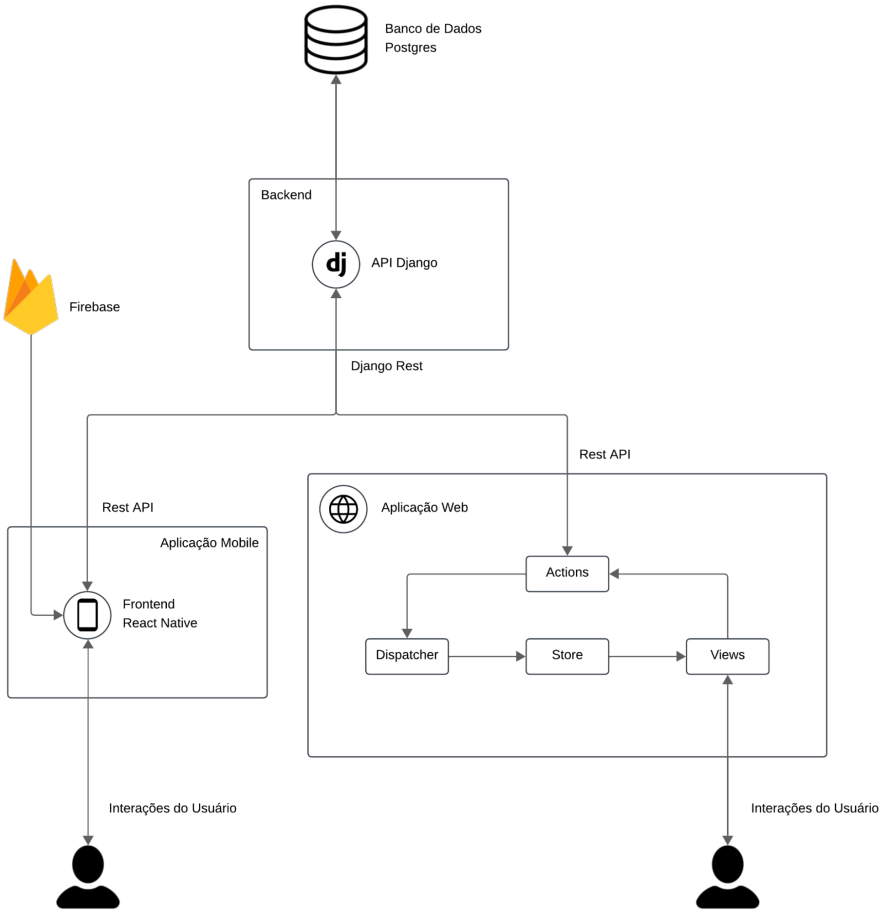

Figura 7 – *Diagrama Arquitetural do Sistema*

Já para o sistema *mobile*, foi escolhida a *Clean Architecture*, apresentada na Seção 2.2.1. Essa escolha se deu baseada na modularização e independência que o emprego de tal arquitetura permite, além de sua ampla aplicabilidade no mercado atual, devido ao fato de garantir fácil manutenção e implementação de novas funcionalidades. Isso proporciona um código expansível sem necessidade de ser modificado. A divisão de camadas da arquitetura do sistema *mobile* pode ser vista na Figura 8.

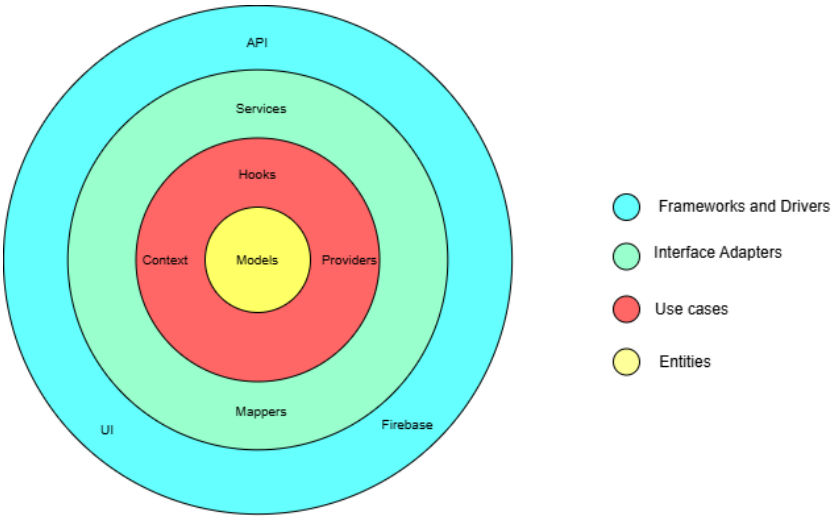

Figura 8 – *Diagrama Arquitetural do Front-End Mobile*

Além disso, alguns dos conceitos de *Clean Architecture* foram aproveitados e implementados,

também, no sistema *web*, como *entities* e *use cases*. Isso pois, mesmo usando arquitetura *flux*, os estados da aplicação ainda têm como centro as mesmas entidades e casos de uso do sistema *mobile*. Dessa forma, é possível maior padronização no código e no tratamento da *API Django*, de modo que, caso se deseje alterar qualquer coisa na API, a lógica das aplicações persiste devido ao isolamento entre as partes.

5.2 Sistema Web

O desenvolvimento do sistema *web* foi dividido entre diferentes fluxos para cada estado da aplicação. Cada fluxo prevê a criação, consulta, edição e exclusão dos principais objetos de cada estado, sendo eles os alunos, as turmas, as matérias, as aulas, as presenças e as métricas. A seguir são apresentados cada um desses fluxos, juntamente com as principais interfaces pertencentes a eles.

5.2.1 Autenticação

O fluxo de autenticação da aplicação *web* tem o objetivo de permitir apenas que os coordenadores do CP tenham acesso ao sistema. Ele consiste nas telas de *login* e de recuperação de senha, vistas na Figura 9.

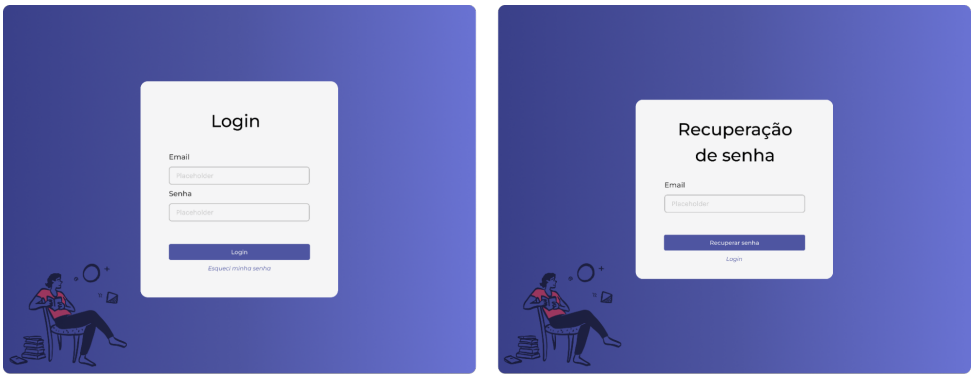

Figura 9 – *Interfaces do fluxo de autenticação*

A lógica de autenticação foi feita utilizando o protocolo de *tokens JWT*. Na ação de *login*, o usuário provê as credenciais de e-mail e senha de modo a retornar o *token* de acesso, com tempo de expiração curto, e o *token refresh*, usado para renová-lo quando ele expirar. Já a redefinição de senha verifica se o e-mail inserido está cadastrado no sistema e envia um *link* para a inserção da nova senha. Então, é gerado um *token* atrelado ao e-mail do usuário, de modo que a requisição só é bem-sucedida caso a senha tenha sido, de fato, alterada.

5.2.2 Gerenciamento de Alunos

O gerenciamento de alunos é o que permite ao CP adicionar e excluir alunos, alterando consequentemente o acesso deles ao sistema *mobile*, e fazer a sua consulta e edição, de forma a recuperar ou alterar as suas informações pessoais. Ainda é possível carregar uma planilha, cujo formato pode ser baixado no próprio sistema, para fazer o cadastro de vários alunos simultaneamente. As principais telas desse fluxo podem ser vistas na Figura 10.

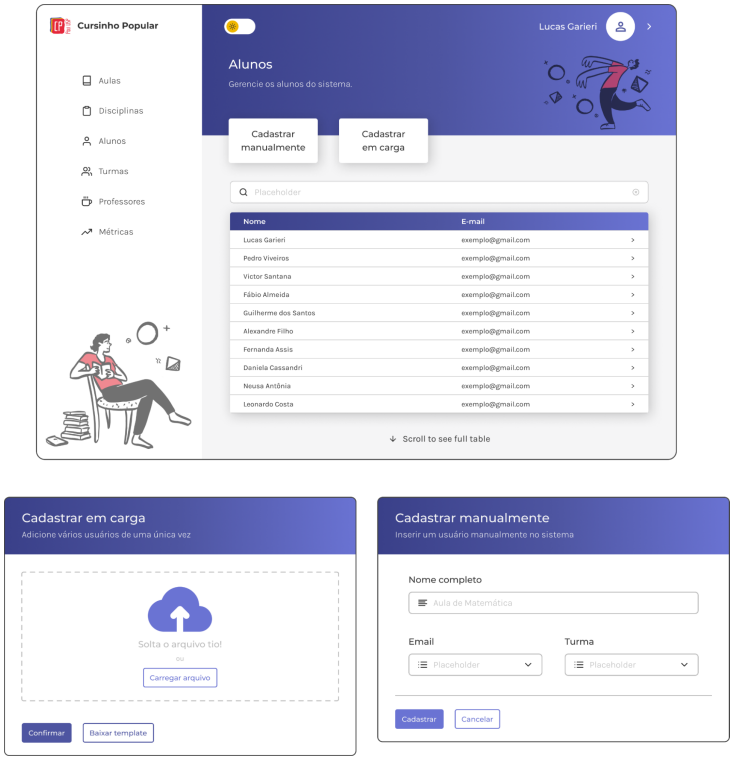

Figura 10 – *Interfaces do fluxo de gerenciamento de alunos*

A partir do cadastro bem-sucedido, os alunos recebem um e-mail para que possam cadastrar suas próprias senhas e acessar o sistema, fluxo similar ao de redefinição de senha, especificado na Seção 5.2.1.

5.2.3 Gerenciamento de Turmas

O gerenciamento de turmas permite que o CP adicione, exclua, consulte e edite as turmas do cursinho. Também permite o controle de algumas informações, como os alunos presentes em cada turma, a sala e a modalidade das aulas entre presencial e *online*. As principais telas desse fluxo podem ser vistas na Figura 11.

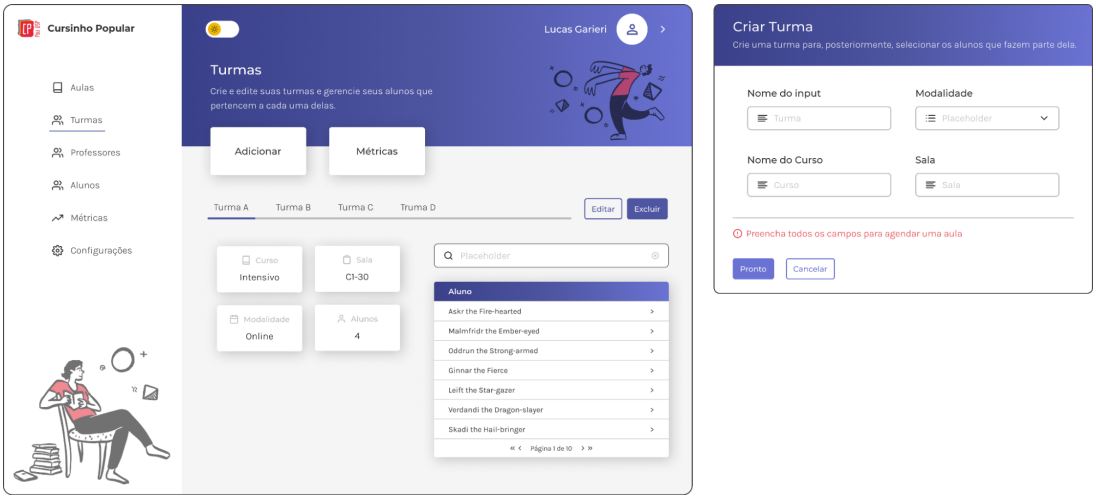

Figura 11 – *Interfaces do fluxo de gerenciamento de turmas*

5.2.4 Gerenciamento de Frentes

O fluxo de frentes do sistema tem o objetivo de permitir que o CP crie, edite e exclua diferentes frentes. Isso visto que o cursinho opera com matérias definidas, como português e matemática, mas possui diferentes frentes dentro de uma matéria, como geometria e álgebra, por exemplo, que podem sofrer alterações. A principal tela desse fluxo pode ser vista na Figura 12.

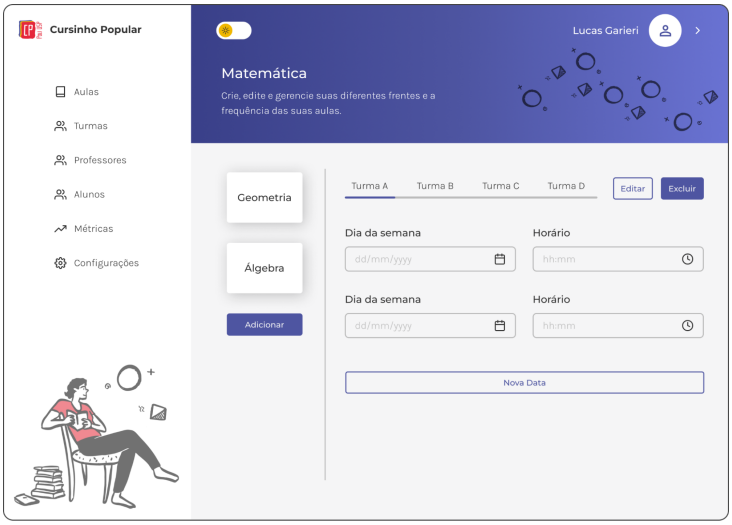

Figura 12 – *Interfaces do fluxo de gerenciamento de frentes*

5.2.5 Gerenciamento de Aulas

O gerenciamento de aulas permite que o CP crie, edite, consulte ou exclua aulas de maneira unitária. Ainda é possível marcar uma recorrência de aula para determinada turma, agendando, por exemplo, aulas de geometria para uma turma toda semana no mesmo dia e horário. As principais telas desse fluxo podem ser vistas na Figura 13.

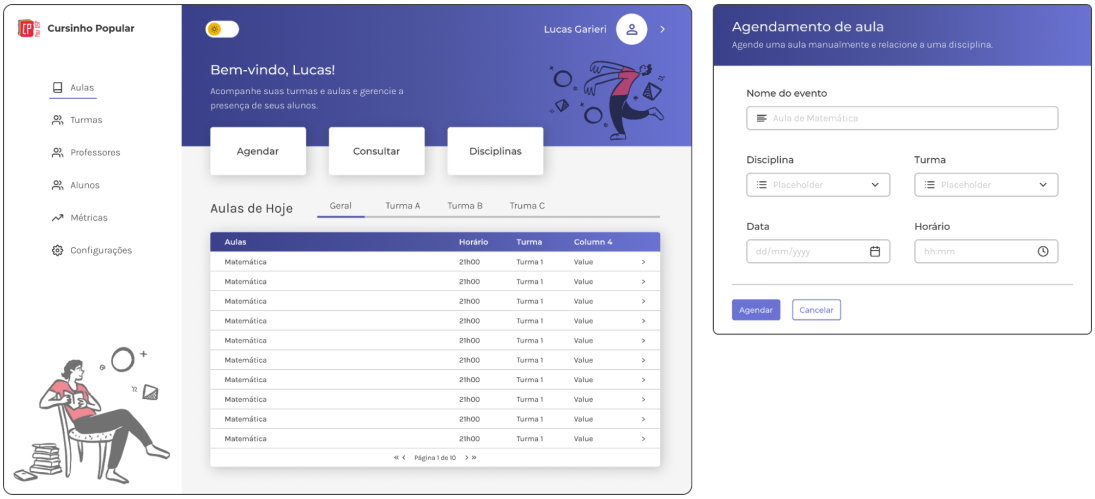

Figura 13 – *Interfaces do fluxo de gerenciamento de aulas*

5.2.6 Controle de Presença

O controle de presença é o que permite ao CP editar informações referentes à presença de uma aula, como a palavra-chave cadastrada e o horário em que a presença abrirá automaticamente. É possível, também, abrir a presença de forma manual e alterar a presença de um aluno em determinada aula manualmente. A principal tela desse fluxo pode ser vista na Figura 14.

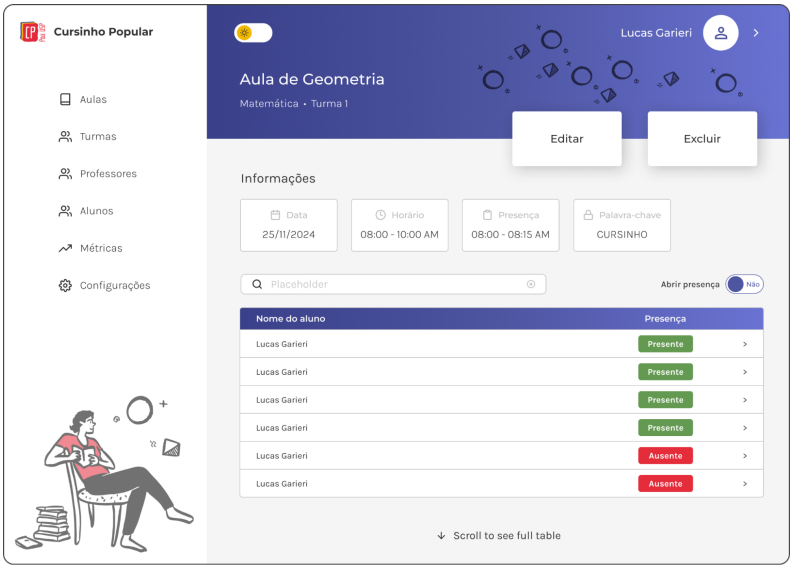

Figura 14 – *Interfaces do fluxo de controle de presença*

5.2.7 Vizualização de Métricas

A visualização de métricas é a funcionalidade que permite ao cursinho acompanhar as métricas de presença dos alunos, como a quantidade de aulas em que ele esteve presente, histórico de cada aula e taxa de faltas de cada turma ou matéria. Isso possibilita ao CP ter uma ideia melhor de possíveis pontos de evasão e se eles estão ligados a algum fator externo, como professores, matérias ou datas específicas. A principal tela desse fluxo pode ser vista na Figura 15.

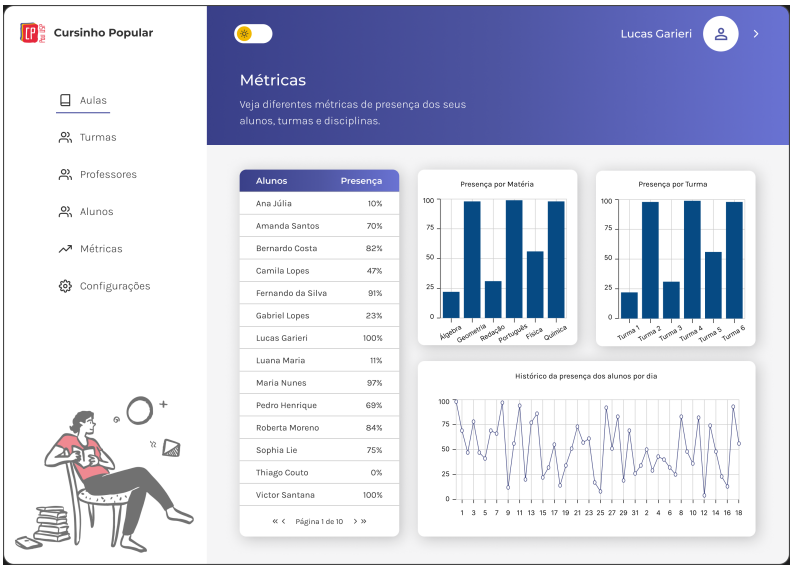

Figura 15 – *Interface do fluxo de visualização de métricas*

5.3 Sistema Mobile

O desenvolvimento do sistema móvel também pode ser dividido pelos fluxos a serem desenvolvidos. Trata-se de um sistema de quatro fluxos principais: autenticação, registro da presença, disciplinas e horários e o fluxo de métricas e ajustes.

5.3.1 Autenticação

O fluxo de autenticação da aplicação *mobile* visa garantir que apenas alunos com credenciais geradas através do sistema *web* possam acessar o sistema. Uma vez cadastrados os alunos, estes recebem seu *login* (endereço de e-mail) e uma senha pré-definida, que pode ser alterada através do e-mail registrado, utilizando o mesmo fluxo já apresentado no *frontend web*. As principais telas desse fluxo podem ser vistas na Figura 16.

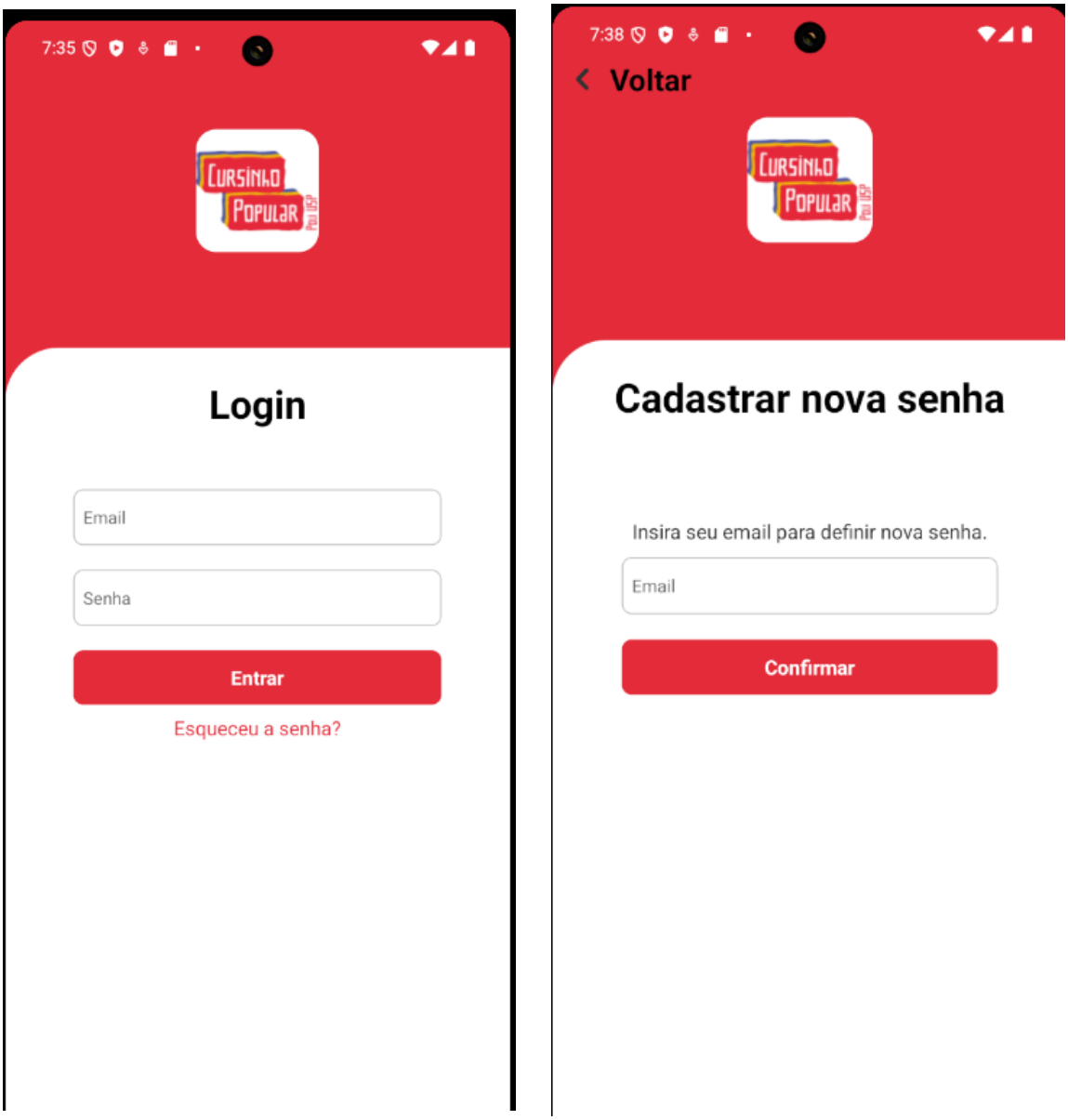

Figura 16 – *Interfaces do fluxo de autenticação mobile*

5.3.2 Fluxo de Registro de Presença

A tela principal do aplicativo apresenta para o aluno as aulas que ocorrerão no dia corrente, sendo que, ao iniciar o horário de uma aula, o *card* correspondente torna-se ativo, indicando assim que aquela aula se encontra em progresso. Uma vez que a presença esteja disponível para ser registrada, um botão surge no *card* que, quando clicado, abre um modal para a inserção da palavra-chave. As principais telas desse fluxo podem ser vistas na Figura 17.

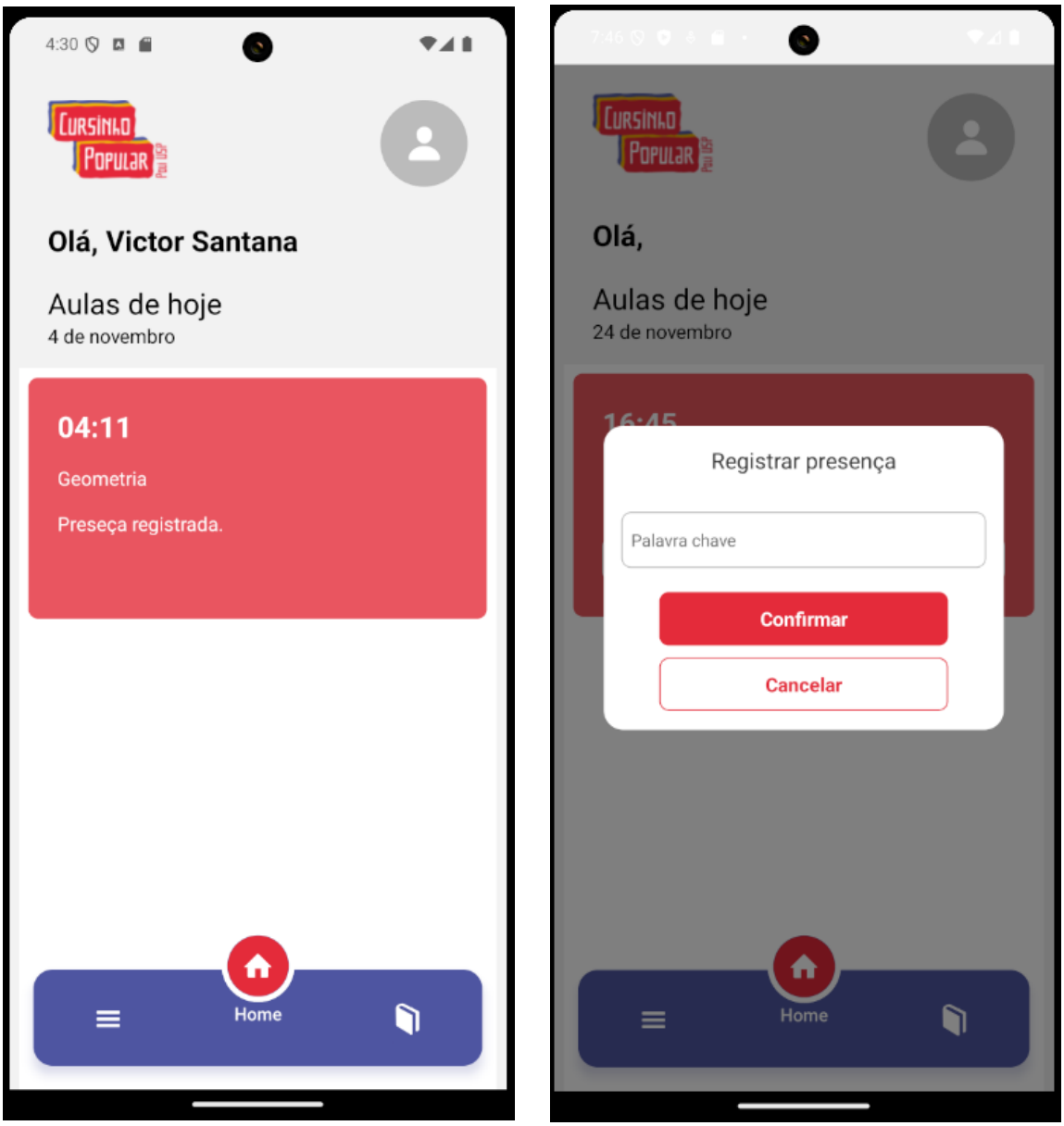

Figura 17 – *Interfaces do fluxo de autenticação mobile*

5.3.3 Fluxo de Disciplinas e Horários

A ideia do segundo fluxo do aplicativo é permitir ao aluno um fácil gerenciamento das suas disciplinas, assim como as frentes relacionadas a cada uma, tendo acesso também ao horário das aulas. Ao clicar em uma das disciplinas, o usuário acessa seus respectivos detalhes. Além disso, através do menu, o aluno tem acesso a uma visualização das suas aulas na semana através de uma grade horária, a qual

oferece a dinamicidade de apresentar em tempo real eventuais mudanças de última hora nas aulas da semana. As principais telas desse fluxo podem ser vistas na Figura 18.

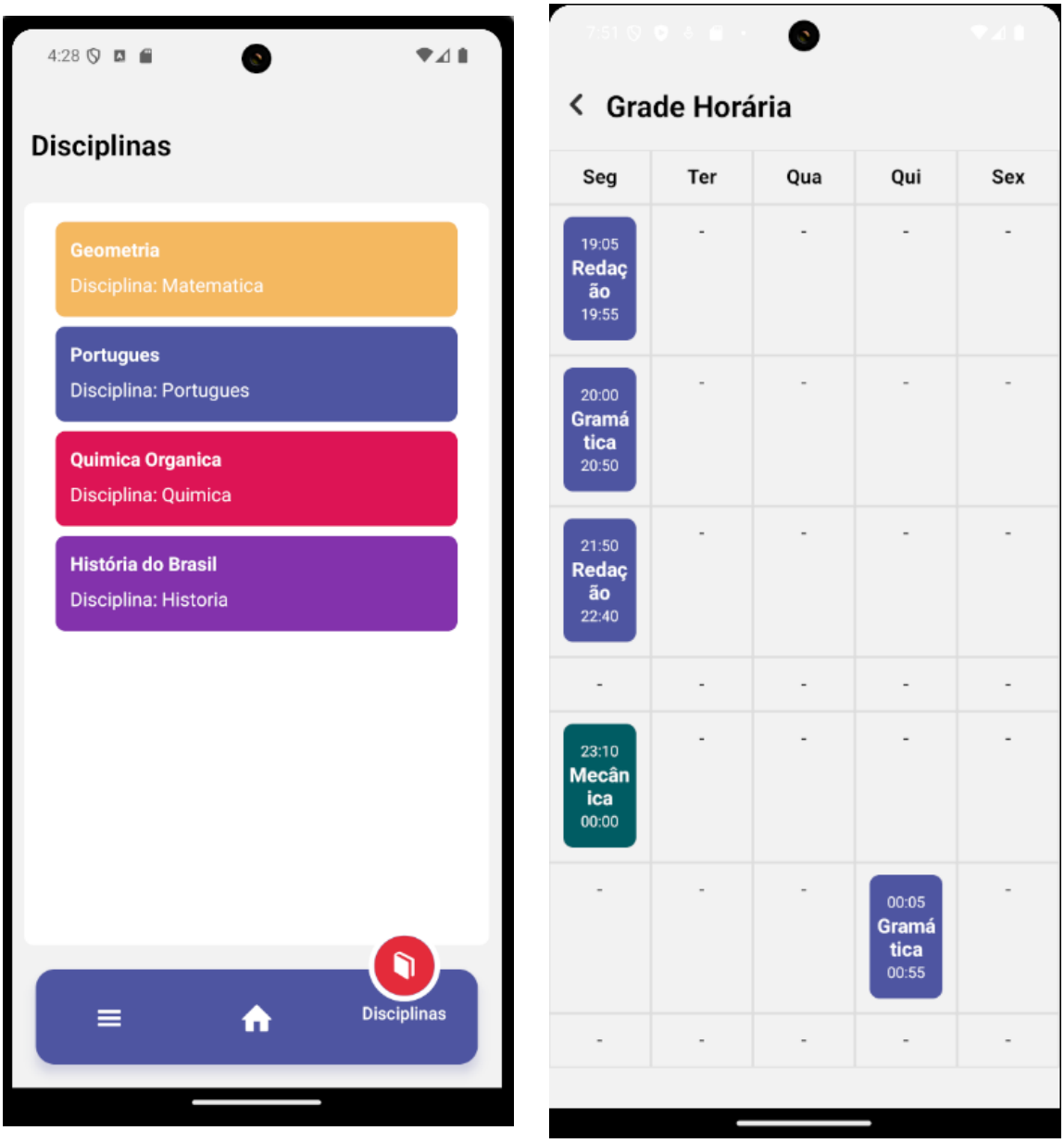

Figura 18 – *Interfaces do fluxo de horários e disciplinas mobile*

5.3.4 Fluxo de Métricas e Ajustes

O terceiro fluxo do aplicativo fornece ao aluno a possibilidade de configurar aspectos do sistema, tais como suas informações pessoais e de *login*. Além disso, o usuário também pode acessar informações relativas à sua presença nas aulas e eventuais benefícios ativos no momento da consulta. As principais telas desse fluxo podem ser vistas na Figura 19.

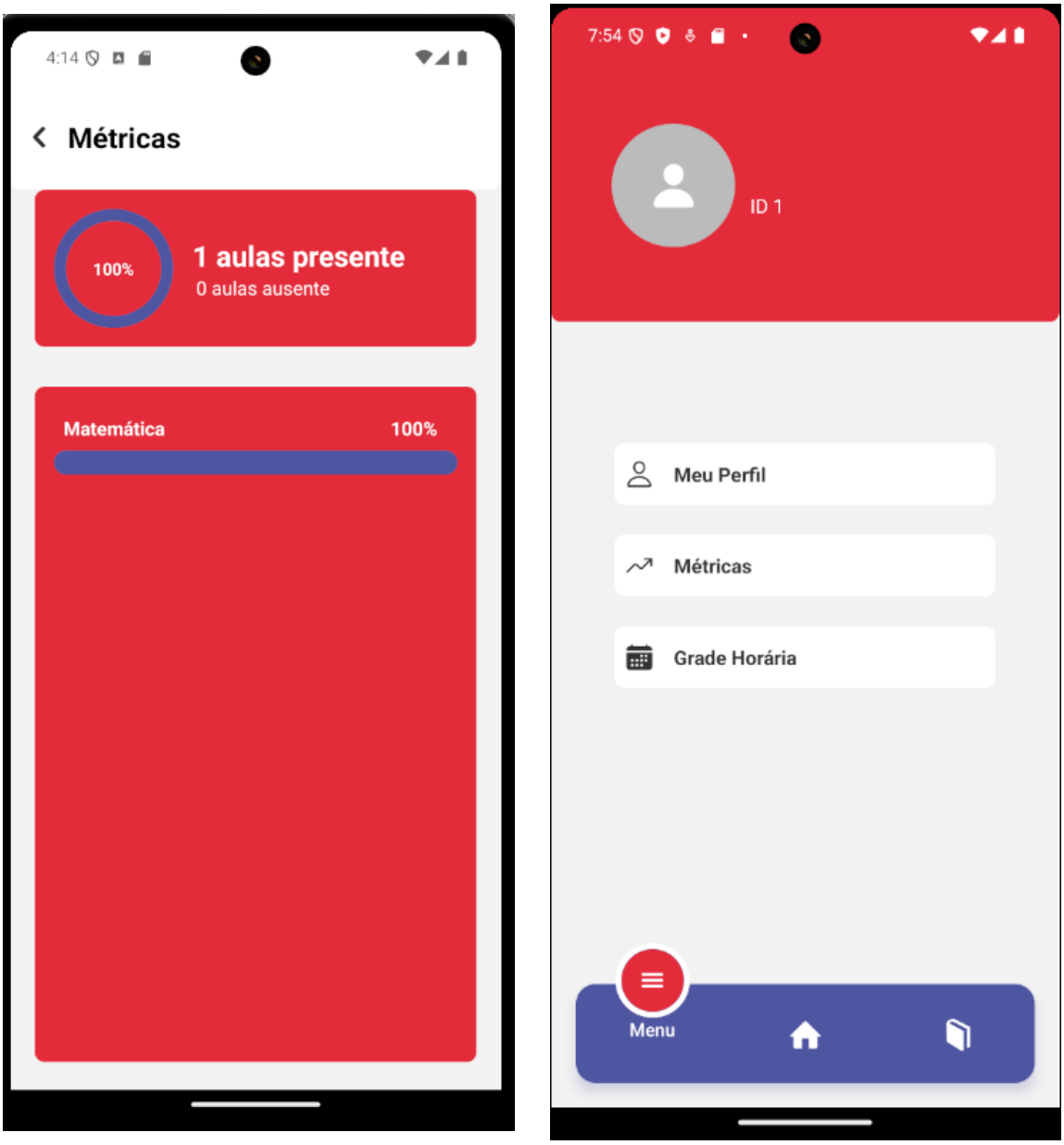

Figura 19 – *Interfaces do fluxo de métricas e ajustes mobile*

5.4 Infraestrutura de Software

Visto que, dado o final do projeto, espera-se que o Cursinho Popular da Poli (CP) tenha acesso ao serviço livremente para melhorias ou ajustes, escolheu-se um provedor em nuvem que permite uma maior personalização do sistema.

Tendo em mente reduzir os custos em razão do cunho social da organização, além da pouca familiaridade da gestão do CP com as tecnologias de infraestrutura, optou-se por um provedor que não só tivesse custos de manutenção baixos, mas também apresentasse automações para o *deploy* de novas versões.

Dessa forma, foram cotados três fornecedores distintos, escolhidos a partir de pesquisa, familiaridade e sugestões externas, considerando também sua qualidade, reputação e disponibilidade de documentação. A seguir, são apresentados os três fornecedores: *Amazon Web Services*, *Railway* e *Choreo*.

5.4.1 Amazon Web Services (AWS)

A escolha pela *AWS* foi baseada, principalmente, em sua reputação em meio aos provedores de computação em nuvem. Por ser muito conhecido, apresenta ampla documentação, facilitando o desenvolvimento pela gama de ferramentas disponíveis. No entanto, não apresenta automações para *deploys* em sua plataforma, dificultando a manutenção por parte de pessoas não familiarizadas com a tecnologia.

A cotação foi feita por meio da calculadora de preços da *AWS* que, a partir dos parâmetros de precificação, permite estimar o custo mensal em dólares da computação alocada. Os principais componentes que geram custo para a computação são *EC2*, *RDS*, *S3* e *Load Balancer*. A Figura 20 mostra tais serviços e como seria a comunicação e o fluxo de dados entre eles, caso fosse escolhido esse provedor.

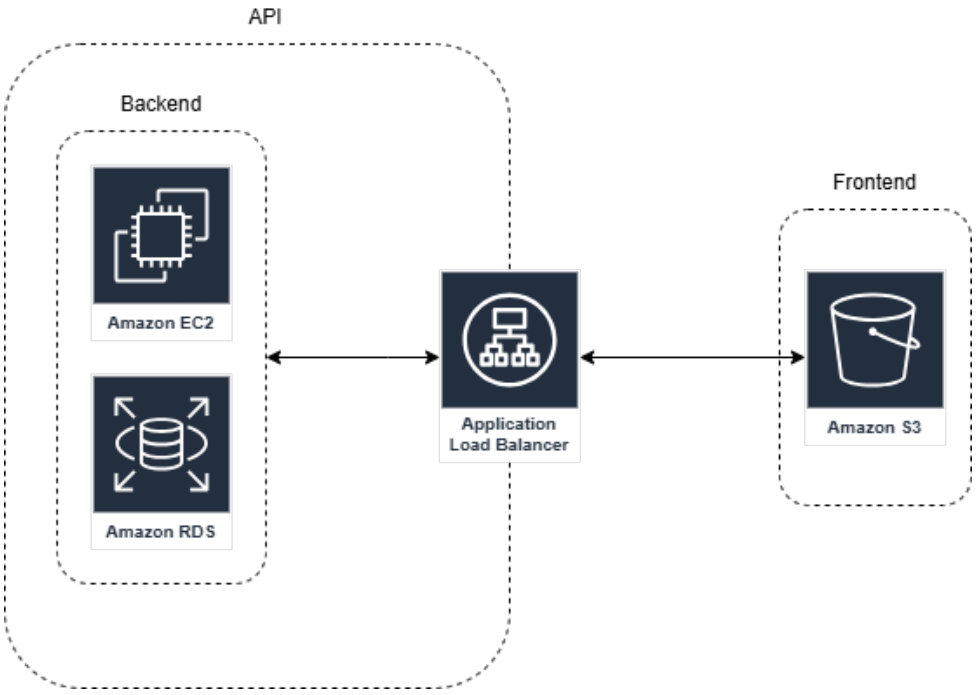

Figura 20 – *Diagrama da infraestrutura*

A *Amazon EC2* representa uma máquina, ou parte de uma, responsável por expor os IPs para acesso pela *internet*. Já o *RDS* representa o banco de dados em si, desenvolvido em *PostgreSQL*, onde os dados do sistema ficam armazenados. O *Load Balancer* auxilia no roteamento das chamadas vindas da interface (*web* ou *mobile*), aumentando ou diminuindo a necessidade de *hardware* e ordenando as chamadas. Por fim, o *S3* é responsável pelo armazenamento de arquivos estáticos, tal qual a interface web ou planilhas e figuras, por exemplo.

Assim, sabendo da função de cada um dos componentes de infraestrutura, foi feita a estimativa de custos apresentada na Tabela 8.

Tabela 8 – Estimativa de custos da *AWS*

Componente	**Características que impactam custo**	**Custo mensal**
EC2	• Instâncias compartilhadas • Linux • Cargas de trabalho oscilantes de acordo com horário • Instância *t3.medium* • Tipo de precificação: *Saving Plans*	US$36,94
RDS	• Armazenamento: 20GB • Nós: 1 • Tipo de instância: *db.t3.medium* • Utilização sob demanda • 100% de utilização por mês • Implantação em única Zona de Disponibilidade (AZ)	US$158,41
S3	• Armazenamento: 1GB/mês • Solicitações de escrita: 10 • Solicitações de leitura: 30000	US$0,06
Load Balancer	• Quantidade de *Load Balancers*: 1	US$24,87
Custo Total		US$220,28

Foi considerado 100% da utilização do banco de dados no mês para a estimativa de custo a fim de considerar o gasto máximo com a infraestrutura alocada. No entanto, o custo médio é reduzido pela metade em meses de utilização usual do sistema. Assim, o investimento mensal médio seria em torno de US$110.

Levando isso em conta, além da dificuldade técnica para manutenção do sistema, descartou-se a *AWS* como opção, ao menos durante o período de testes da aplicação.

5.4.2 Railway

Essa plataforma tem como principais vantagens a automação para *deploy* a partir de repositórios do *GitHub*, além de gerar diagramas da arquitetura do sistema. Quanto à precificação, o plano mais apropriado, dentre os oferecidos na plataforma, seria o *Pro*, focado para pequenas empresas que visam inserir suas aplicações na nuvem.

Seu custo base é de US$20,00 ao mês para cada desenvolvedor cadastrado na plataforma, o qual deve ser somado aos custos dos recursos utilizados (Railway, 2024). A tabela para utilização dos recursos é fixa e é apresentada na Tabela 9.

Como o gasto é variável em relação à utilização no mês e os recursos se auto-regulam conforme o

Tabela 9 – precificação de recursos na *Railway*

Recurso	**Custo**
RAM	US$10 / GB / mês (US$0.000231 / GB / minuto)
CPU	US$20 / vCPU / mês (US$0.000463 / vCPU / minuto)
Network Egress	US$0.10 / GB (US$0.000000095367432 / KB)
Volume Storage	US$0.25 / GB / mês (US$0.000005787037037 / GB / minuto)

período, a estimativa de custos é um pouco mais complexa de ser feita, porém pode-se inferir que seria utilizada, ao menos, uma vCPU para o servidor e 10GB de armazenamento, os quais geraram um custo de US$25 por si só.

Dito isso, em razão do custo base de US$20,00, somado ao custo de utilização, buscamos uma outra alternativa que permitisse maior personalização dos recursos alocados e, portanto, menor custo, ao menos para a versão usada nos testes.

5.4.3 Choreo

A plataforma tem como principais vantagens a automação de *deploy* para determinadas aplicações com base na linguagem de programação, bem como a precificação de acordo com a utilização e a escolha do desenvolvedor.

Dentre os planos disponíveis, o mais adequado é o *Developer*, visto que o CP apresenta poucos desenvolvedores. Além disso, o número de usuários levantados não exige que as máquinas apresentem mais de um nó para o bom funcionamento. A principal diferença quanto ao plano superior, *Team*, é o *deploy* em outras regiões além dos Estados Unidos, e também a expansão de nós em momentos de pico.

O custo é dado por componente, sendo que estes são ditados pelo sistema enviado para a nuvem. No caso do presente projeto, seriam dois componentes: uma API em *Django* e uma interface em *React*. O plano apresenta custo zero pelos primeiros cinco componentes e também US$100 em créditos gratuitos mensais para utilização de recursos. No caso, o único recurso utilizado seria o banco de dados, que custa US$0.14 por hora, sendo que os demais, como a máquina que dispõe a API ou a interface *web*, são desconsiderados do cálculo. Além disso, há o custo também de utilização da plataforma de US$10 semanalmente. Isso resulta em um custo mensal total de US$142,20 , considerando o uso contínuo.

5.4.4 Conclusão

Comparativamente, podemos analisar os custos totais de cada provedor em nuvem na Tabela 10.

Tabela 10 – comparação de custo de infraestrutura de testes

Provedor	**Custo Total Mensal Aproximado**
AWS	US$220
Railway	US$20 + Recursos
Choreo	US$40

Assim, analisando o custo mensal para o período de teste, previsto para duas semanas de duração, o menor custo seria ao escolher o *Choreo* como provedor de nuvem em razão dos 100 dólares gratuitos

no mês para uso de recursos, o custo final se deve inteiramente ao uso da plataforma (em razão da taxa de US$10 semanais). Isso, aliado à ferramenta de automação de *deploy*, inclinou o grupo a escolher essa plataforma.

5.5 Desafios e Soluções Técnicas

A concepção de um produto juntamente a um cliente real desde seu início, representou um desafio a mais para o escopo do projeto. A necessidade de testagens regulares, reuniões de alinhamento e acompanhamento e múltiplas entrevistas, somada à dificuldade de encontrar horários satisfatórios para ambas as partes, resultou em um avanço um pouco mais lento do que o esperado.

Ao mesmo tempo, a dinâmica de se trabalhar com um cliente e de ter que entender suas necessidades e problemas é algo que agrega para a importância do projeto desenvolvido. Esses fatores justificam a aplicação das metodologias de *Design Thinking* (LEWRICK; LINK; LEIFER, 2018) e *Design Sprint* (KNAPP; ZERATSKY; KOWITZ, 2016) escolhidas pelo grupo. Além disso, o uso de uma metodologia ágil, como *Scrum*, para o desenvolvimento foi feito dada a imprevisibilidade que essa dinâmica causa.

Ainda assim, tal dinâmica trouxe outros atritos significativos. Houve divergências na disponibilidade de membros do CP para atender a cada etapa do projeto, resultando em conflitos e culminando em atrasos e na necessidade de repriorizações e revisões de cronograma juntamente aos *stakeholders* do projeto.

Além disso, graças ao cunho social do cursinho e da sua missão, entender a realidade e as necessidades dos alunos se mostrou ainda mais importante. Isso também contribuiu para uma maior limitação quanto aos custos atrelados ao projeto, uma vez que o objetivo foi limitá-los ao máximo.

Por fim, dado que o produto será lançado e implementado pelo Cursinho da Poli, que passará a ser responsável pela sua manutenção, o rigor necessário com o projeto aumenta. Para garantir que o produto entregue seria de fácil manutenção e escalável, escolheu-se arquiteturas e tecnologias que contemplassem tais requisitos. Ainda, para certificar a qualidade do sistema, um plano de testes extensivo e que usa vários métodos, incluindo um teste prático com o CP, foi implementado.

6 Testes e Resultados

Terminado o desenvolvimento da primeira versão do sistema, montou-se um plano de testes para validar o produto com o Cursinho da Poli e mapear possíveis correções e melhorias. Para isso, foi desenvolvido um cronograma semanal, visto na Tabela 10, que prevê três etapas de validação. Nessa fase, novamente, a dificuldade imposta pela existência de um cliente real representou um desafio. A dificuldade em marcar horários e a necessidade de validação antes do lançamento do produto geram uma importância ainda maior para a fase de teste.

Tabela 11 – *Plano de testes do sistema*

	NOVEMBRO				DEZEMBRO	
	Semana 1	Semana 2	Semana 3	Semana 4	Semana 1	Semana 2
Teste de Usabilidade		X	X			
Teste Prático Controlado			X	X		
Pesquisa de Feedback					X	X

6.1 Teste de Usabilidade

A primeira etapa de testes foi composta de entrevistas com os professores e alunos do cursinho para a realização de testes de usabilidade, semelhante aos que foram feitos na fase de prototipagem do projeto. Foram feitas três entrevistas com os professores e três com os alunos, nas quais foram passadas a eles atividades referentes às principais funcionalidades de cada sistema, sendo elas, para o sistema *web*:

- Criar, editar e excluir uma matéria;
- Criar, editar e excluir uma turma;
- Cadastrar um aluno e adicioná-lo a uma turma;
- Agendar aulas recorrentes para uma matéria;
- Editar a palavra-chave e horário da presença de uma aula;
- Abrir a marcação de presença de uma aula;
- Marcar a presença de um aluno manualmente.

E, para o sistema mobile, no caso da entrevista com os alunos:

- Consultar as aulas do dia;
- Registrar a presença em uma aula;
- Consultar seu perfil e sua quantidade de faltas;
- Abrir a sua grade horária.

Foi pedido para que os entrevistados compartilhassem sua tela durante o procedimento, feito de maneira remota, e incentivou-se que eles verbalizassem suas ações e pensamentos. Ao final, liberou-se um tempo para dúvidas, sugestões e opiniões, sendo os dados anotados para possíveis correções e melhorias.

Todos os professores e os alunos conseguiram realizar as atividades sem problemas e avaliaram o sistema como intuitivo e completo. A interface do sistema e a personalização de acordo com a imagem do Cursinho da Poli também foram elogiadas.

Dentre as sugestões levantadas pelos professores, foi citada a melhoria de feedback para o usuário em momentos de carregamento das telas. Além disso, sugeriu-se, para trabalhos futuros, adicionar uma visualização das aulas marcadas em um formato de calendário, mais próximo ao que seria, por exemplo, o *Google Calendar*, ferramenta que foi, inclusive, citada pelos professores a parâmetro de comparação. Já para os alunos, foram citadas funcionalidades que, apesar de não terem relação direta com a presença, contribuem para que o aluno centralize o gerenciamento de seus estudos na aplicação *mobile*, como a adição de seções de simulados e de avisos, por exemplo.

6.2 Teste Prático Controlado

A segunda etapa consistiu em um teste prático feito em um ambiente real, porém controlado, por duas semanas. O Cursinho da Poli concordou em convidar três professores e quatro alunos da sua turma *online* para começar a usar o sistema. Durante esse período, a turma, matérias dos professores participantes e alunos voluntários foram cadastrados no sistema e, durante essas aulas, a presença foi marcada pela aplicação *mobile*.

Optou-se pela turma *online* uma vez que, como explicado no Capítulo 1, a sua presença atualmente não é considerada pelo Cursinho da Poli. Dessa forma, não haveria a necessidade de fazer com que o aluno marcasse presença duas vezes, uma pelo aplicativo e outra pela assinatura da lista.

Nesta etapa, os participantes foram encorajados a utilizar o sistema livre e normalmente, como usariam caso este já tivesse sido lançado. Pediu-se, também, que o CP e os voluntários comunicassem ao grupo quaisquer problemas, erros, melhorias, sugestões ou opiniões que surgissem durante o processo.

Ao final do processo, foi possível ter uma ideia melhor do funcionamento do projeto em um ambiente real. O teste também possibilitou a identificação e correção de alguns erros, como a expiração do *token* para a troca de senha dos alunos, explicado na Seção 5.2.1, após dois dias e um erro causado no agendamento de aulas pela inserção de um horário errado.

6.3 Pesquisa de Feedback

Por fim, a última etapa consistiu em um teste qualitativo de avaliação do sistema por parte das pessoas envolvidas na segunda etapa. A coleta desses dados foi feita por meio de um formulário preenchido pelos voluntários, com perguntas a respeito do desempenho, funcionalidade, apresentação e curva de aprendizagem do sistema, entre outros fatores.

Dentre as sete pessoas que participaram do teste prático controlado, seis responderam ao formulário, sendo quatro alunos e dois professores. Dentre eles, apenas uma pessoa declarou ter alguma dificuldade com o sistema, referente ao problema mencionado na Seção Seção 6.2. Além disso, pediu-se que os participantes dessem uma nota, variando entre um e cinco, para os fatores mencionados no início da seção. O resultado pode ser visto na Tabela 11.

Tabela 12 – *Notas dos usuários ao sistema*

Aspecto Avaliado	**Nota Média**
Experiência geral do sistema	4,17
Intuitividade do sistema	4,50
Beleza, clareza e organização da interface	4,83
Curva de aprendizado do sistema	4,00

Para as perguntas feitas apenas para os professores, todos os participantes disseram que a solução diminuiu o tempo necessário para a preparação e marcação da presença. Além disso, ambos acreditam que o sistema vai retirar o atraso atual na contabilização da presença, como pode ser visto nos gráficos mostrados na Figura 21.

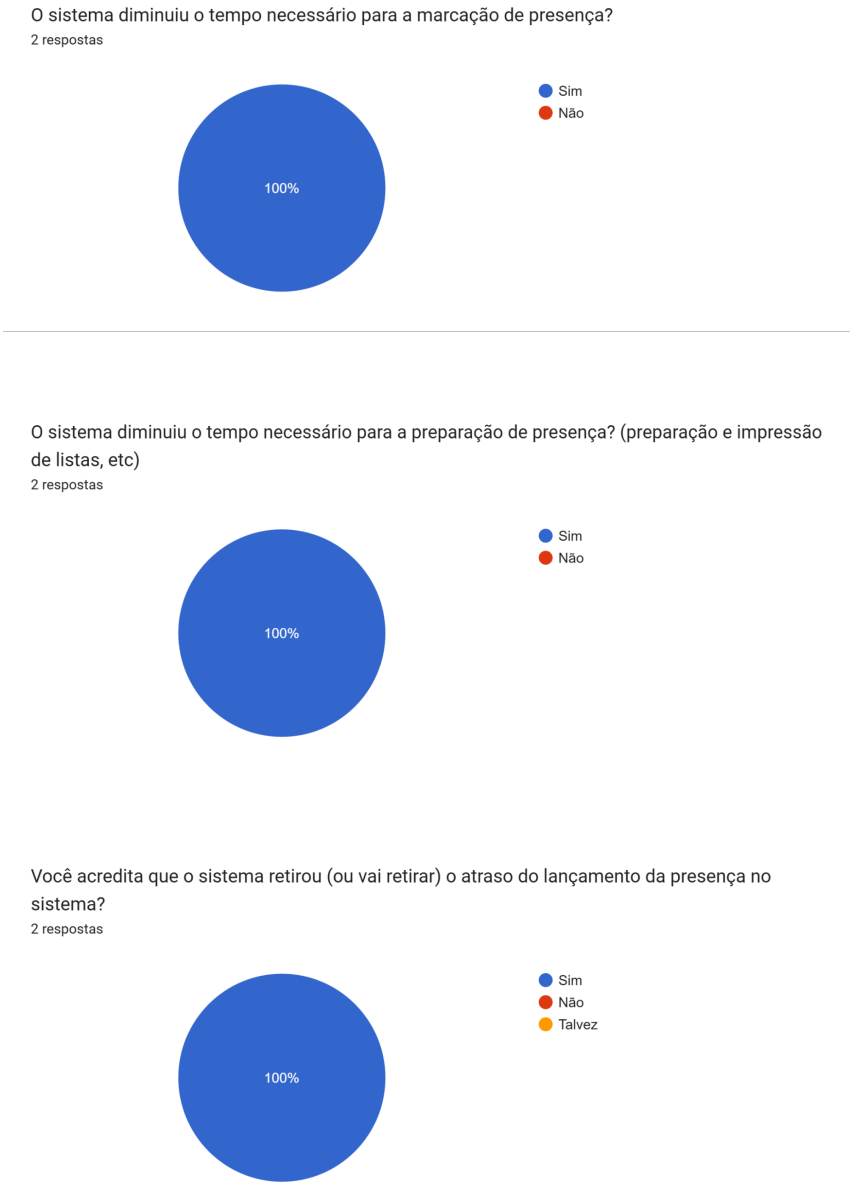

Figura 21 – Respostas dos professores sobre os objetivos do sistema

Já para os alunos, todos disseram desconhecer a existência de uma maneira para acompanharem sua presença antes da existência do sistema *mobile*. Além disso, três deles fazem uso de algum benefício oferecido pelo cursinho, porém três não controlam sua própria presença, reforçando a importância do aluno ter acesso facilitado a essa métrica. Os gráficos representando as respostas dos alunos são mostrados na Figura 22.

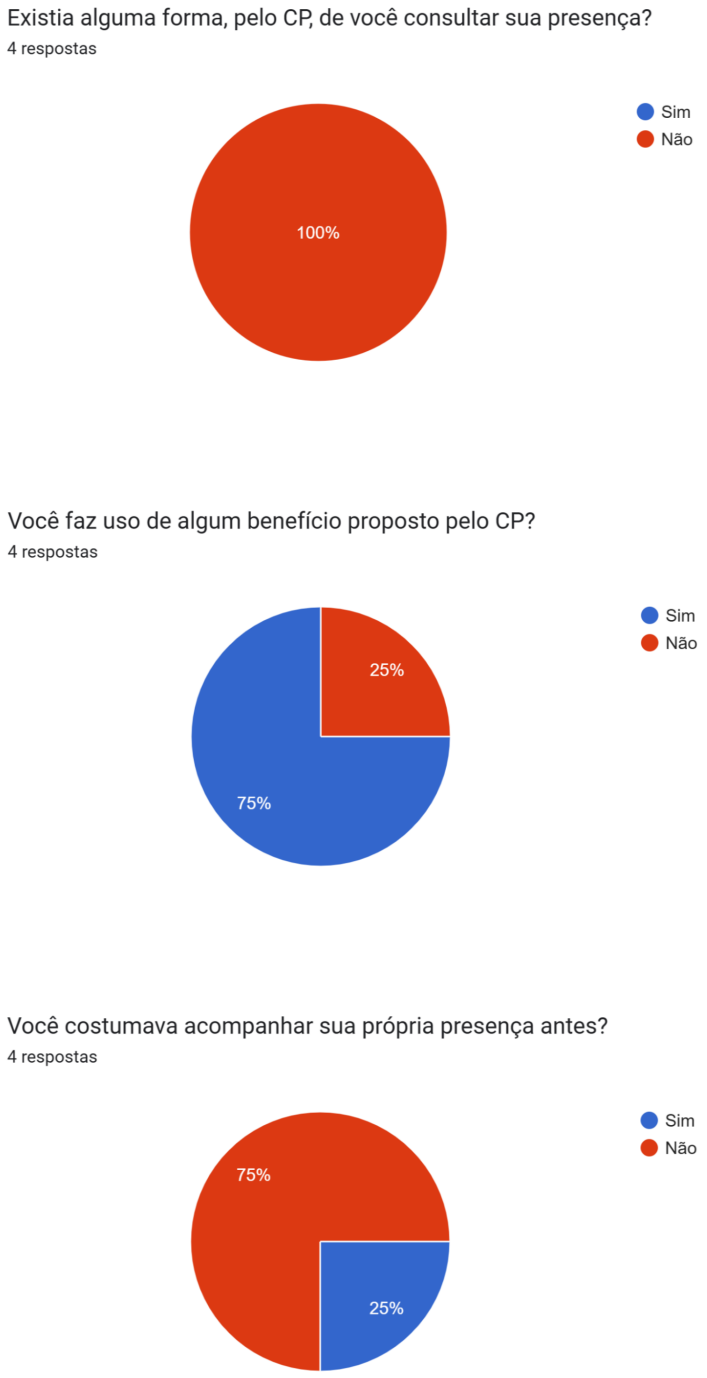

Figura 22 – Respostas dos professores sobre controle de presença

6.4 Resultados Encontrados

Os testes realizados demonstram que a solução atendeu aos requisitos e objetivos propostos, além de possibilitar a identificação e correção de erros antes do seu *deploy*. Foi possível, também, observar melhor a maneira com que os usuários interagem com o sistema e coletar seu *feedback*, que indicaram alta satisfação, com notas médias acima de 4,0 para os fatores apresentados.

Concluiu-se que o sistema propõe uma maneira eficiente de substituir a lista de presença e a necessidade de computá-la manualmente, além de oferecer aos alunos maior acessibilidade à sua frequência. Além disso, tanto alunos quanto professores avaliaram positivamente a experiência e a interface, apesar de algumas sugestões, que foram analisadas, priorizadas e melhor descritas na Seção 7.3.

7 Considerações Finais

O trabalho desenvolvido teve por objetivo o desenvolvimento de um sistema que pudesse ser utilizado como forma de otimizar alguns dos processos realizados no cotidiano do Cursinho Popular da Poli. Dessa forma, buscou-se entender quais eram as suas maiores necessidades e aplicar metodologias de ideação e *design* de soluções para desenvolver uma que fosse personalizada e adaptada para o caso do CP. Assim, foi possível desenvolver uma solução prática que atingiu os requisitos levantados e obteve um grande índice de satisfação.

7.1 Conclusões do Projeto

A solução final consiste em um sistema composto por uma plataforma *web*, para uso dos professores e coordenadores do CP, e uma plataforma *mobile*, para o uso dos alunos. A primeira é responsável por permitir que os professores gerenciem suas aulas e controlem a presença de seus alunos. Já a segunda permite que os estudantes do cursinho registrem sua presença e controlem suas faltas.

Baseado nos resultados dos testes, apresentados no Capítulo 6, concluiu-se que o sistema atingiu os requisitos inicialmente levantados e obteve um índice de satisfação alto, com uma nota média acima de 4. Além disso, o sistema propõe uma maneira eficiente de marcar e contabilizar a presença dos alunos, além de permitir que o cursinho comece a aferir a frequência da turma *online*. Ele também oferece, aos alunos, uma maior acessibilidade para consultar essas métricas. Assim, todos os problemas mapeados durante o processo de design foram endereçados e resolvidos.

Nesse processo, o uso de metodologias como *Design Thinking* (LEWRICK; LINK; LEIFER, 2018) e *Design Sprint* (KNAPP; ZERATSKY; KOWITZ, 2016), que focam nas necessidades do usuário, contribuiu para um melhor entendimento dos problemas e para a ideação da solução. Além disso, foi possível o desenvolvimento de um sistema personalizado para as necessidades do CP, o que implicou nas notas atingidas durante o teste das aplicações.

Além disso, considerando os desafios enfrentados de custo e disponibilidade, o sistema está hospedado em uma plataforma que garante sua disponibilidade 24 horas por dia, com custo dentro do orçamento inicialmente estipulado. Isso garante que os requisitos não funcionais do sistema também sejam satisfeitos.

7.2 Contribuições

O presente trabalho contribuiu para demonstrar como a automatização do processo de aferição de presença impactou e melhorou o cotidiano do Cursinho Popular da Escola Politécnica. Além disso, foi possível analisar como o uso de metodologias de *design* que focam nas necessidades do usuário e podem contribuir para a criação de uma solução mais personalizada, de forma a alinhar os objetivos e a experiência do usuário.

Ainda, a presença de um cliente real, como o CP, mostra os desafios que o desenvolvimento de um sistema pode apresentar. Dessa forma, a aplicação de uma metodologia ágil pode auxiliar no planejamento do projeto, proporcionando maior adaptabilidade e previsibilidade mediante atrasos.

Por fim, o projeto se destaca pelo seu impacto social, ao contribuir para a missão do Cursinho Popular da Poli e seu objetivo de levar a educação de forma gratuita a pessoas de baixa renda. Assim, a solução proposta contribui em retirar o atraso do processo de aferição de presença e promover maior acessibilidade aos dados de frequência dos alunos. Dessa forma, é possível detectar evasões de forma mais rápida, possibilitando intervenções mais eficientes por parte do cursinho.

7.3 Limitações e Trabalhos Futuros

Apesar do sistema desenvolvido ter atingido todos os requisitos inicialmente propostos, existem algumas limitações a serem consideradas em seu contexto atual. A etapa de testes, apresentada no Capítulo 6, também contribuiu para a coleta de sugestões, dos alunos e dos professores, para o sistema, que foram discutidas pelo grupo e priorizadas dentro do escopo do projeto.

- **Necessidade de manutenção:** como o sistema ficará em posse do Cursinho Popular da Poli, foi pensado ao máximo facilitar processos como o seu *deploy*, tratado na Seção 5.4. No entanto, a necessidade de manutenção no código e das aplicações, por parte do CP, pode ser um desafio visto que a solução não foi desenvolvida por eles. Quanto a isso, foi deixada uma documentação para auxiliar essa atividade da melhor forma possível.
- **Escalabilidade do banco de dados:** atualmente o sistema guarda as informações de presença dos alunos e aulas por tempo indeterminado, dada a necessidade de consulta e controle do cursinho. Um tempo de uso grande pode fazer com que o banco de dados cresça rapidamente, resultando em um aumento de custo e complexidade de manutenção, o que não é desejado dado o contexto social do CP.
- **Otimização da interface:** apesar do sistema *web* apresentar certo grau de responsividade permitindo seu uso em telas menores, como *tablets* ou monitores com proporções menos usuais, ele atualmente não suporta seu acesso por um aparelho *mobile*.
- **Quadro de avisos:** uma sugestão comum entre os alunos se refere a existência de uma área, dentro da aplicação mobile, que permitisse aos monitores anexarem avisos direcionados a cada turma. Com o aparato das *push notifications* já configurado, bastaria a adição de um novo modelo ao banco de dados para armazenar os avisos e adicionar a possibilidade de conceder permissão para criá-los apenas para alguns alunos.
- **Acompanhamento de simulados:** outra sugestão levantada pelos alunos que participaram dos testes do sistema é a adição de uma área, no aplicativo *mobile*, para acompanhamento de notas e simulados. Tal funcionalidade necessitaria da criação de um fluxo, na aplicação *web*, para realizar o cadastro das notas dos alunos, o que poderia ser feito de maneira similar ao cadastro em carga implementado.
- **Exportação de métricas:** por fim, dar a possibilidade de exportar as métricas de presença do sistema *web* em um formato específico, como pdf, pode ajudar o cursinho a compartilhar tais dados mais facilmente e a gerar relatórios periódicos para controlar a evasão dos alunos.

Referências

AHMED, N. *What is Design Thinking Process? How to use design thinking?* 2023. Disponível em: <https://www.linkedin.com/pulse/what-design-thinking-process-how-use-nasir-ahmed/>. Citado 2 vezes nas páginas 11 e 25.

CETIC. *Pesquisa sobre o Uso das Tecnologias de Informação e Comunicação nas Escolas Brasileiras 2022*. 2022. Disponível em: <https://cetic.br/media/docs/publicacoes/2/20231122132216/tic_educacao_2022_livro_completo.pdf>. Citado na página 21.

COHN, M. *User Stories and User Story Examples by Mike Cohn*. 2024. Disponível em: <https://www.mountaingoatsoftware.com/agile/user-stories>. Citado na página 32.

DAM, R. F.; SIANG, Y. S. *Define and Frame Your Design Challenge by Creating Your Point Of View and Ask "How Might We"*. 2020. Disponível em: <https://www.interaction-design.org/literature/article/define-and-frame-your-design-challenge-by-creating-your-point-of-view-and-ask-how-might-we>. Citado na página 27.

Django Software Foundation. *Django documentation*. 2024. Accessed: 2024-12-08. Disponível em: <https://docs.djangoproject.com/en/5.1/>. Citado na página 49.

FACEBOOK. *Flux: An Application Architecture for React*. 2014. Disponível em: <https://facebookarchive.github.io/flux/docs/in-depth-overview>. Citado 3 vezes nas páginas 30, 31 e 49.

FOUNDATION, I. D. *How Might We*. 2016. Disponível em: <https://www.interaction-design.org/literature/topics/how-might-we>. Citado na página 27.

FOUNDATION, I. D. *What Are Personas?* 2016. Disponível em: <https://www.interaction-design.org/literature/topics/personas>. Citado na página 27.

FOUNDATION, I. D. *What are User Interviews?* 2016. Disponível em: <https://www.interaction-design.org/literature/topics/user-interviews>. Citado na página 27.

FOUNDATION, I. D. *What is Prototyping?* 2019. Disponível em: <https://www.interaction-design.org/literature/topics/prototyping>. Citado na página 37.

HOLVILLA, T.; MYLLYAHO, E.; PORRES, I. Evaluating the mvc architecture in web applications. In: *Proceedings of the International Conference on Web Engineering*. Vienna, Austria: [s.n.], 2010. p. 93–100. Citado na página 30.

KNAPP, J.; ZERATSKY, J.; KOWITZ, B. *Sprint: How to Solve Big Problems and Test New Ideas in Just Five Days*. New York: Simon & Schuster, 2016. Citado 7 vezes nas páginas 23, 25, 27, 33, 40, 62 e 69.

LEWRICK, M.; LINK, P.; LEIFER, L. *The Design Thinking Playbook: Mindful Digital Transformation of Teams, Products, Services, Businesses and Ecosystems*. Hoboken, New Jersey: John Wiley & Sons, Inc., 2018. Citado 7 vezes nas páginas 23, 25, 29, 33, 40, 62 e 69.

MARTIN, R. C. *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. Boston: Prentice Hall, 2017. Citado na página 29.

Railway. *Pricing Plans | Railway Docs*. 2024. Disponível em: <https://docs.railway.com/reference/pricing/plans#resource-usage-pricing>. Acesso em: 23 nov. 2024. Citado na página 60.

SCHWABER, K.; SUTHERLAND, J. *Scrum Guide*. 2020. Disponível em: <https://scrumguides.org/scrum-guide.html>. Citado 2 vezes nas páginas 32 e 40.

UNIVERSITY, Y. *User Journey Maps | Usability & Web Accessibility*. 2024. Disponível em: <https://usability.yale.edu/understanding-your-user/user-journey-maps>. Citado na página 27.

VIVEIROS, P.; GARIERI, L.; SANTANA, V. *Backend: Repositório de backend do projeto*. 2024. Disponível em: <https://github.com/pedrohrodrig/cp-presenca-cursinho-backend>. Citado na página 49.