CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA

ETEC PROFESSOR ADHEMAR BATISTA HEMÉRITAS

Ensino Médio integrado ao Técnico em "Informática para a Internet"

Flávia Christine Hidalgo Pereira

Israel Cruz de Oliveira Filho

ARQ360

São Paulo – SP

2024

Flávia Christine Hidalgo Pereira

Israel Cruz De Oliveira Filho

ARQ360

Trabalho apresentado pelos alunos Flávia e Israel à Escola de Ensino Técnico Professor Adhemar Batista Heméritas, como parte das exigências para obtenção do título de Técnico em Informática para a Internet.

Orientador: Professor Valter Silva.

São Paulo – SP

2024

Dedicamos este trabalho a nossos gatos, que até nos dias mais difíceis, estiveram conosco para nos confortar.

AGRADECIMENTOS

Agradecemos imensamente um ao outro por todo o companheirismo que tivemos ao longo desta jornada.

Agradecemos a todos os nossos professores pelo inestimável apoio e pelos ensinamentos profundos que nos proporcionaram. Em particular, gostaríamos de destacar o professor Felipe Martins e o professor Alexandre De Aguiar E Silva, cujas orientações e encorajamento foram decisivos para que pudéssemos reconhecer e explorar plenamente nosso potencial.

Eu, Flávia, gostaria de dedicar um agradecimento especial ao meu irmão Lucas César. Ele tem sido não apenas um irmão excepcional, mas também minha luz no fim do túnel, meu guia e, acima de tudo, meu melhor amigo.

“All I want to do is be more like me, and be less like you”
(Tudo o que eu quero fazer é ser mais eu mesmo, e ser menos como você)
- Linkin Park

RESUMO

O presente trabalho aborda o desenvolvimento do ARQ360, um aplicativo inovador que integra visualização em 360 graus e armazenamento de projetos arquitetônicos, voltado para dispositivos móveis. A proposta surge como resposta a uma lacuna no mercado, marcada pela necessidade de ferramentas que ofereçam praticidade, eficiência e acessibilidade aos profissionais da área.

O projeto foi fundamentado em pesquisas bibliográficas, de mercado, quantitativas e qualitativas, que permitiram compreender as necessidades do público-alvo e estruturar soluções viáveis. Utilizando uma arquitetura em camadas, foram desenvolvidas funcionalidades essenciais, como um design intuitivo, um sistema robusto para gerenciamento de dados e um site complementar que reflete a identidade visual do aplicativo e facilita seu acesso.

Os resultados foram amplamente positivos, com feedbacks que destacaram a eficiência e a praticidade da plataforma. Além de cumprir os objetivos propostos, o ARQ360 proporcionou avanços significativos no cotidiano de arquitetos, otimizando a apresentação de projetos e promovendo maior interação com seus clientes.

Este trabalho reflete a aplicação prática do conhecimento adquirido na formação acadêmica e demonstra como soluções tecnológicas podem transformar a experiência profissional no setor da arquitetura, oferecendo um produto que atende às demandas do mercado com inovação e qualidade.

Palavras-chave: Arquitetura;360 graus; Inovação; Usabilidade.

ABSTRACT

The present work addresses the development of ARQ360, an innovative application that integrates 360-degree visualization and storage of architectural projects, aimed at mobile devices. The proposal arises as a response to a gap in the market, marked by the need for tools that offer practicality, efficiency, and accessibility to professionals in the area.

The project was based on bibliographic, market, quantitative and qualitative research, which allowed understanding the needs of the target audience and structuring viable solutions. Using a layered architecture, essential functionalities were developed, such as an intuitive design, a robust system for data management, and a complementary website that reflects the visual identity of the application and facilitates its access.

The results were largely positive, with feedback that highlighted the efficiency and practicality of the platform. In addition to meeting the proposed objectives, the ARQ360 provided advances in in the daily lives of architects, optimizing the presentation of projects and promoting greater interaction with their clients.

This work reflects the practical application of the knowledge acquired in academic training and demonstrates how technological solutions can transform the professional experience in the architecture sector, offering a product that meets the demands of the market with innovation and quality.

Keywords: Architecture; 360 degrees; Innovation; Usability

LISTA DE ILUSTRAÇÕES

LISTA DE GRÁFICOS E TABELAS

SUMÁRIO

1.0 INTRODUÇÃO

A integração entre arquitetura e tecnologia tem revolucionado a forma como os profissionais do setor desenvolvem e apresentam seus projetos, mas ainda enfrenta desafios significativos. Atualmente, muitos arquitetos precisam recorrer a múltiplos aplicativos e plataformas para atender a diferentes demandas de um único projeto, como armazenamento de dados, criação de apresentações visuais e compartilhamento de informações com clientes. Essa fragmentação não apenas aumenta a complexidade do trabalho, mas também pode gerar inconsistências nos dados, atrasos na entrega e dificuldades na adaptação a diferentes ferramentas.

Além disso, a maior parte dessas soluções está focada em dispositivos de alta performance, como desktops e notebooks, negligenciando a crescente demanda por ferramentas acessíveis em dispositivos móveis. Essa limitação restringe a mobilidade dos profissionais, que frequentemente precisam apresentar projetos em reuniões externas, feiras ou diretamente nas residências dos clientes. A falta de opções robustas e otimizadas para smartphones e tablets compromete a flexibilidade e a praticidade no cotidiano do arquiteto, tornando o processo menos eficiente e aumentando o esforço necessário para atender às expectativas dos clientes.

Esses desafios evidenciam a necessidade de soluções mais integradas e acessíveis, que acompanhem a evolução das tecnologias móveis e atendam às exigências de um mercado cada vez mais dinâmico e conectado. Neste contexto, o ARQ360 surge como uma solução inovadora. O ARQ360 oferece funções essenciais para arquitetos, como visualização em 360 graus de projetos, armazenamento seguro de arquivos e um sistema prático para compartilhamento com clientes. Além disso, permite acesso por dispositivos móveis, garantindo mobilidade e agilidade nas apresentações. Suas funcionalidades foram desenvolvidas para otimizar o fluxo de trabalho e melhorar a interação entre profissionais e clientes.

1.1 Objetivo Geral

Proporcionar uma solução confiável e acessível para o armazenamento e visualização de projetos arquitetônicos. Trazendo uma melhor experiencia tanto para os arquitetos quanto para os clientes e revolucionar o ramo da arquitetura.

1.2 Objetivo Especificado

Desenvolver um aplicativo para dispositivos móveis que integre visualização interativa em 360 graus e armazenamento seguro de projetos arquitetônicos, com o intuito de oferecer uma solução eficiente e acessível para arquitetos e clientes.

2.0 FUNDAMENTAÇÃO TEÓRICA

Tendo em vista os objetivos apresentados, foram realizadas pesquisas para ter o completo conhecimento sobre o assunto, tanto sobre a arquitetura quanto sobre a tecnologia utilizada.

2.1 Arquitetura

A arquitetura é a arte e ciência de projetar e construir ambientes, incluindo edifícios, espaços públicos, paisagens e planejamentos urbanos. Ela envolve a criação de projetos, organização e estruturação de espaços e ambientes com o objetivo de atender às necessidades humanas, funcionais e estéticas. Para isso é considerado aspectos práticos como funcionalidade, estética, segurança, eficiência, sustentabilidade, acessibilidade, contexto, orçamento, interação humana, regulamentações locais, viabilidade técnicas etc. Os arquitetos são os profissionais que desempenham o processo de criação desses espaços e ambientes, procurando atender às necessidades específicas dos clientes, fazendo com que seus projetos sejam seguros, funcionais, sustentáveis e visualmente atraentes de acordo com o que foi solicitado.

2.2 Visualização em 360 na arquitetura

Imagens 360 são conteúdos digitais que capturam um ambiente de forma completa. Para serem criadas, essas imagens devem seguir um formato específico conhecido como Equirretangular.

Essencialmente, uma imagem 360 pode ser visualizada como uma esfera, com o ponto de observação localizado no seu centro. A partir desse ponto, é possível olhar ao redor e visualizar o ambiente em todas as direções e ângulos.

Quando exibida em uma tela plana, a imagem 360 pode parecer distorcida, o que é normal. Isso ocorre porque a imagem não é estática, mas interativa. Para uma visualização adequada, é necessário usar um software ou plataforma apropriada, ou seja, um player 360.

2.3 Tecnologia na Arquitetura

A tecnologia e a arquitetura sempre estiveram interligadas, impulsionando a construção de projetos mais ambiciosos, a busca por melhores soluções e o reconhecimento no mercado. As ferramentas atuais impactam diretamente o tempo de trabalho e a qualidade dos resultados. Com o avanço contínuo de computadores e softwares, o setor de arquitetura tem passado por transformações significativas.

Atualmente, os arquitetos utilizam uma ampla gama de ferramentas e tecnologias para apresentar seus projetos aos clientes. Softwares de modelagem 3D e renderização, como AutoCAD e SketchUp, são essenciais para criar representações visuais detalhadas e realistas. Impressoras 3D e maquetes físicas permitem uma compreensão tátil e tridimensional dos projetos, enquanto apresentações digitais e vídeos ajudam a ilustrar os designs de forma dinâmica. Além disso, tablets e dispositivos móveis são úteis para mostrar detalhes do projeto de maneira interativa em reuniões.

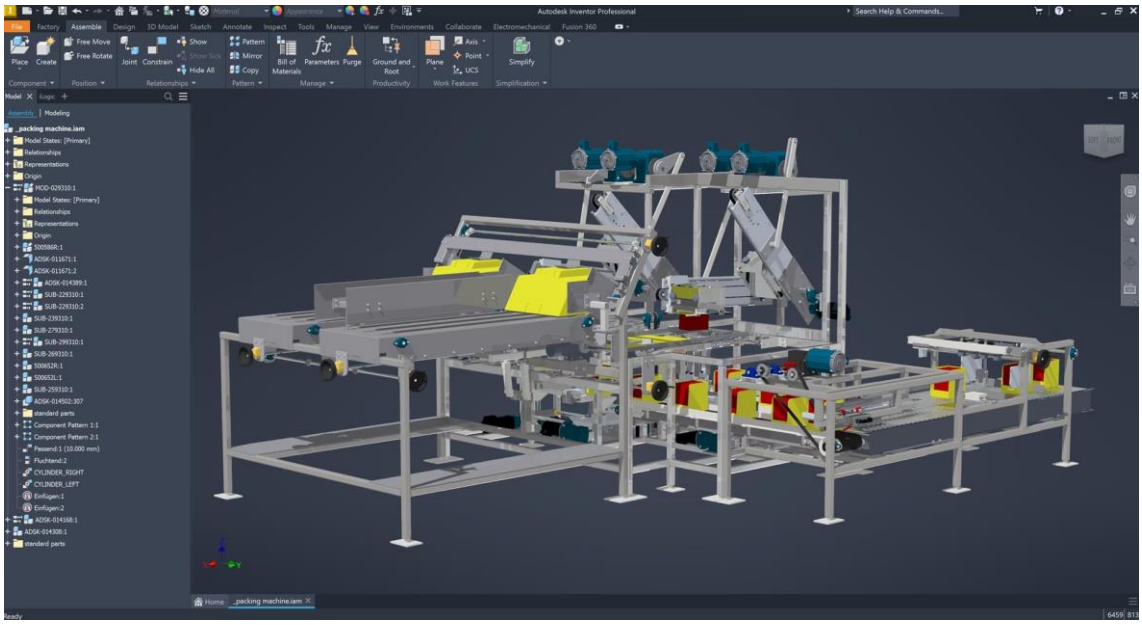
Figura 1: Figura 1: SketchUp 2019

Fonte:Download SketchUp 2019 Full Detailed Installation And Activation - Design of Architect (designofarchitects.blogspot.com)

2.4 Armazenamento de projetos arquitetônicos

Atualmente, arquitetos armazenam seus projetos utilizando algumas soluções digitais e tecnologias para garantir acessibilidade, segurança e colaboração eficiente. Entre as opções mais comuns estão os serviços de armazenamento em nuvem, como Google Drive e Dropbox, que oferecem acesso remoto e integração com outras ferramentas. Além disso, plataformas especializadas em gerenciamento de projetos e Building Information Modeling (BIM), como Autodesk BIM 360, são amplamente usadas para centralizar e gerenciar informações do projeto. Servidores locais também são utilizados por algumas empresas para manter o controle total sobre os dados, enquanto soluções de backup garantem a proteção contra perda de informações.

Figura 2: AutoDesk



Fonte: https://www.autodesk.com/pt/products/inventor/features

2.5 Impacto da tecnologia na comunicação entre arquitetos e clientes

Antes da introdução da Visualização em 360 graus no campo da arquitetura, arquitetos dependiam de métodos tradicionais como plantas bidimensionais, maquetes físicas, e renderizações 2D para comunicar suas ideias aos clientes. Embora essas ferramentas fossem úteis, elas frequentemente limitavam a capacidade dos clientes de entenderem plenamente o projeto. Plantas 2D e até mesmo renderizações 3D em telas planas não

conseguiam capturar a verdadeira sensação de escala, profundidade e materialidade de um espaço, levando a mal-entendidos e frustrações. Clientes sem experiência em arquitetura muitas vezes tinham dificuldades em visualizar como os desenhos bidimensionais se traduziriam em espaços tridimensionais reais, o que podia resultar em surpresas desagradáveis durante ou após a construção.

Com a chegada da Visualização em 360 graus, essa dinâmica mudou consideravelmente. Agora, os arquitetos podem criar ambientes imersivos onde os clientes podem ter mais imersão ao ver seus projetos. Isso facilita a compreensão do design, mas também permite que os clientes se familiarizem com a estrutura do projeto. A Visualização em 360 graus permite que os clientes entendam a visão do arquiteto com muito mais clareza, minimizando os riscos de mal-entendidos e reduzindo a necessidade de alterações dispendiosas durante a construção. Esta tecnologia aprimora a comunicação e a colaboração, tornando o processo de design mais fluido e satisfatório para ambas as partes

3.0 METODOLOGIA

Após a análise dos dados obtidos, é possível iniciar a construção do projeto, começando pela definição dos métodos de pesquisa utilizados para seu desenvolvimento e avaliando sua adequação ao público-alvo.

3.1 Pesquisas

Foram realizados quatro tipos de pesquisas para adquirir o conhecimento necessário sobre o projeto e avaliar sua viabilidade.

3.1.1 Pesquisa Bibliográfica:

Para a conclusão do trabalho, foram realizadas inúmeras pesquisas bibliográficas com o objetivo de aprofundar o conhecimento na área estudada. Essas pesquisas permitiram uma compreensão mais ampla e detalhada dos conceitos, tecnologias e práticas envolvidas. Com o intuito de exemplificar o embasamento teórico adquirido, foi incluída uma seção de fundamentação teórica que apresenta as principais referências utilizadas, destacando os estudos e fontes que contribuíram para a construção do conhecimento apresentado ao longo do trabalho.

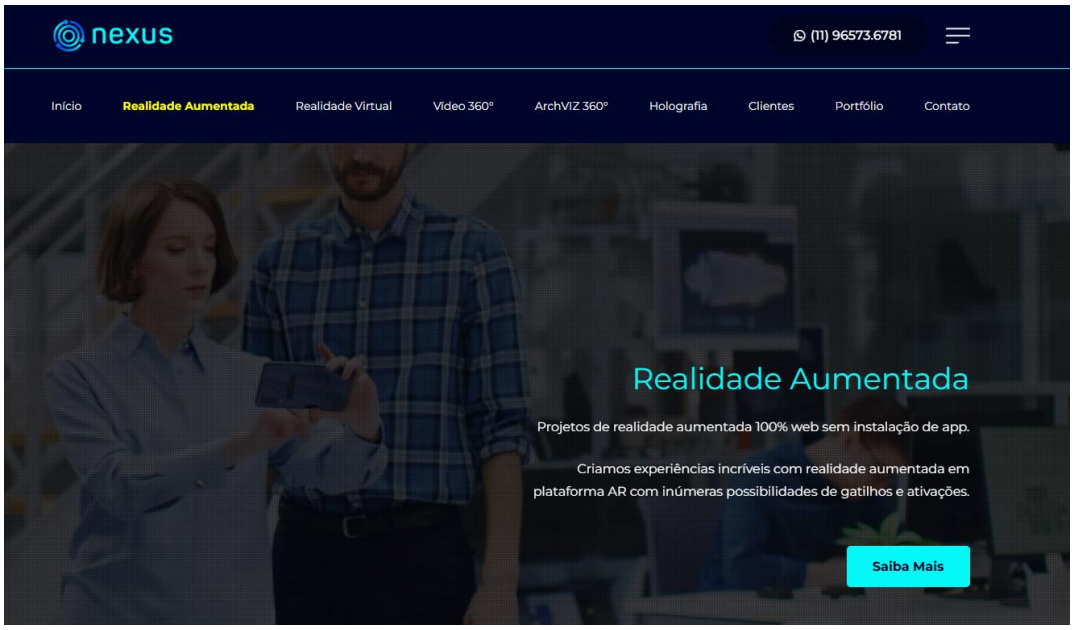
3.1.2 Pesquisa de Mercado

Uma Pesquisa de mercado consiste em um processo de coleta e análise de dados sobre um mercado para entender consumidores, concorrentes e oportunidades, ajudando na tomada de decisões de negócios.

Analisando o mercado de arquitetura com a visualização em 360 graus, é possível afirmar alguns concorrentes, são eles:

Nexus VR: Eles oferecem o "Decorado Virtual 360º", que permite aos clientes uma experiência realista e imersiva de como será seu futuro lar, através de ilustrações 3D que simulam os ambientes projetados.

Figura 3: Site da empresa Nexus



Fonte: Nexus - Realidade Virtual e Realidade Aumentada (nexusvr.com.br)

Aeon VR**:** Eles desenvolvem tours virtuais 360°, permitindo que os clientes explorem os projetos de forma completa, como se estivessem dentro do ambiente, utilizando óculos VR.

Figura 4: Site da empresa Aeon VR

Fonte: Aeon VR | Realidade Virtual

Em uma breve conclusão podemos ver que dentre essas empresas, elas optaram por utilizar websites ao invés de um aplicativo para dispositivos moveis, o que dá uma grande chance de destaque para o projeto no mercado, por essa falha significativa.

3.1.3 Pesquisa Quantitativa

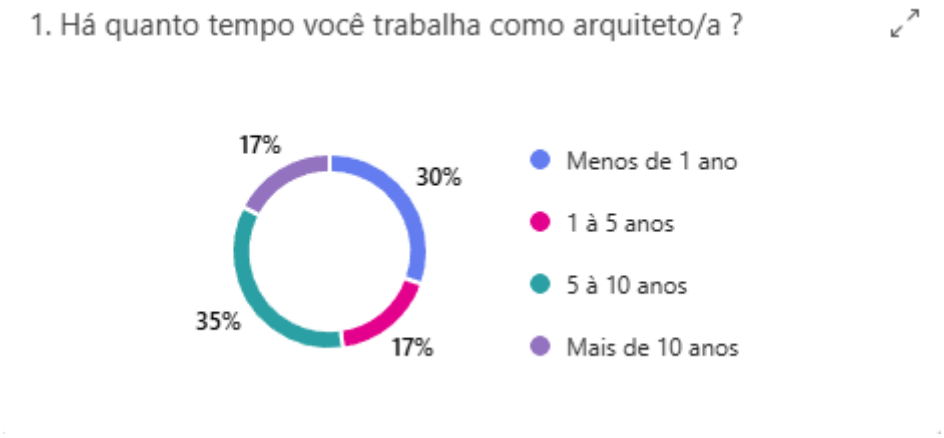
Uma Pesquisa Quantitativa coleta dados numéricos para identificar padrões e medir fenômenos, usando questionários e surveys para obter resultados objetivos e generalizáveis.

Foi utilizada a plataforma Microsoft Forms para conseguir as respostas.

1- Há quanto tempo você trabalha como arquiteto/a?

Essa pergunta permite entender a quanto tempo estão na área e consequentemente conseguimos avaliar a faixa etária do público.

Tabela 1 : Há quanto tempo você trabalha como arquiteto/a?

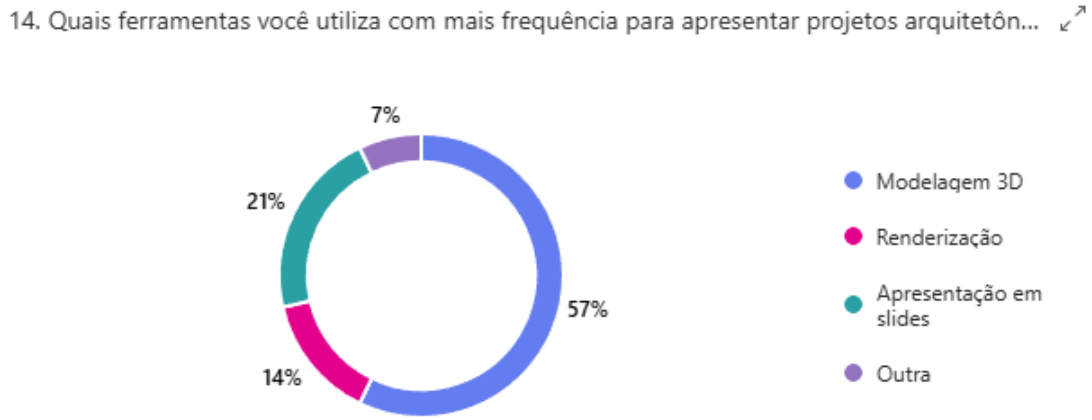


Fonte: Autoria Própria

2- Quais ferramentas você utiliza com mais frequência para apresentar projetos arquitetônicos?

Esta pergunta visa identificar as principais ferramentas utilizadas pelos arquitetos em suas práticas profissionais, fornecendo insights sobre os métodos predominantes de apresentação de projetos.

Tabela 2: Quais ferramentas você utiliza com mais frequência para apresentar projetos arquitetônicos?

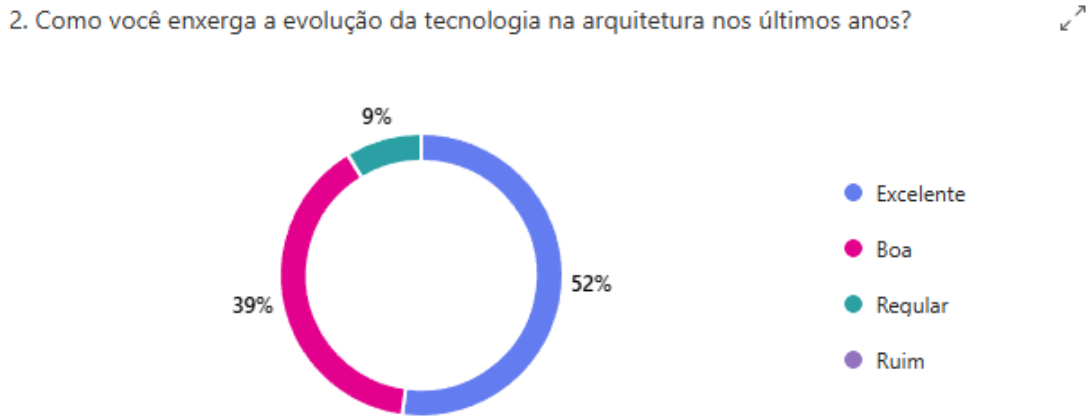


Fonte: Autoria Própria

3- Como você enxerga a evolução da tecnologia na arquitetura nos últimos anos?

Essa pergunta visa entender se o público alvo tem contato com a evolução da tecnologia na arquitetura e o quão bom é considerada.

Tabela 3: Como você enxerga a evolução da tecnologia na arquitetura nos últimos anos?



Fonte: Autoria Própria

4- Você acredita que a tecnologia pode proporcionar melhorias significativas na forma como os arquitetos apresentam seus projetos aos clientes?

Com essa resposta é possível ver qual a opinião do público sobre as melhorias que ocorrem devido a implementação da tecnologia na arquitetura, gerando uma melhor experiência aos clientes.

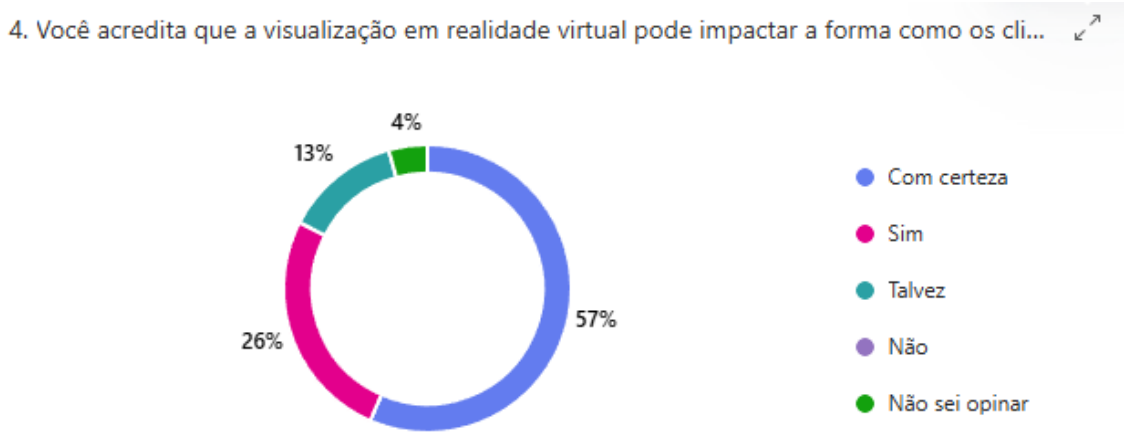
Tabela 4: Você acredita que a tecnologia pode proporcionar melhorias significativas na forma como os arquitetos apresentam seus projetos aos clientes?

Fonte: Autoria Própria

5- Você acredita que a visualização em realidade virtual pode impactar a forma como os clientes percebem seus projetos?

Dando continuidade a última pergunta, é possível analisar se o público considera relevante esse impacto.

Tabela 5: Você acredita que a visualização em realidade virtual pode impactar a forma como os clientes percebem seus projetos?

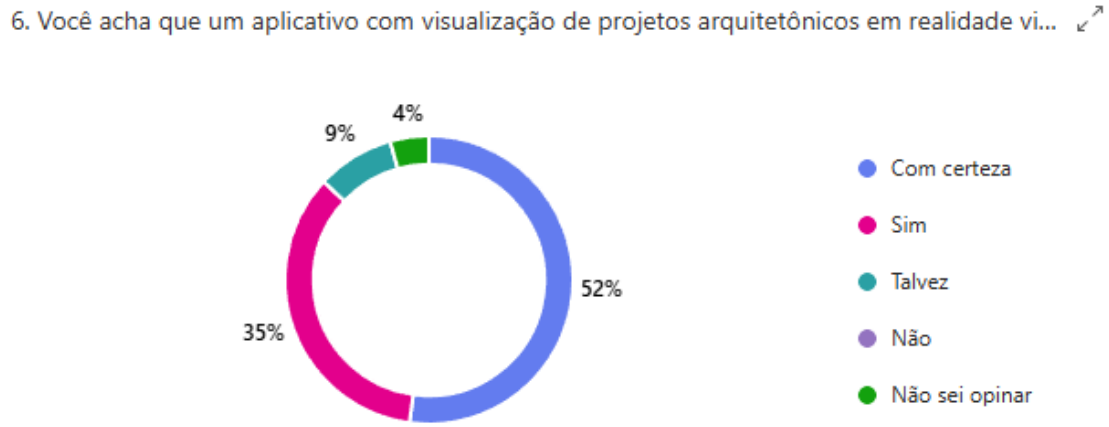


Fonte: Autoria Própria

6- Você acha que um aplicativo com visualização de projetos arquitetônicos em realidade virtual poderia melhorar sua comunicação com os clientes?

Essa pergunta mostra se o público considera significante a mudança na comunicação.

Tabela 6: Você acha que um aplicativo com visualização de projetos arquitetônicos em realidade virtual poderia melhorar sua comunicação com os clientes?



Fonte: Autoria Própria

7- Em uma escala de 1 a 10, quão importante você considera a utilização da tecnologia de realidade virtual em seu trabalho?

Esta pergunta visa entender se o público já faz uso da tecnologia no seu ambiente de trabalho e a sua importância ao público.

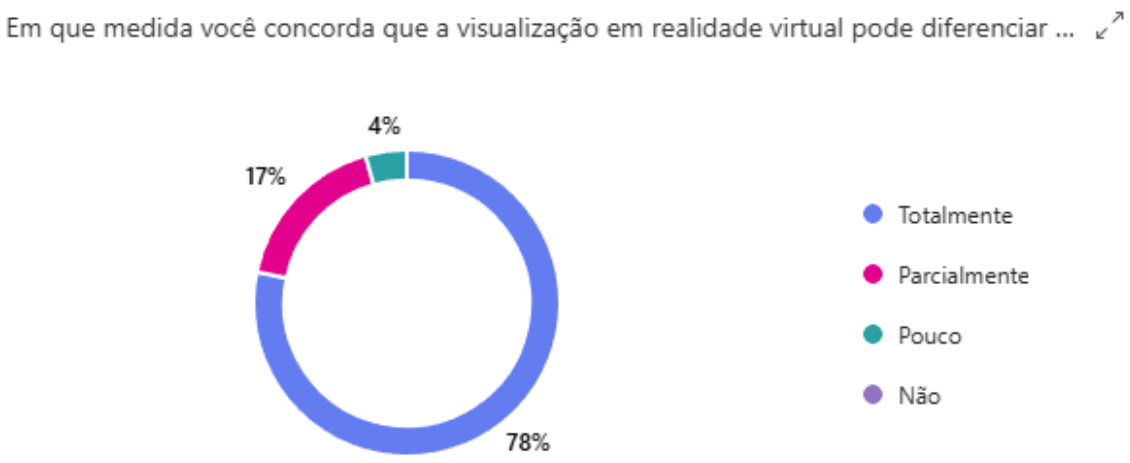
Tabela 7: Em uma escala de 1 a 10, quão importante você considera a utilização da tecnologia de realidade virtual em seu trabalho?

Fonte: Autoria Própria

8- Em que medida você concorda que a visualização em realidade virtual pode diferenciar seus projetos no mercado?

Com essa pergunta é possível ver se o público procura ter um diferencial no mercado.

Tabela 8: Em que medida você concorda que a visualização em realidade virtual pode diferenciar seus projetos no mercado?

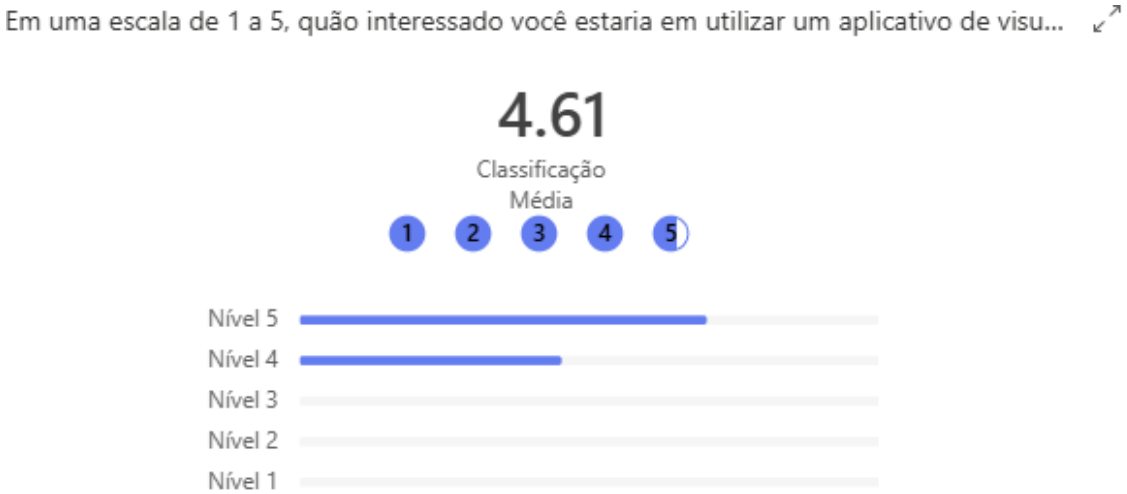


Fonte: Autoria Própria

9- Em uma escala de 1 a 5, quão interessado você estaria em utilizar um aplicativo de visualização em realidade virtual para apresentar seus projetos aos clientes?

Essa pergunta ajuda a ver se nosso projeto gera interesse no público.

Tabela 9: Em uma escala de 1 a 5, quão interessado você estaria em utilizar um aplicativo de visualização em realidade virtual para apresentar seus projetos aos clientes?



Fonte: Autoria Própria

3.1.4 Pesquisa Qualitativa

A Pesquisa Qualitativa explora comportamentos e experiências em profundidade, usando entrevistas e observações para entender motivações e percepções, oferecendo insights detalhados.

Foi utilizada a plataforma Microsoft Forms para conseguir as respostas.

1- Quais são os principais desafios que você enfrenta ao apresentar projetos aos clientes de forma tradicional? (apenas imagens)

Essa pergunta visa entender as dificuldades atuais do público ao apresentar um projeto de forma tradicional.

Tabela 10: Quais são os principais desafios que você enfrenta ao apresentar projetos aos clientes de forma tradicional? (apenas imagens)

anonymous	A falta de interesse por ser algo "cotidiano"
anonymous	Falta de detalhes
anonymous	Falta de visualização por parte dos leigos, principalmente quando se trata de area.
anonymous	ao meu ver, uma apresentação de projetos aos clientes de forma tradicional como desenhos à mão requerem mais técnica, capricho e tempo de produção, e ainda assim, uma representação do projeto em desenho no papel não é tão real quanto uma representação em programas 3D

Fonte: Autoria Própria

2- Que recursos específicos você gostaria de ver em um aplicativo de visualização de projetos arquitetônicos com implementação em realidade virtual para arquitetura?

Essa pergunta ajuda a perceber o que é necessário adicionar e melhorias para o projeto.

Tabela 11: Que recursos específicos você gostaria de ver em um aplicativo de visualização de projetos arquitetônicos com implementação em realidade virtual para arquitetura?

anonymous	Alguns detalhes importantes para um design eficiente
anonymous	Uma boa organização seria excelente
anonymous	Áreas confiáveis e proporcionar uma visualização detalhada em questão de decorados.
anonymous	não sei

Fonte: Autoria Própria

3- Se você já utilizou aplicativos de visualização de projetos arquitetônicos em imagens 360 graus, descreva sua experiência. O que funcionou bem e o que poderia ser melhorado?

Esta pergunta permite que os usuários compartilhem suas experiências passadas com aplicativos de visualização em 360 graus, identificando pontos positivos e áreas para melhorias. Suas respostas podem fornecer insights valiosos para o aprimoramento desses aplicativos.

Tabela 12: Se você já utilizou aplicativos de visualização de projetos arquitetônicos em imagens 360 graus, descreva sua experiência. O que funcionou bem e o que poderia ser melhorado?

anonymous	Não usei ate o momento
anonymous	Sim, a experiência foi boa porem senti falta de um aplicativo para acessar pelo celular mesmo sem o uso do navegador
anonymous	Nunca utilizei
anonymous	Não utilizei

Fonte: Autoria Própria

4- Que tipo de suporte ou recursos adicionais você gostaria de ver em um aplicativo de visualização de projetos arquitetônicos em imagens 360 graus?

Nessa pergunta é possível compreender quais recursos ou suportes adicionais os usuários consideram importantes para melhorar a experiência de uso de aplicativos de visualização em 360 graus. Suas respostas podem guiar o desenvolvimento de futuras funcionalidades ou serviços relacionados.

Tabela 13: Que tipo de suporte ou recursos adicionais você gostaria de ver em um aplicativo de visualização de projetos arquitetônicos em imagens 360 graus?

anonymous	Seria interessante uma integração com o óculos de realidade virtual
anonymous	Não sei
anonymous	acho que uma organização boa ja deixaria o app com melhor uso
anonymous	Nenhuma

Fonte: Autoria Própria

5- Como você armazena seus projetos atualmente?

Por fim, essa pergunta mostra a forma em que o público vem armazenando seus projetos, para se obter uma noção de possíveis concorrentes.

Tabela 14: Como você armazena seus projetos atualmente?

anonymous	Google drive
anonymous	Utilizo a plataforma autocad
anonymous	Google drive
anonymous	Deixo em pastas no meu notebook

Fonte: Autoria Própria

3.2 Levantamento de Requisitos

O levantamento de requisitos define o conjunto de funcionalidades que o sistema deve oferecer, bem como os atributos de qualidade que precisam ser suportados. Ele deve ser elaborado de maneira precisa, completa, consistente e de fácil entendimento para os principais interessados no sistema.

Nesta seção, o objetivo é detalhar os requisitos funcionais e não funcionais do "ARQ360", facilitando a compreensão das necessidades do sistema.

3.2.1 Requisitos funcionais

Os requisitos funcionais descrevem as funcionalidades específicas que o sistema deve oferecer. Esses requisitos são essenciais para garantir que o sistema atenda às necessidades dos usuários e cumpra seus objetivos principais. Abaixo estão os requisitos funcionais identificados:

Figura 5: Requisitos Funcionais 1

Código	Requisito
RF01	O sistema deve permitir que novos usuários criem uma conta no sistema.
RF02	O sistema deve oferecer um mecanismo de autenticação para que os usuários possam fazer login de forma segura.
RF03	O sistema deve permitir que os usuários façam upload de imagens para associá-las a seus perfis ou projetos.
RF04	O sistema deve permitir que os usuários criem e gerenciem assinaturas para acessar funcionalidades exclusivas.
RF05	O sistema deve permitir que os usuários visualizem os detalhes do seu perfil.
RF06	O sistema deve permitir que os usuários editem seus dados pessoais, como nome e e-mail.
RF07	O sistema deve permitir que os usuários alterem suas senhas.
RF08	O sistema deve permitir que os usuários atualizem a foto do perfil.
RF09	O sistema deve oferecer uma funcionalidade para recuperação de senha, caso o usuário esqueça a sua.
RF10	O sistema deve enviar um código de recuperação de senha para o e-mail do usuário.

Fonte: Autoria Própria

Figura 6: Requisitos Funcionais 2

Código	Requisito
RF11	O sistema deve permitir que os usuários visualizem detalhes sobre o seu plano de assinatura.
RF12	O sistema deve permitir que os usuários listem todos os projetos que possuem.
RF13	O sistema deve permitir que os usuários visualizem os detalhes de um projeto específico.
RF14	O sistema deve permitir que os usuários listem todos os clientes associados aos seus projetos.
RF15	O sistema deve permitir que os usuários listem projetos específicos baseados em um cliente.
RF16	O sistema deve permitir que os usuários pesquisem projetos com base no nome do projeto.
RF17	O sistema deve permitir que os usuários criem novos projetos.
RF18	O sistema deve permitir que os usuários adicionem cômodos aos projetos.
RF19	O sistema deve permitir que os usuários adicionem novos clientes.
RF20	O sistema deve permitir que os usuários excluam clientes da sua lista.
RF21	O sistema deve permitir que os usuários excluam projetos.
RF22	O sistema deve permitir que os usuários associem clientes a projetos específicos.
RF23	O sistema deve permitir que os usuários visualizem os detalhes de um cômodo específico dentro de um projeto.
RF24	O sistema deve permitir que os usuários editem os detalhes de um projeto existente.
RF25	O sistema deve permitir que os usuários excluam cômodos de um projeto.

Fonte: Autoria Própria

3.2.2 Requisitos não funcionais

Os requisitos não funcionais descrevem os atributos de qualidade e as condições sob as quais o sistema deve operar. Eles garantem que o sistema seja utilizável, seguro, e de fácil manutenção.

Figura 7: Requisitos não funcionais

Código	Requisito
RNF01	O sistema deve ser intuitivo e fácil de usar, com uma interface amigável para todos os tipos de usuários.
RNF02	O sistema deve garantir a proteção dos dados dos usuários, especialmente durante o processo de autenticação e gerenciamento de senhas.
RNF03	O sistema deve ser capaz de lidar com múltiplos usuários simultâneos sem degradação perceptível de desempenho.
RNF04	O sistema deve ser projetado para acomodar um aumento no número de usuários e projetos sem perda significativa de desempenho.
RNF05	O sistema deve criptografar as senhas dos usuários.
RNF06	O sistema deve ser fácil de manter e atualizar, permitindo que novos recursos sejam adicionados ou problemas sejam corrigidos com mínimo impacto nas operações atuais.

Fonte: Autoria Própria

3.3 Definição de atores e casos de uso

Os atores de um sistema são as entidades que interagem com ele, realizando ações e utilizando suas funcionalidades. Eles podem ser usuários finais, sistemas externos ou dispositivos que se comunicam com a aplicação. A identificação e a descrição dos atores são essenciais para compreender como o sistema será utilizado e quais são as suas principais funcionalidades.

A imagem a seguir apresenta os principais atores do sistema e suas respectivas funções:

Figura 8: Atores

Atores	Descrição
	O Ator Arquiteto é a entidade central do sistema, sendo o principal destinatário das funcionalidades mais relevantes.
	O Ator Cliente atua como uma entidade secundária, cuja principal função é visualizar os projetos do arquiteto.

Fonte: Autoria Própria

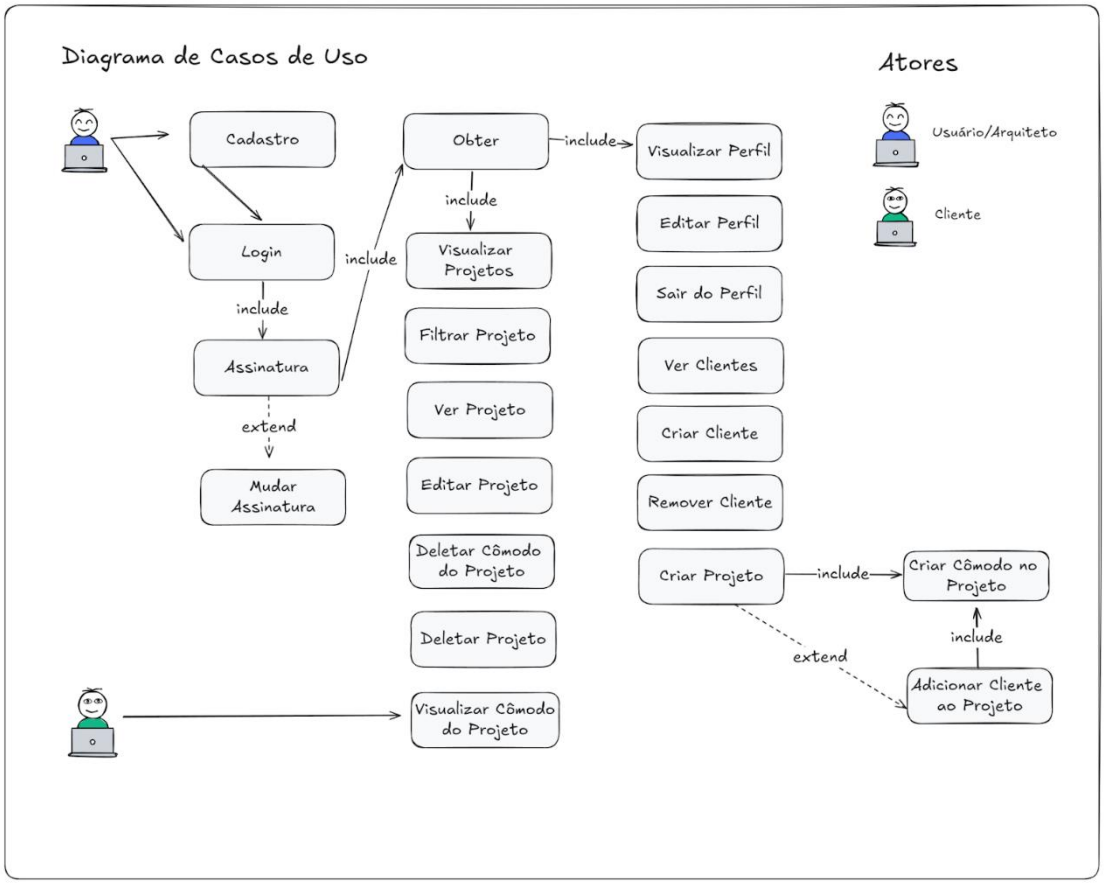
A imagem acima ilustra como esses atores interagem com o sistema. Este mapeamento é fundamental para compreender o fluxo de interação entre as entidades e o sistema, garantindo que todos os requisitos levantados anteriormente sejam atendidos de maneira estruturada e eficiente. A visualização clara das relações entre os atores e suas funcionalidades assegura

que as necessidades de cada entidade sejam contempladas e que a aplicação funcione conforme esperado.

O diagrama de casos de uso é uma ferramenta essencial na modelagem de sistemas, que utiliza a definição dos atores e o levantamento dos requisitos para criar uma representação visual das interações entre os usuários e o sistema. Este diagrama mapeia como os diferentes atores (usuários ou outros sistemas) interagem com o sistema e quais funcionalidades são acionadas por essas interações. Ele organiza essas interações de forma lógica, mostrando o fluxo de ações e a sequência em que ocorrem.

A seguir, uma imagem que mostra o diagrama de casos de uso.

Figura 9: Diagrama de Casos de Uso



Fonte: Autoria Própria

Ao representar essas interações graficamente, o diagrama de casos de uso serve como um mapa para o desenvolvimento do software. Ele ajuda a identificar e esclarecer as principais funcionalidades do sistema e como elas se relacionam com os atores envolvidos. Esse mapeamento é crucial para garantir que todos os requisitos sejam atendidos e que o sistema seja desenvolvido de acordo com as necessidades dos usuários e as especificações definidas.

3.4 Modelagem e Fluxo da Aplicação

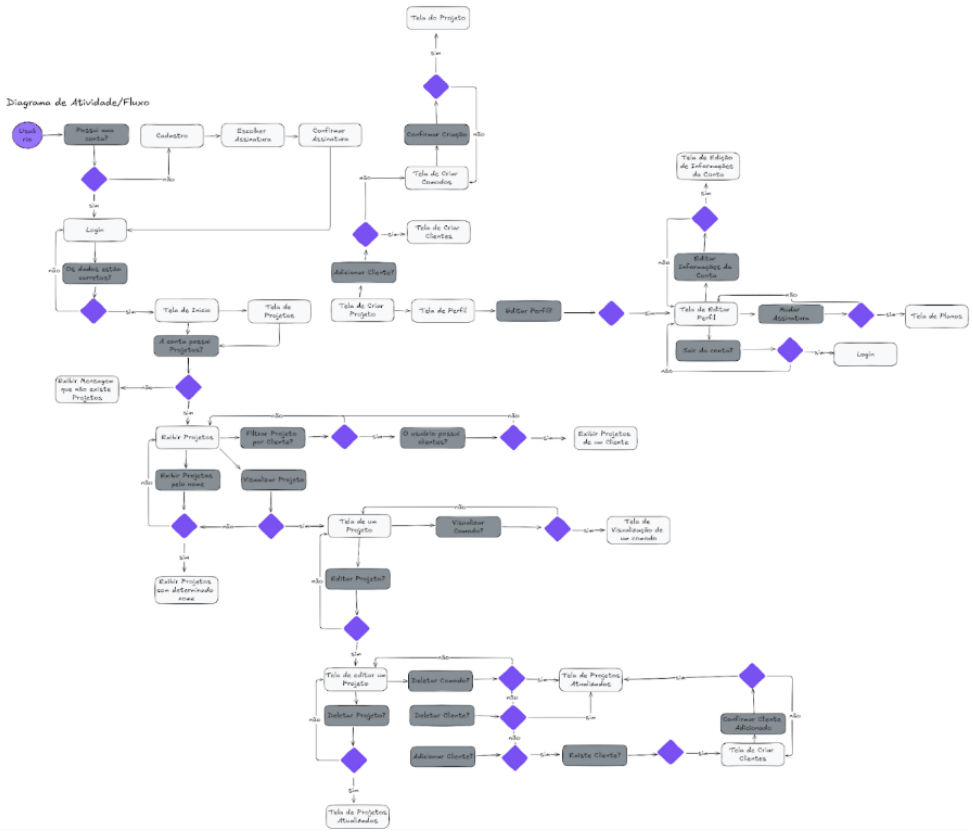
A modelagem e o fluxo da aplicação fornecem uma visão detalhada de como o sistema será estruturado e desenvolvido. Essa etapa é crucial para definir o percurso que a aplicação seguirá, estabelecendo as bases para a sua arquitetura, organização e funcionamento. Ela serve como um guia para o desenvolvimento, garantindo que todas as partes do sistema estejam alinhadas e funcionem de maneira integrada e eficiente.

3.4.1 Diagrama de Atividade

O diagrama de atividades é uma representação visual que ilustra o fluxo de controle e as atividades sequenciais em um sistema. Ele é utilizado para detalhar os processos internos, demonstrando como as diferentes ações e decisões se interligam dentro de um fluxo contínuo. No contexto do desenvolvimento de software, um diagrama de atividades é essencial para mapear o caminho que os usuários ou sistemas seguem ao realizar tarefas, facilitando a compreensão e a documentação de processos complexos.

Abaixo temos a representação dos diagramas de atividade de forma geral.

Figura 10: Diagrama de Atividades



Fonte: Autoria Própria

3.5 Arquitetura em Camadas

Para estruturar o desenvolvimento do software de maneira eficiente e organizada, adotou-se a abordagem de Arquitetura em Camadas. Essa metodologia permite uma separação clara das responsabilidades, facilitando o isolamento de conceitos, a manutenção do sistema, e a implementação de novas funcionalidades. Além disso, promove uma sequência lógica no desenvolvimento das tarefas, garantindo que cada etapa do processo seja executada de forma sistemática e coerente. As camadas foram definidas da seguinte forma:

Figura 11: Diagrama de Atividade

Fonte: Autoria Própria

- Camada de Design

Esta camada é dedicada ao desenvolvimento de toda a interface visual e experiência do usuário, com base nos diagramas de atividades, casos de uso e fluxos de aplicação previamente elaborados. Inclui a criação de layouts, definição de estilos, e a aplicação de princípios de usabilidade que irão guiar a interação dos usuários com o sistema. Todo o design visual do aplicativo e do site é concebido aqui, garantindo que a apresentação estética e funcional atenda aos requisitos estabelecidos.

- Camada de Sistema

Responsável pelo gerenciamento completo dos dados utilizados pelo sistema. Nesta camada, são implementadas as estruturas de banco de dados, bem como as operações de manipulação, armazenamento, recuperação e validação de dados. A lógica de negócios relacionada à integridade dos dados, como regras de validação e controle de acesso, também é tratada aqui. A camada de dados garante que as informações sejam gerenciadas de maneira segura, eficiente e conforme os requisitos funcionais e não funcionais definidos.

- Camada de Dados

Esta camada engloba a lógica de aplicação e a comunicação entre a camada de apresentação com o sistema que também se comunica com a camada de dados. Inclui a implementação de APIs, serviços de autenticação, controle de sessões, e todas as operações que permitem a integração e o funcionamento harmonioso das diferentes partes do sistema. A camada de sistema é fundamental para a execução das regras de negócio e para garantir que os dados sejam corretamente processados e transmitidos entre as camadas de dados e apresentação.

- Camada de Apresentação

A camada de apresentação é onde a interação do usuário com o sistema ocorre. Ela inclui todas as interfaces acessíveis aos usuários finais do aplicativo móvel. Além de fornecer uma interface amigável e responsiva, esta camada é responsável por renderizar os dados fornecidos pela camada de sistema e garantir que as ações do usuário sejam corretamente capturadas e processadas.

Essa organização em camadas não apenas facilita a manutenção e evolução do software, mas também melhora a modularidade, permitindo que mudanças em uma camada não impactem diretamente as demais. Com essa abordagem, o desenvolvimento se torna mais estruturado, permitindo um controle mais

rigoroso sobre cada fase do projeto e assegurando que o produto final atenda às expectativas tanto técnicas quanto de usabilidade.

- Comunicação entre Camadas

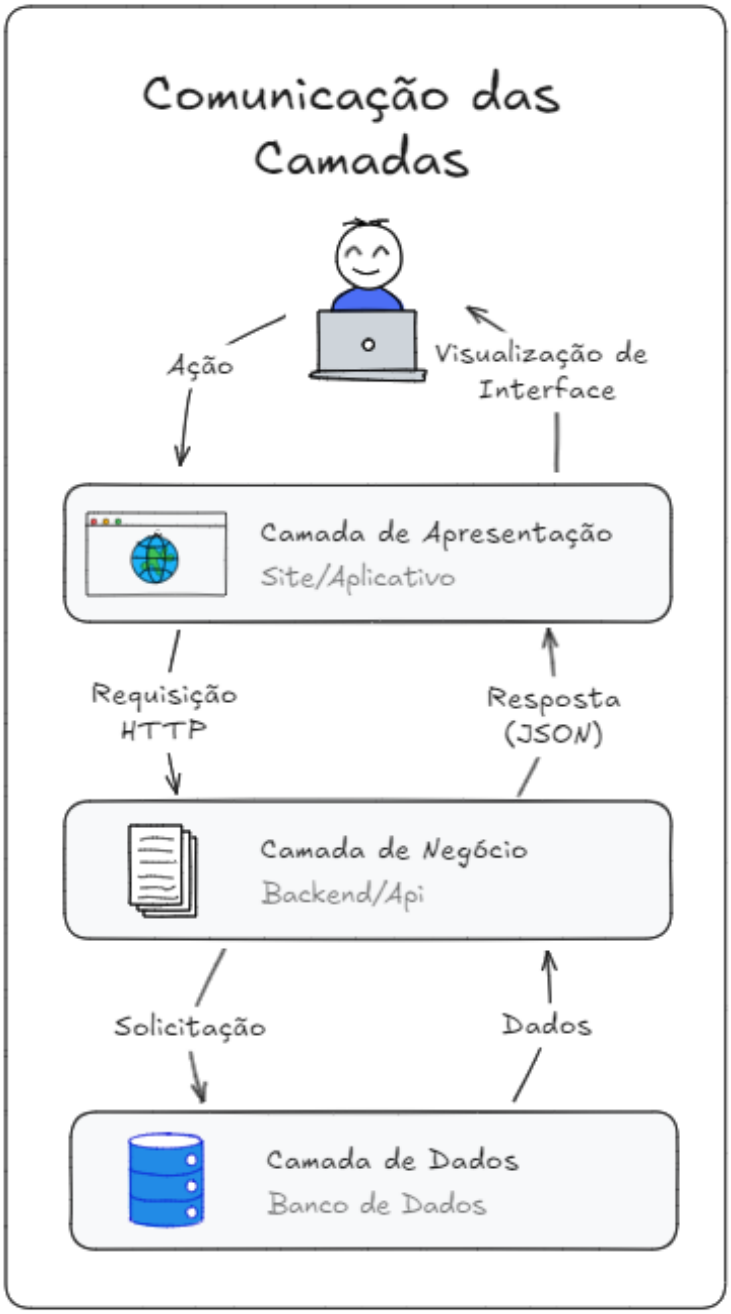
A comunicação entre as diferentes camadas do sistema ocorre de forma direta entre três camadas principais: dados, sistema e apresentação. A camada de design, embora crucial para a estilização, não se comunica diretamente com as outras camadas. Em vez disso, ela influencia indiretamente a camada de apresentação, fornecendo estilos e diretrizes visuais.

Dentro da camada de apresentação, temos dois softwares distintos: um site e um aplicativo móvel. O site atua como uma plataforma de promoção, encorajando os usuários a baixar o aplicativo. O aplicativo se conecta à camada de sistema por meio de uma API (Interface de Programação de Aplicações) que é exposta pelo sistema. Essa API usa o protocolo HTTP para permitir a comunicação entre o aplicativo e o sistema. Quando o aplicativo precisa de dados, ele faz uma requisição HTTP para a API do sistema.

A camada de sistema é responsável por processar essas requisições. Ela se comunica diretamente com a camada de dados para buscar, manipular e validar as informações solicitadas. Depois de obter os dados necessários da camada de dados, a camada de sistema envia a resposta de volta ao aplicativo por meio da API. Assim, a camada de sistema serve como um intermediário, garantindo que o aplicativo receba as informações corretas e atualizadas.

Este fluxo de comunicação é essencial para o funcionamento harmonioso do sistema. Abaixo, uma ilustração que detalha visualmente o fluxo de comunicação entre as camadas para uma compreensão mais clara.

Figura 12: Diagrama de Implementação



Fonte: Autoria Própria

3.6 Ferramentas e Tecnologias Utilizadas

Nesta seção, serão apresentadas as tecnologias, ferramentas e conceitos que serão empregados no desenvolvimento da aplicação.

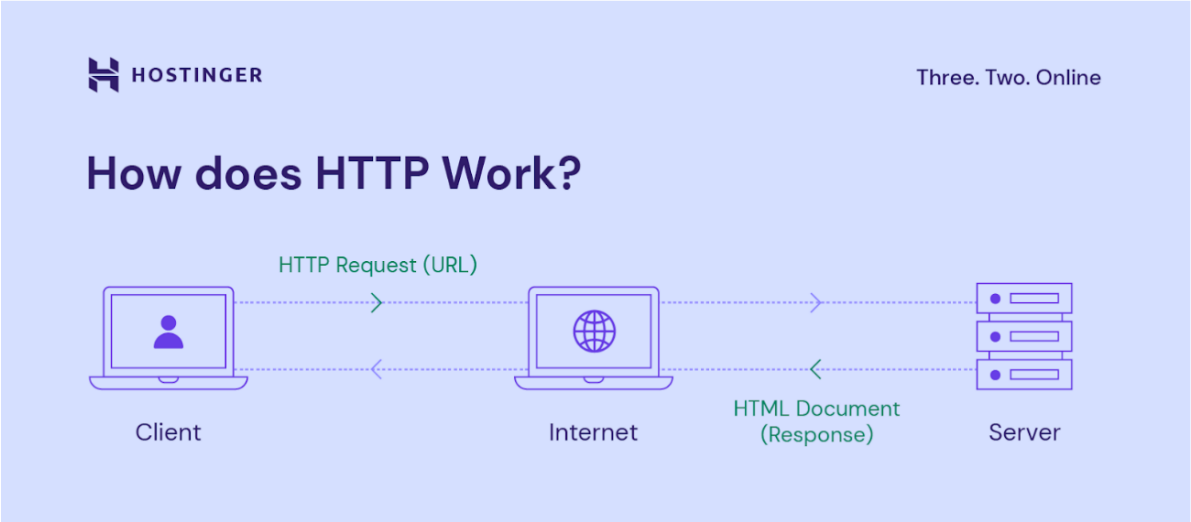
3.6.1 Tecnologias

Abaixo serão apresentadas as tecnologias utilizadas para desenvolver o projeto.

- Protocolo HTTP

O Hypertext Transfer Protocol (HTTP) é um protocolo de comunicação utilizado para a transferência de dados na web. Ele define um conjunto de regras para a troca de informações entre clientes e servidores, permitindo a transmissão de páginas web, imagens e outros recursos.

Figura 13: Procolo HTTP



Fonte: HTTP vs HTTPS: Comparação, Prós e Contras, e Como Configurar (hostinger.com.br)

- API (Application Programming Interface)

Uma API (Interface de Programação de Aplicações) é um conjunto de definições e protocolos que permite a comunicação entre diferentes sistemas de software. Ela define as formas e os métodos através dos quais um programa pode interagir com outro, facilitando a integração e a utilização de serviços e funcionalidades externas. APIs são amplamente usadas para a troca de dados entre o frontend e o backend de aplicações web e móveis.

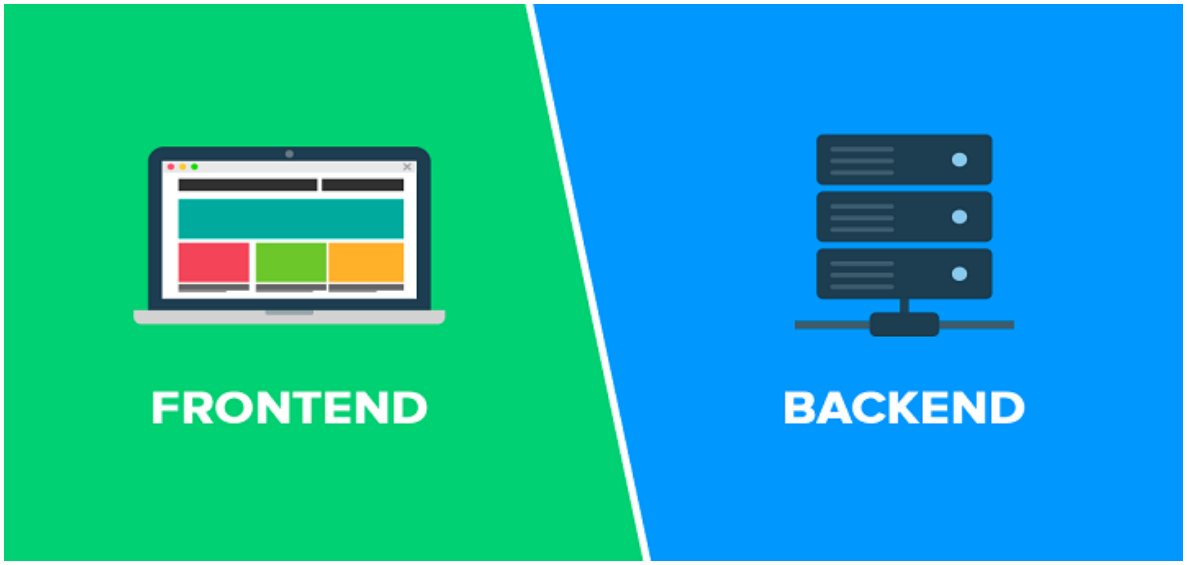
Figura 14: API

Fonte: Integração de APIs (rpvtecnologia.com.br)

- Backend

O backend refere-se à parte do desenvolvimento de software que lida com o processamento e gerenciamento de dados e lógica de negócios em um sistema. É a camada do software que opera nos servidores, manipulando requisições, processando dados, e interagindo com o banco de dados. O backend é responsável por fornecer APIs e serviços que suportam a funcionalidade do frontend, garantindo a lógica de aplicação e a persistência de dados.

Figura 15: Backend

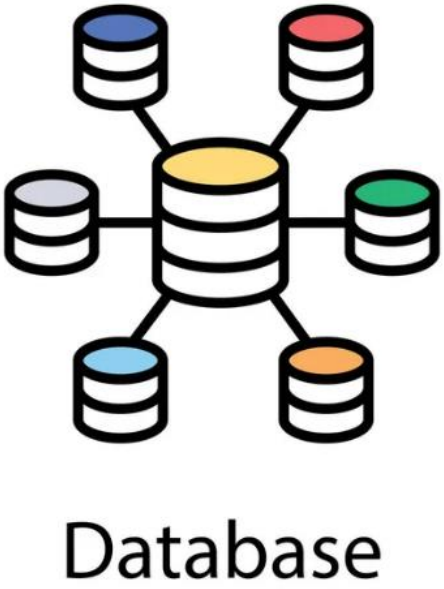


Fonte: ¿Qué es un backend en una app móvil? Te explicamos lo que es - Owius

- Banco de Dados

Um banco de dados é um sistema organizado para armazenar, gerenciar e recuperar dados de forma eficiente. Ele permite a persistência de informações e a execução de operações como inserção, atualização e consulta de dados. Os bancos de dados podem ser relacionais, como o PostgreSQL, que organiza os dados em tabelas e utiliza SQL para consultas, ou não relacionais, que armazenam dados de maneira mais flexível. A escolha do banco de dados depende das necessidades específicas da aplicação e do tipo de dados a serem gerenciados.

Figura 16: Banco de Dados

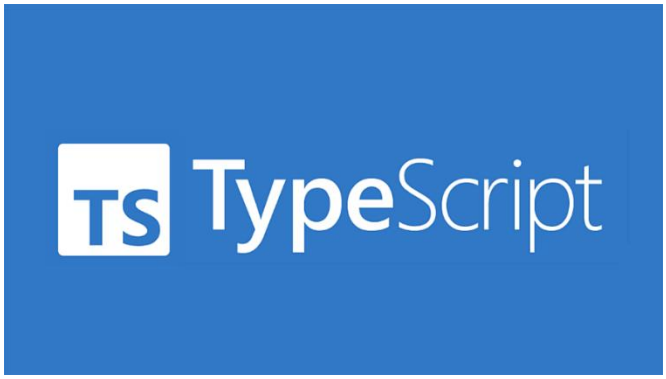


Fonte: Flat Design Data Warehouse Diagram: vetor stock (livre de direitos) 1135592927 | Shutterstock

- TypeScript

TypeScript é uma linguagem de programação desenvolvida pela Microsoft, que é um superconjunto do JavaScript. Adiciona tipagem estática opcional e outros recursos avançados, como classes e interfaces, que ajudam a melhorar a manutenção e a escalabilidade de projetos JavaScript, especialmente em aplicações grandes e complexas.

Figura 17: Typescript



Fonte: O que há de novo no TypeScript 4.7 (lsantos.dev)

- React Native

React Native é um framework de desenvolvimento de aplicações móveis criado pelo Facebook. Permite que desenvolvedores construam aplicações nativas para iOS e Android utilizando JavaScript e a biblioteca React. Com React Native, é possível compartilhar a maior parte do código entre plataformas, o que reduz o tempo de desenvolvimento.

Figura 18: React Native

Fonte: Curso gratis para desarrollar aplicaciones móviles con React Native (facialix.com)

- Expo

Expo é uma plataforma que facilita o desenvolvimento com React Native, oferecendo um conjunto de ferramentas e serviços para criar, testar e publicar aplicações móveis. Ele fornece uma série de bibliotecas e APIs prontas para uso, além de um ambiente de desenvolvimento simplificado.

Figura 19: Expo

Fonte: Unlocking the Power of Expo: Why You Should Use It for React Native Development in 2024 | by Sanjin Sehic | ScaleUp | Medium

- Tailwind CSS

Tailwind CSS é um framework de design para desenvolvimento web que utiliza uma abordagem de utilitários. Em vez de criar estilos personalizados, Tailwind CSS oferece classes pré-definidas para estilizar componentes de maneira rápida e eficiente, promovendo a criação de interfaces responsivas e customizáveis.

Figura 20: Tailwind CSS

Fonte: Mastering Tailwind CSS Customization | Mario Yonan

- Three.js

Three.js é uma biblioteca JavaScript que facilita a criação e a renderização de gráficos 3D em navegadores web. Baseada em WebGL, ela permite a criação de ambientes e objetos tridimensionais, tornando-a uma escolha popular para o desenvolvimento de visualizações 3D e jogos na web.

Figura 21: Three.js

Fonte: You Need To Dive Into Three.JS… Here's Why. | by mohamed | Medium

- ReactJS

ReactJS é uma biblioteca JavaScript desenvolvida pelo Facebook para a construção de interfaces de usuário. Baseada no conceito de componentes reutilizáveis, ela permite a criação de aplicações web interativas e de alto desempenho, facilitando a manutenção e a escalabilidade de grandes projetos.

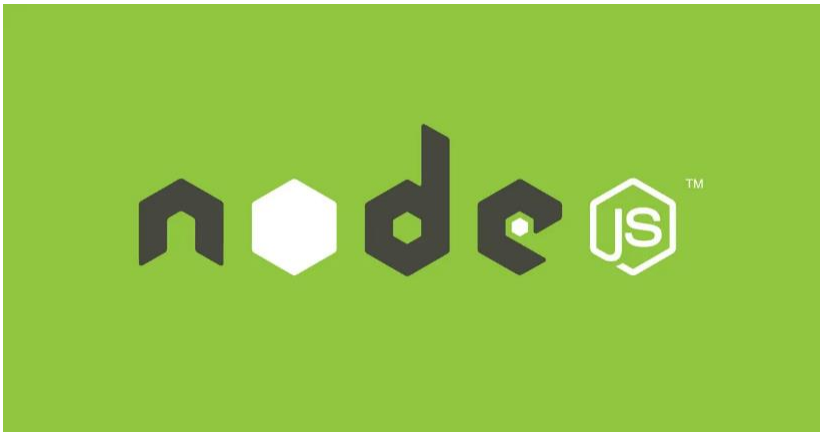
Figura 22: React.js

Fonte: React js - EcuRed

- Node.js

Node.js é um ambiente de execução JavaScript do lado do servidor, construído sobre o motor V8 do Google Chrome. Ele permite o desenvolvimento de aplicações web escaláveis e eficientes, utilizando JavaScript tanto no front-end quanto no back-end, aproveitando sua natureza assíncrona e baseada em eventos.

Figura 23: Node.js



Fonte: Node JS Banner - Nextflow

- NestJS

NestJS é um framework para desenvolvimento de aplicações backend em Node.js, que utiliza TypeScript como sua linguagem principal. Ele é baseado em conceitos de programação orientada a objetos e fornece uma arquitetura modular e extensível, ideal para a construção de sistemas escaláveis e robustos.

Figura 24: NestJS

Fonte: Mastering Data Management: Exploring the Repository Pattern in NestJS | by Yoyoplenty | Stackademic

- Prisma

Prisma é uma ferramenta de mapeamento objeto-relacional (ORM) para Node.js e TypeScript, que facilita a interação com bancos de dados relacionais. Ele oferece uma camada de abstração entre a aplicação e o banco de dados, proporcionando um sistema de tipos robusto e uma API intuitiva para consultas e manipulação de dados.

Figura 25: Prisma

Fonte: Aprenda a utilizar o Prisma.io com Node.js - Nine Labs

- Stripe

Stripe é uma plataforma de pagamentos online que oferece uma API para integrar pagamentos em websites e aplicativos móveis. Ela suporta uma ampla gama de métodos de pagamento e fornece ferramentas para gerenciar transações, assinaturas e faturamento, simplificando o processo de processamento de pagamentos.

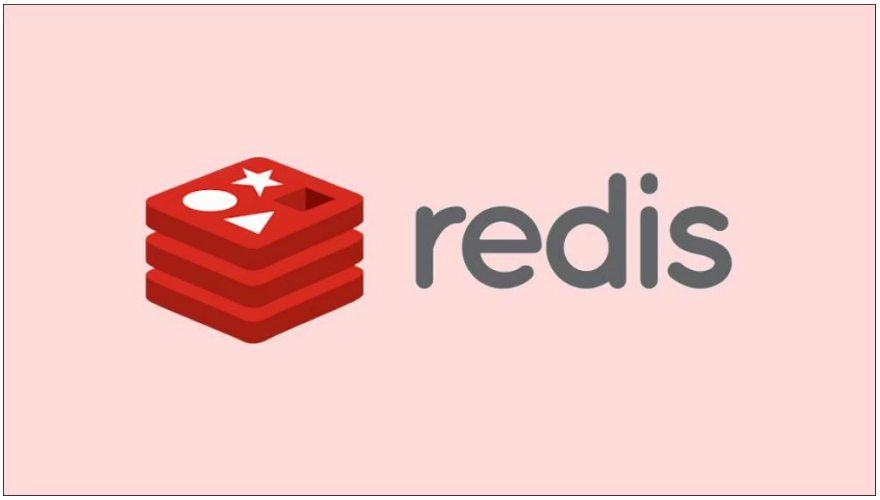
Figura 26: Stripe

Fonte: The Growth Marketing Behind Stripe’s Success (nbtdigital.com)

- Redis

Redis é um banco de dados em memória, baseado em estrutura de dados, que oferece alta performance para operações de leitura e escrita. Utilizado principalmente como cache ou para armazenar dados temporários, ele suporta várias estruturas de dados como strings, listas e conjuntos, e é amplamente empregado para otimização de aplicações web.

Figura 27: Redis



Fonte: How to Quickly Deploy Redis as a Docker Container (howtogeek.com)

- PostgreSQL

PostgreSQL é um sistema de gerenciamento de banco de dados relacional de código aberto, conhecido por sua robustez e conformidade com padrões SQL. Oferece suporte a tipos de dados avançados, como JSON e XML, e é utilizado para armazenar e gerenciar dados de aplicações complexas e de grande escala.

Figura 28: Postgre SQL

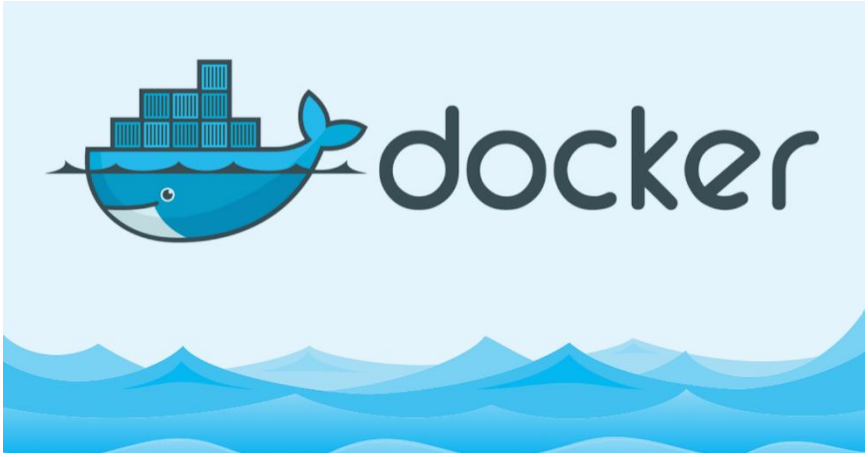
Fonte: PostgreSQL logo and symbol, meaning, history, PNG (1000logos.net)

3.6.2 Ferramentas

- Docker

Docker é uma plataforma para a criação, distribuição e execução de aplicações em contêineres. Ele permite que desenvolvedores empacotem aplicações e suas dependências em contêineres isolados, garantindo consistência entre diferentes ambientes e facilitando o gerenciamento e a escalabilidade em sistemas complexos.

Figura 29: Docker



Fonte: What is Docker? A Revolutionary Change in Cloud Computing | Logz.io

- Visual Studio Code

Visual Studio Code (VS Code) é um editor de código-fonte desenvolvido pela Microsoft. Ele é amplamente utilizado por desenvolvedores devido à sua interface intuitiva, extensibilidade e suporte a uma variedade de linguagens e ferramentas, tornando-o ideal para desenvolvimento web e de software.

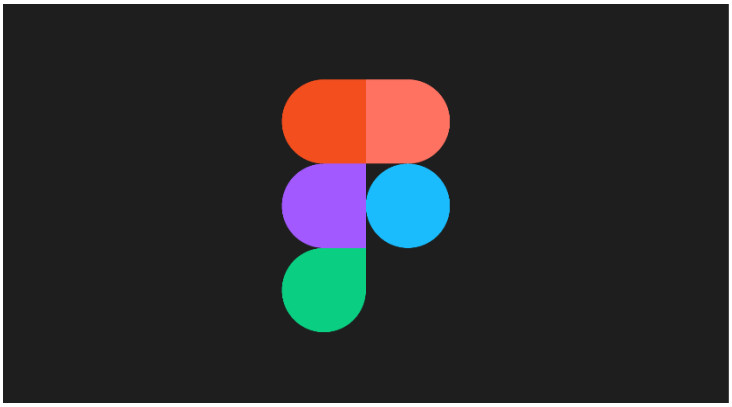
Figura 30: Visual Studio Code

Fonte: Program | Microsoft Learn

- Figma

Figma é uma ferramenta de design de interfaces e prototipagem baseada em nuvem. Permite que designers criem e colaborem em projetos de design de forma interativa, com recursos para criação de layouts, protótipos e design de interfaces, facilitando a comunicação e a integração entre equipes de design e desenvolvimento.

Figura 31: Figma



Fonte: Figma LOGO | Figma

4.0 DESENVOLVIMENTO DO APLICATIVO E SITE

Essa seção abordará como foi realizado o desenvolvimento do aplicativo e site propostos para este projeto.

4.1 Planejamento

Para obter uma boa organização no dia a dia sem se perder do foco do trabalho, é essencial ter uma maneira de documentar e adicionar prazos as tarefas. Para conseguir tal organização foi utilizada a Plataforma Jira.

O Jira é uma ferramenta de gestão ágil de projetos usada pelas equipes para planejar, monitorar, lançar e dar suporte a softwares de alto nível com confiança.

O planejamento foi iniciado estabelecendo um cronograma para realizar as tarefas no tempo correto sem mais atrasos que eram recorrentes.

Figura 32: Cronograma

Fonte: Autoria Própria

Além do cronograma, foi feita uma divisão de tarefas para que cada integrante consiga mostrar seu melhor e desempenhar melhor seu papel no projeto.

A integrante Flávia ficou responsável pela parte do Design da aplicação e toda a Monografia.

Figura 33: Tarefas Flávia

Chave	Resumo	Status	Responsável
TAREFA-2	INTRODUÇÃO	CONCLUÍDO	flaviachristinehp
TAREFA-48	Objetivo Geral	CONCLUÍDO	flaviachristinehp
TAREFA-49	Objetivo Especificado	CONCLUÍDO	flaviachristinehp
TAREFA-5	FUNDAMENTAÇÃO TEÓRICA	CONCLUÍDO	flaviachristinehp
TAREFA-15	Arquitetura	CONCLUÍDO	flaviachristinehp
TAREFA-16	Realidade Virtual	CONCLUÍDO	flaviachristinehp
TAREFA-17	Visualização 360 na Arquitetura	CONCLUÍDO	flaviachristinehp
TAREFA-18	Tecnologia na Arquitetura	CONCLUÍDO	flaviachristinehp
TAREFA-19	Armazenamento de Projetos Arquitetônicos	CONCLUÍDO	flaviachristinehp
TAREFA-20	Impacto da Tecnologia na Comunicação entre Arquitetos e C...	CONCLUÍDO	flaviachristinehp
TAREFA-7	METODOLOGIA	CONCLUÍDO	flaviachristinehp

Fonte: Autoria Própria

O integrante Israel ficou responsável pelo Desenvolvimento do Software e contribuiu com a Monografia.

Figura 34: Tarefas Israel

TAREFA-8	APRESENTAÇAO DO PROJETO	CONCLUÍDO	Israel Cruz
TAREFA-9	CONCLUSÃO	CONCLUÍDO	Israel Cruz
TAREFA-21	DESENVOLVIMENTO	CONCLUÍDO	Israel Cruz
TAREFA-50	DESENVOLVIMENTO DOS DIAGRAMAS	CONCLUÍDO	Israel Cruz
TAREFA-59	DESENVOLVIMENTO DO BANCO DE DADOS	CONCLUÍDO	Israel Cruz
TAREFA-62	CRIAÇÃO DOS REPOSITORIOS	CONCLUÍDO	Israel Cruz
TAREFA-63	CRIAÇÃO DO BACKEND	CONCLUÍDO	Israel Cruz
TAREFA-72	CRIAÇÃO DO APLICATIVO	CONCLUÍDO	Israel Cruz
TAREFA-83	CRIAÇÃO DO SITE PARA DIVULGAÇÃO	CONCLUÍDO	Israel Cruz
TAREFA-85	Documentação	CONCLUÍDO	Israel Cruz
TAREFA-86	Aplicativo	CONCLUÍDO	Israel Cruz
TAREFA-88	Banco de Dados	CONCLUÍDO	Israel Cruz

Fonte: Autoria Própria

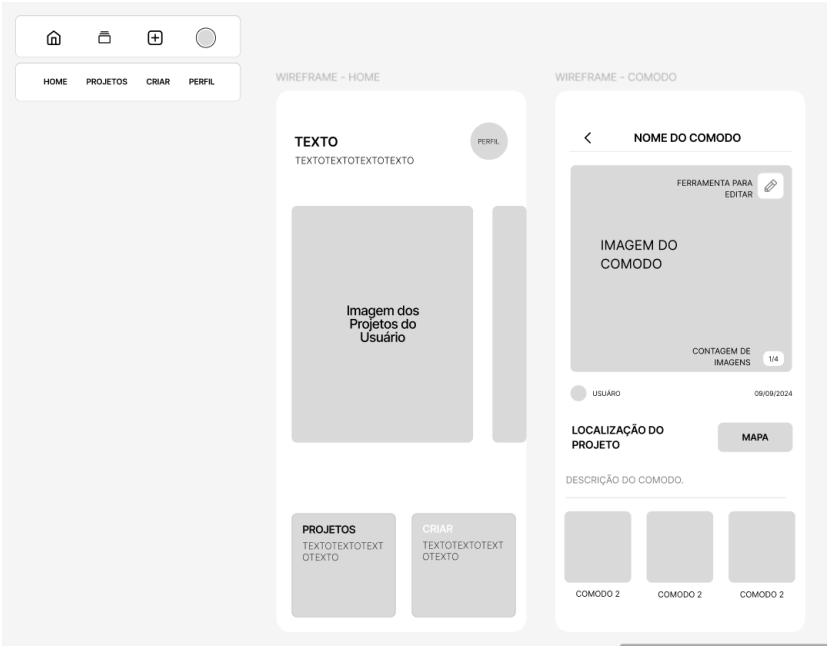
4.2 Camada do Design

O design de aplicativos visa criar interfaces de usuário intuitivas e eficientes para dispositivos móveis, como celulares e tablets. Esse processo envolve a consideração de diversos fatores essenciais, como usabilidade, experiência do usuário (UX), arquitetura da informação e design visual. Um design bem executado não apenas torna o aplicativo visualmente atraente, mas também facilita a navegação e o uso, promovendo uma interação mais agradável e eficiente.

4.2.1 Ideia do Protótipo do Aplicativo

Foi iniciado o projeto com o objetivo de criar uma interface que transmita conforto e modernidade, associando-se a casas e imóveis, enquanto mantemos um conceito minimalista para garantir uma excelente usabilidade e compreensão imediata por parte dos usuários. Com isso em mente, foi desenvolvida uma estrutura preliminar do design para o aplicativo e o site, a fim de orientar nosso processo. A figura abaixo ilustra como foi pensada a estrutura do aplicativo.

Figura 35: Wireframes



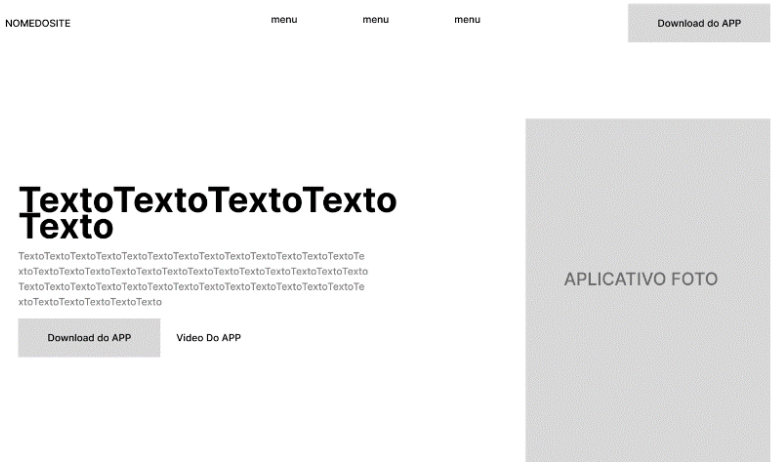
Fonte: Autoria Própria

4.2.2 Ideia do Protótipo do Site

Além do aplicativo, foi necessário desenvolver um site para disponibilizar o download da ferramenta, uma solução ideal considerando o orçamento limitado do projeto. O site manteve a mesma estética visual do aplicativo e adotou uma estrutura voltada para a promoção, destacando todas as funcionalidades e benefícios oferecidos pela plataforma.

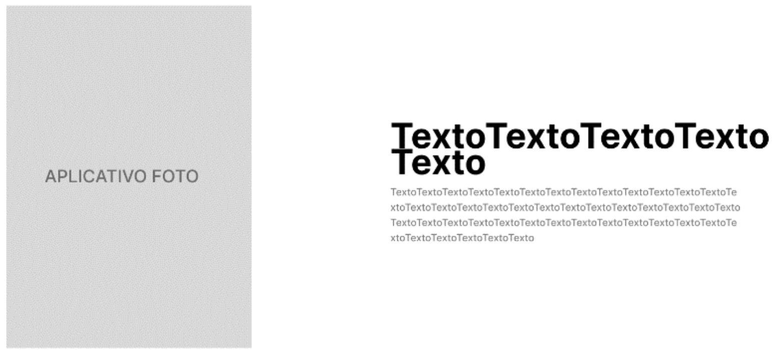
A figura abaixo ilustra a estrutura projetada para o site.

Figura 36: Wireframes Site



Fonte: Autoria Própria

Figura 37: Wireframes Site 2



Fonte: Autoria Própria

Figura 38: Wireframes Site 3

Fonte: Autoria Própria

4.2.3 Teoria das Cores

A teoria das cores é um conjunto de princípios e diretrizes que orientam a escolha e a combinação de cores em projetos visuais. No projeto foram escolhidas as cores que remetessem a conforto, sofisticação, minimalismo e autoridade.

As cores principais usadas no projeto foram

- marrom, que transmite estabilidade, conforto e conexão com a natureza;
- branco que transmite pureza, clareza e sofisticação;
- preto que transmite sofisticação, autoridade e mistério.

Figura 39: Logo ARQ360

Fonte: Autoria Própria

Figura 40: Paleta de Cores

Fonte: #655345 Color Info - Coolors

4.2.4 UX/UI

UX, ou Experiência do Usuário, é o processo de criar produtos e serviços que ofereçam uma interação positiva e eficiente para os usuários. Foca em garantir que o produto seja fácil de usar, acessível a todos e visualmente agradável.

Ao desenvolver UI/UX (Interface e Experiência do Usuário), é fundamental criar uma interface que elimine estresse, dúvidas ou inseguranças no usuário. O objetivo principal é garantir uma interação fluida, eficiente e positiva, onde o produto seja fácil de usar, acessível para todos e visualmente atraente. A harmonia entre funcionalidade e design contribui para uma experiência satisfatória, evitando frustrações e promovendo a confiança do usuário.

Como parâmetros para este projeto, foram utilizadas duas normas essenciais no design de aplicações, sendo elas:

As Heurísticas de Nielsen, que compõem dez princípios gerais que servem como diretrizes para a avaliação e o design de interfaces de usuário, visando otimizar a usabilidade

1. Visibilidade de status do sistema
2. Correspondência entre o sistema e o mundo real
3. Liberdade de controle fácil para o usuário
4. Consistência e padrões
5. Prevenções de erros
6. Reconhecimento em vez de memorização
7. Flexibilidade e eficiência de uso
8. Estética e design minimalista
9. Ajuda aos usuários a reconhecerem, diagnosticarem e recuperarem-se de erros
10. Ajuda e documentação

E as Heurísticas de Steve Krug presentes no livro “Não me faça pensar”, o livro ensina como criar sites e aplicativos intuitivos, para que os usuários compreendam facilmente como usá-los, sem esforço

1. Mantenha as coisas simples: O princípio básico do livro é que a navegação em um site ou aplicação deve ser óbvia, de forma que o usuário não precise "pensar" sobre como usá-la.
2. Teste de usabilidade é essencial: Krug argumenta que testes simples de usabilidade, mesmo com poucos usuários, são eficazes para identificar problemas e melhorar a experiência.
3. Design intuitivo: Bons designs de interface facilitam a navegação e minimizam a necessidade de decisões complicadas. O design deve ser funcional e visualmente claro.

4. Não complique o texto: As informações e as funcionalidades devem ser diretas e claras, evitando termos técnicos ou longos textos que possam confundir o usuário.
5. Padrões de navegação: A consistência e a familiaridade são essenciais para uma boa experiência de usuário. É melhor seguir convenções conhecidas do que tentar reinventar a roda.

Ao aplicar essas normas de design, a aplicação se torna mais detalhada, eficiente e capaz de gerar melhores resultados, atendendo a todos os públicos de forma mais eficaz.

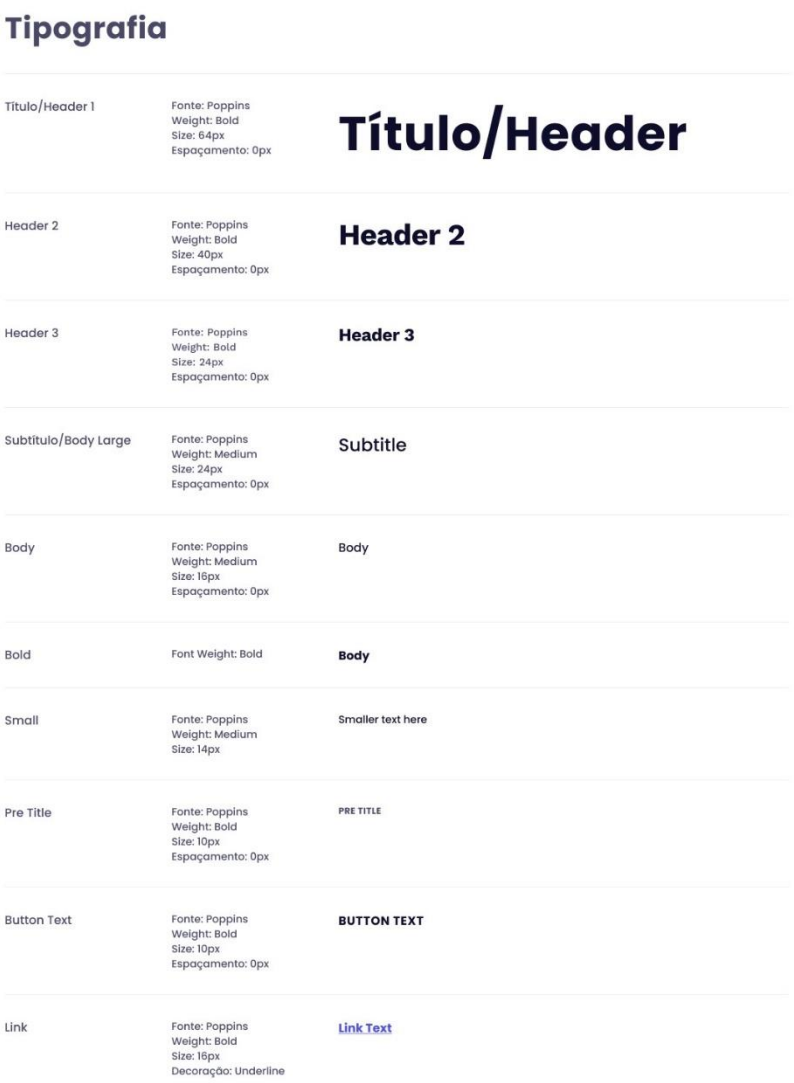
4.2.5 Design System

Um Design System é um conjunto de diretrizes, componentes e padrões que definem a identidade visual e funcional de um produto digital.

No projeto, foi estabelecido um design system desde o início para orientar de forma clara e eficiente a camada de design. Isso facilita a consistência visual e funcional, além de agilizar o processo de desenvolvimento.

A imagem abaixo ilustra o design system das fontes utilizadas ao longo do projeto, destacando a escolha tipográfica que garante consistência e legibilidade em toda a interface.

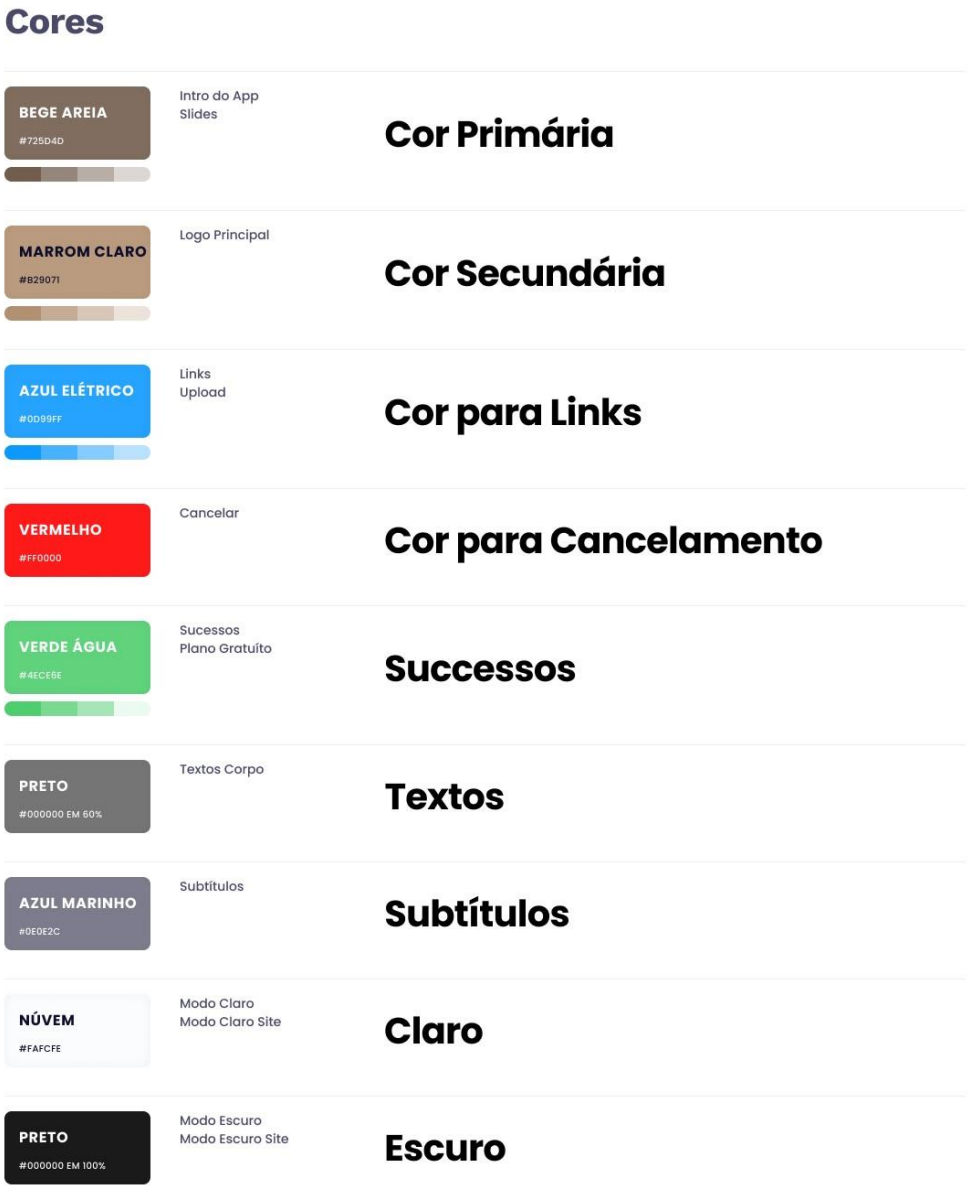
Figura 41: Tipografia



Fonte: Autoria Própria.

A imagem abaixo apresenta o design system das cores selecionadas para o projeto, destacando a paleta que assegura consistência visual e harmonia em toda a interface.

Figura 42: Paleta de Cores 2

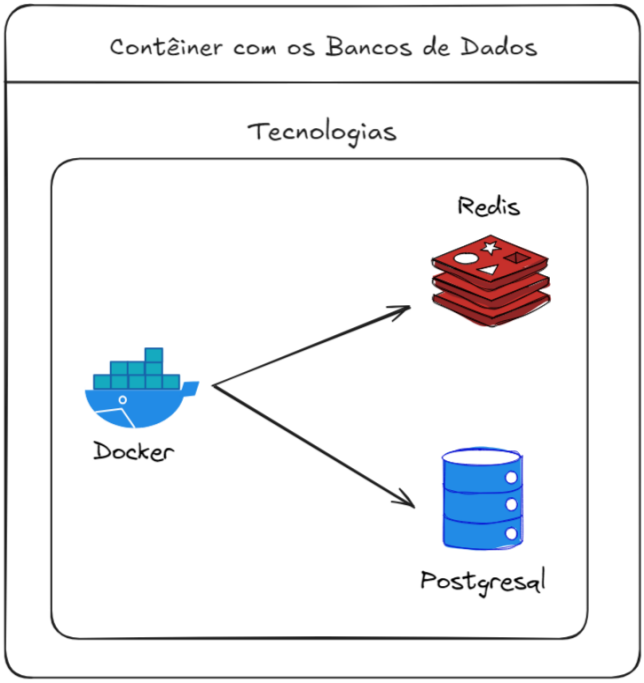


Fonte: Autoria Própria.

4.3 Camada de Dados

Para desenvolver a camada de dados, foi feita uma análise das necessidades da aplicação com base nas ideias já concebidas. Foi decidido primeiro selecionar as tecnologias que comporiam essa camada antes de estruturar o banco de dados em si. Optou-se pelo PostgreSQL como banco de dados principal por sua natureza relacional, e pelo Redis como um banco intermediário para cache de dados. Para garantir maior segurança e praticidade no ambiente de desenvolvimento, foram implementados os serviços em containers usando Docker, permitindo assim maior escalabilidade. Para facilitar a replicação do ambiente em diferentes cenários, foi criado um arquivo `docker-compose` que pode ser utilizado em qualquer ambiente. Abaixo apresenta uma representação ilustrada do contêiner.

Figura 43: Contêiner com os Bancos de Dados

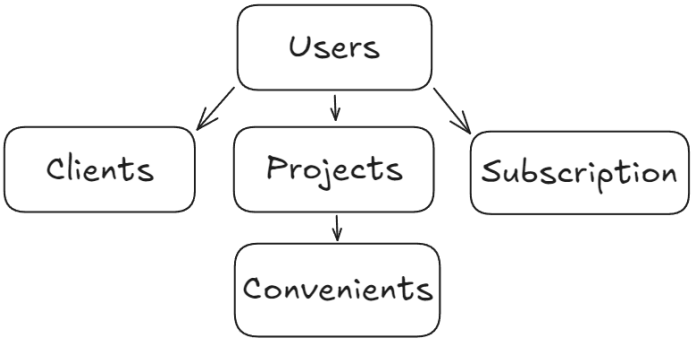


Fonte: Autoria Própria.

4.3.1 Entidades e Relacionamentos

Com base nos atores, casos de uso e na prototipação da aplicação, foi desenvolvido um conceito de entidades que representam a ideia em formato de dados a serem armazenados no banco de dados. A imagem abaixo ilustra essas entidades.

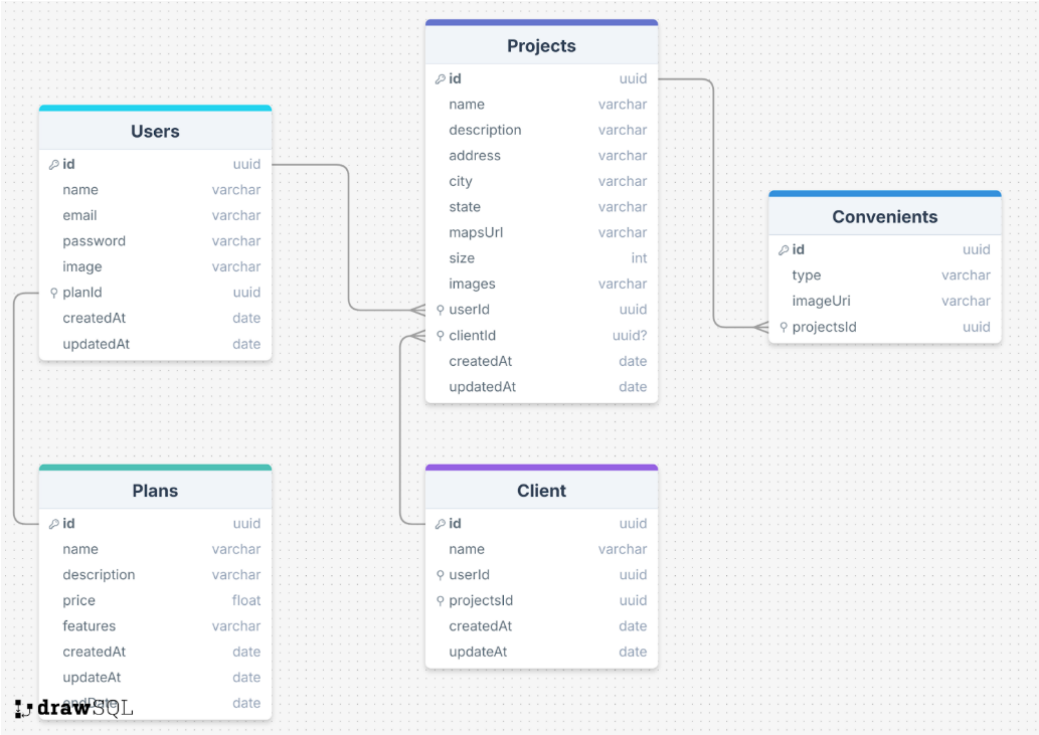
Figura 44: Contêiner de Banco de Dados



Fonte: Autoria Própria.

Após revisar o conceito, foram estruturadas as entidades de maneira organizada, garantindo que todas as relações entre elas estivessem devidamente estabelecidas. A imagem a seguir mostra essas entidades e seus relacionamentos.

Figura 45: Entidades e Relacionamentos



Fonte: Autoria Própria.

4.3.2 Implementação

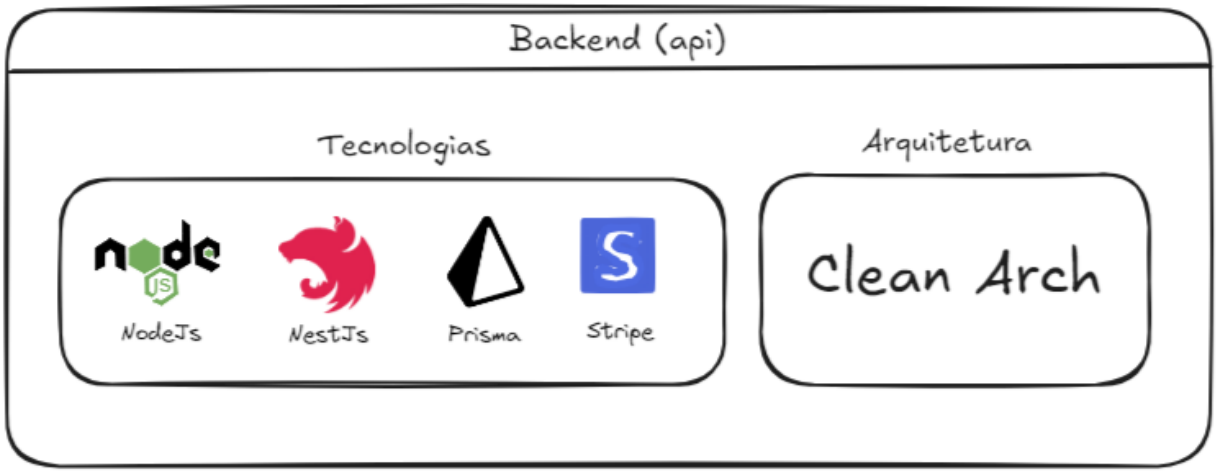
Para a implementação, foi utilizada o Prisma, que se conecta ao container do Docker e permite uma implementação rápida, eficiente e traz um suporte para comunicação com outras camadas.

4.4 Camada do Sistema

Na camada de sistema, foi desenvolvida uma API que utiliza os protocolos HTTP para gerenciar a comunicação entre as diferentes camadas do projeto. Para essa implementação, optou-se por usar Node.js, que permiti a execução do código no lado do servidor de forma eficiente. A escolha da linguagem de programação recai sobre TypeScript, devido à sua robustez e suporte a tipagem estática, o que facilita a manutenção e escalabilidade do código.

Adicionalmente, o framework NestJS foi empregado para a construção da API. O NestJS é um framework altamente opinativo que segue boas práticas de desenvolvimento e arquiteturas modernas. Isso garante que a API não apenas se mantenha organizada e modular, mas também facilite a escalabilidade e a evolução contínua do sistema ao longo do tempo.

Figura 46: Detalhes do Backend

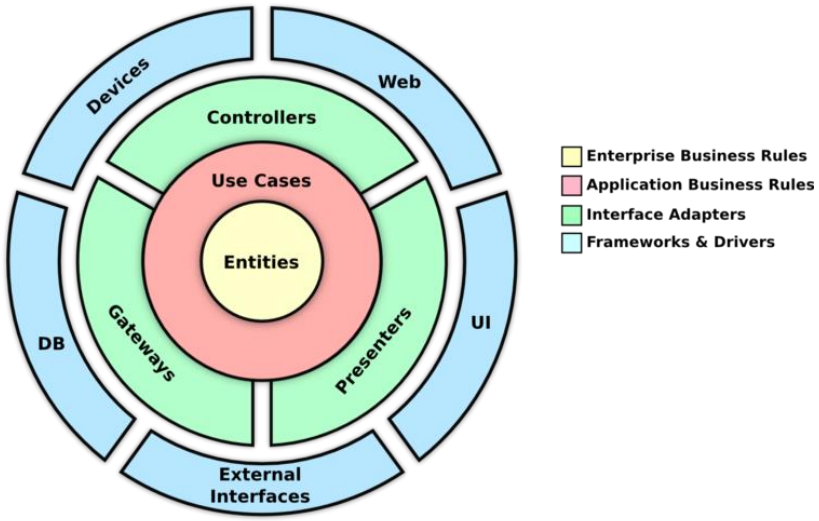


Fonte: Autoria Própria.

4.4.1 Arquitetura do sistema

A arquitetura do sistema foi projetada com o objetivo de garantir escalabilidade e facilidade de manutenção, adotando o padrão de arquitetura desenvolvido por Robert C. Martin, conhecido como "Uncle Bob", e amplamente reconhecido como Clean Architecture. Este padrão promove uma separação clara das responsabilidades dentro do sistema, o que facilita a manutenção, melhora a legibilidade e assegura a adesão às boas práticas e design patterns estabelecidos.

Figura 47: Representação da Clean Architecture



Fonte: Autoria Própria.

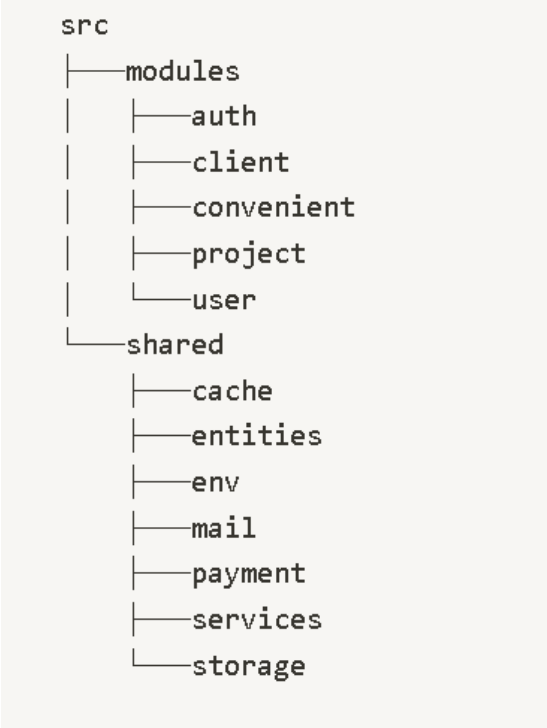
Além da Clean Architecture, a estrutura do sistema incorpora o conceito de arquitetura modular. Isso significa que cada entidade e funcionalidade do sistema são isoladas em módulos distintos, o que simplifica a identificação e o gerenciamento das partes do sistema. Essa abordagem modular não só melhora a organização do código, mas também facilita a implementação de alterações e a integração de novas funcionalidades sem impactar outras áreas do sistema.

Com base nesses princípios arquiteturais, a estrutura de pastas do sistema foi organizada de maneira a refletir essas escolhas de design. A disposição das pastas segue uma estrutura que facilita a navegação e a compreensão dos diferentes componentes e suas interações, promovendo uma organização lógica e eficiente. A estrutura final das pastas do sistema é a seguinte:

4.4.2 Estrutura das pastas

Com base nos princípios arquiteturais, a estrutura de pastas do sistema foi organizada de maneira a refletir essas escolhas de design. A disposição das pastas segue uma estrutura que facilita a navegação e a compreensão dos diferentes componentes e suas interações, promovendo uma organização lógica e eficiente. A estrutura final das pastas do sistema é a seguinte:

Figura 48: Estrutura de Pastas da API



Fonte: Autoria Própria

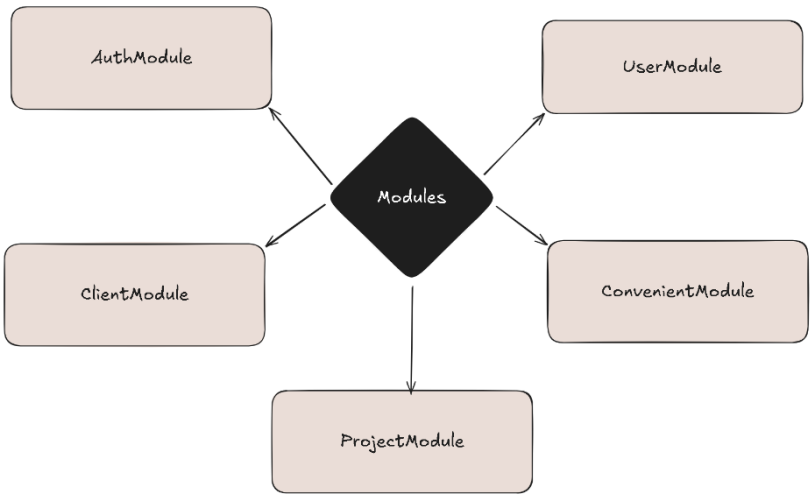
A estrutura das pastas é organizada em duas pastas principais: modules e shared. A seguir, é possível ver detalhadamente como cada uma foi estruturada e como utilizaram a arquitetura já definida.

Modules

No contexto de nossa arquitetura, módulos são unidades fundamentais que agrupam funcionalidades relacionadas a um domínio específico. Na pasta modules, esses módulos representam os domínios estruturados no banco de dados, mas com camadas adicionais que gerenciam as regras de negócio, os endpoints e a interação com o banco de dados para cada domínio.

Abaixo, será detalhado o padrão de estrutura adotado para cada módulo:

Figura 49: Estrutura de Pastas da API 2



Fonte: Autoria Própria

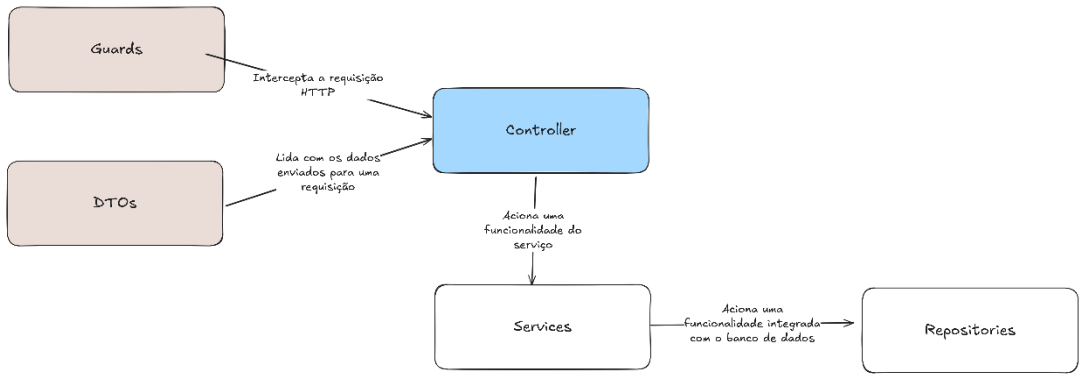
- Controllers: São responsáveis por gerenciar as requisições HTTP. Eles atuam como intermediários entre a solicitação do cliente e a lógica de negócio que será executada no serviço. Cada controller está associado a um ou mais endpoints da API e contém as regras específicas relacionadas ao processamento das requisições de um domínio.
- Services: Contêm a lógica de negócio e executam as operações específicas de cada domínio. O serviço gerencia a interação com o banco de dados e é responsável por coordenar as funcionalidades relacionadas ao domínio, como criar, atualizar ou excluir dados.
- DTOs: São os objetos de transferência de dados, que definem como os dados devem ser estruturados ao serem enviados ou recebidos nas requisições de API. Por exemplo, para criar um usuário, o DTO especifica como os dados de entrada devem ser formatados.
- Guards: Servem para adicionar uma camada extra de segurança ou validação nas requisições. Eles interceptam a rota antes que a lógica do controller seja executada e podem ser usados para verificar condições como autenticação ou autorização do usuário.
- Repositories: No módulo, a parte de repositório é composta por dois arquivos principais. O primeiro é uma interface de contrato**,** que define as operações que o serviço precisa executar para manipular os dados do domínio. Esta interface especifica os métodos que devem ser implementados para realizar as operações necessárias, como buscar, criar, atualizar ou excluir entidades. O segundo arquivo é o próprio repositório, que implementa essa interface e contém a lógica de interação com o banco de dados, executando as operações definidas na interface.

O repositório é responsável por gerenciar a persistência dos dados de um domínio específico.

- Module: Cada módulo tem uma classe Module, que funciona como o ponto de entrada e organização do módulo. No NestJS, essa classe define quais arquivos são controllers, serviços, repositórios, entre outros, e também importa ou exporta os componentes necessários para a operação do módulo.

Abaixo temos uma ilustração que representa o fluxo seguido pelas camadas de um modulo:

Figura 50: Estrutura de Pastas da API 3

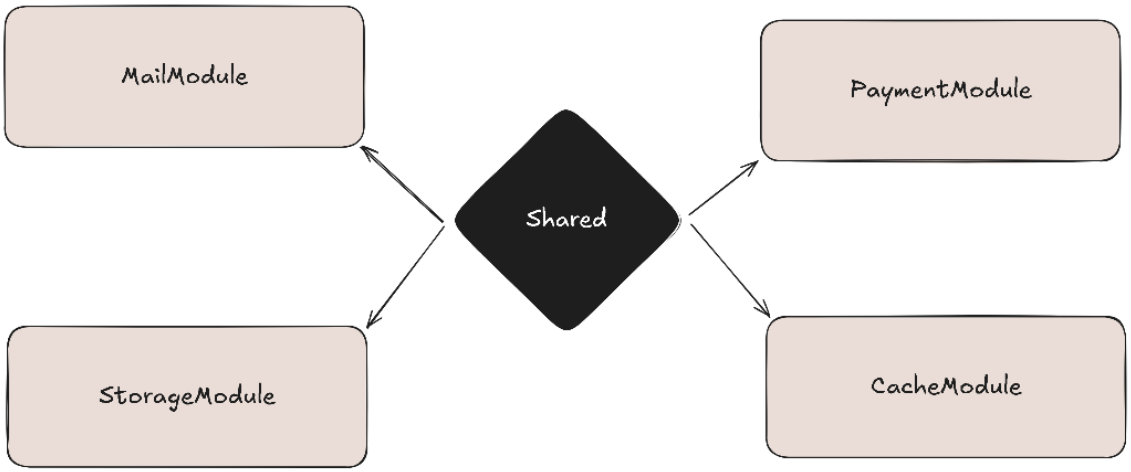


Fonte: Autoria Própria

Shared

Shared, ou "compartilhada", como a tradução do próprio nome sugere, é um conjunto de módulos de serviços que não possuem ligação direta com o sistema, diferentemente dos domínios relacionados ao banco de dados. Esses módulos funcionam como componentes compartilhados entre todos os outros módulos do sistema. Um exemplo típico seria o módulo de pagamentos, envio de e-mails e cache. Vale destacar que, embora esses módulos sigam uma estrutura similar à dos módulos na pasta `modules`, aqui eles não incluem repositórios ou controladores, isolando cada módulo com apenas um serviço específico.

Figura 51: Estrutura de Pastas da API 4



Fonte: Autoria Própria

4.4.2 Bibliotecas Utilizadas

Abaixo estão listadas as bibliotecas adicionais utilizadas para facilitar a criação da API, além das que já vêm integradas como padrão no NestJS. Cada biblioteca possui um breve resumo de sua finalidade:

@nestjs-modules/mailer: Biblioteca para envio de e-mails, oferecendo suporte para templates e configuração de transporte no ambiente do NestJS.

@nestjs/jwt: Suporte para autenticação baseada em JWT (JSON Web Tokens) no NestJS, permitindo a geração e validação de tokens para controle de acesso.

@prisma/client: Cliente oficial do Prisma que permite a interação eficiente e tipada com o banco de dados, simplificando operações de CRUD.

@supabase/supabase-js: Biblioteca JavaScript que facilita a integração com o Supabase, oferecendo acesso a banco de dados em tempo real, autenticação e armazenamento.

bcrypt: Biblioteca utilizada para criptografia e hashing seguro de senhas, permitindo a criação e verificação de senhas com segurança.

class-transformer: Ferramenta para transformar objetos em tipos de dados específicos e converter plain objects em classes, facilitando a manipulação de dados entre camadas.

class-validator: Biblioteca para validação de dados de entrada em classes TypeScript, garantindo que os dados atendam aos critérios definidos antes do processamento.

ioredis: Cliente avançado de Redis para Node.js, com suporte para operações de pub/sub, clusters e gerenciamento de dados em cache.

prisma: Ferramenta ORM que facilita o gerenciamento de bancos de dados de forma tipada em TypeScript, integrando-se ao Prisma Client.

stripe: Biblioteca oficial do Stripe para Node.js, utilizada para integrar funcionalidades de pagamento e gerenciar transações com segurança.

zod: Biblioteca de validação de esquemas em TypeScript, que permite a criação e verificação de tipos com segurança e de forma declarativa.

4.4.3 Integração com a camada de dados

Para integrar a camada de dados, foi utilizada a biblioteca PrismaClient, mencionada anteriormente. Como a camada de dados foi implementada com o Prisma, essa biblioteca oferece uma classe cliente que facilita o acesso e a manipulação dos dados. Com isso, foi criado um módulo específico na pasta *shared*, garantindo que todos os módulos de domínio tivessem acesso a essa funcionalidade.

4.4.4 Autenticação

A camada de apresentação de um software exige um controle de sessão para distinguir usuários. Para gerenciar a autenticação, foi adotado um módulo chamado Auth, que utiliza o JWT (JSON Web Token) para lidar com o processo de autenticação. O JWT é um formato compacto e auto suficiente que armazena informações, como o ID do usuário, em um token assinado, garantindo que qualquer alteração no token o torne inválido. O fluxo de autenticação funciona da seguinte maneira:

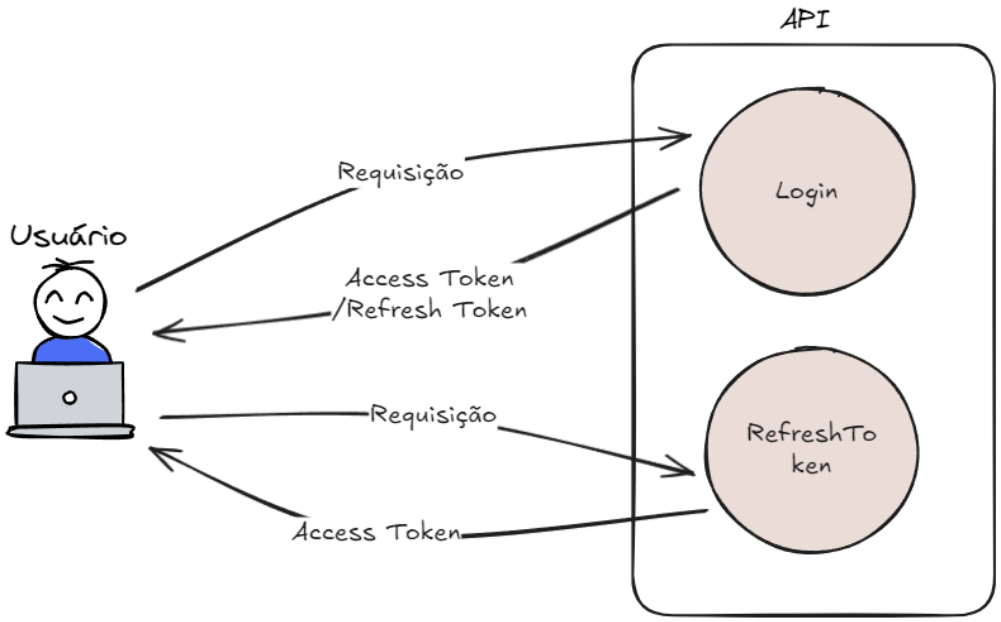
1. Login do Usuário no sistema**:**
   Quando um usuário faz login no sistema, ele envia suas credenciais (usuário e senha) para o servidor. O servidor valida essas informações e, se estiverem corretas, gera dois tokens:
   - Token de Acesso (Access Token): Este token é utilizado para autenticar o usuário em todas as requisições subsequentes. Ele contém o ID do usuário e outras informações relevantes. Esse token tem um tempo de vida limitado, geralmente de minutos a horas.

- Refresh Token: Este token é utilizado para obter um novo token de acesso quando o token de acesso expirar. O refresh token tem um tempo de vida mais longo, podendo ser de 1 mês.

2. Utilização do Token de Acesso: Sempre que o usuário faz uma requisição para acessar uma rota privada (ou seja, uma rota que requer autenticação), o token de acesso é enviado junto com a requisição, geralmente no cabeçalho Authorization. O servidor verifica a validade desse token, e, se estiver correto, permite o acesso ao recurso.
3. Atualização do Token de Acesso com o Refresh Token: Quando o token de acesso expira, o usuário pode usar o refresh token para obter um novo token de acesso, sem precisar realizar o login novamente. O refresh token é enviado para uma rota específica no servidor, que valida o refresh token e, se for válido, emite um novo token de acesso.

Para proteger as rotas privadas, foi criado um guard. No NestJS, o guard intercepta as requisições e valida a autenticidade do token de acesso. Ao ser aplicado nas rotas privadas, o guard garante que apenas usuários autenticados possam acessá-las, aumentando a segurança do sistema.

Figura 52: Fluxo dos Tokens

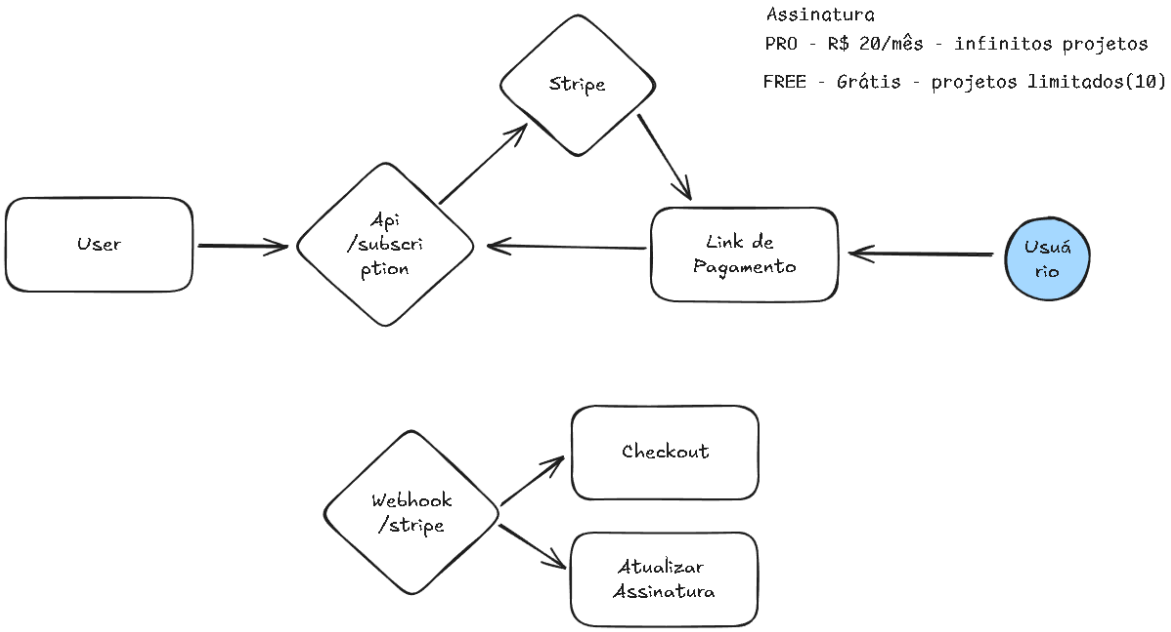


Fonte: Autoria Própria.

4.4.5 Integração com Pagamentos

O sistema integra um mecanismo de pagamento utilizando o Stripe, que oferece um ambiente seguro e pronto para integração. De acordo com a regra de negócios, um plano pago permite a criação de um número ilimitado de projetos, enquanto o plano gratuito limita a criação a apenas 10 projetos. Para suportar essa funcionalidade, foi implementado duas rotas específicas que atendem aos requisitos estabelecidos pelo Stripe.

Figura 53: Fluxo de Pagamentos



Fonte: Autoria Própria.

A imagem acima representa o fluxo lógico que é seguido. Na camada de apresentação, uma rota da API é responsável por gerar um link de pagamento, que é utilizado pelo usuário para concluir a transação. Além disso, temos uma segunda rota que o Stripe utiliza para executar processos internos, como a adição de usuários aos planos e a atualização automática das assinaturas.

4.4.6 Rotas da API

As rotas da API foram criadas para atender às necessidades da camada de apresentação. Abaixo, estão detalhadas as principais rotas implementadas para autenticação, usuários, projetos, cômodos e clientes.

Figura 54: Principais Rotas 1

Método	Rota	Descrição	Parâmetros/Corpo da Requisição	Resposta	Autenticado
POST	/auth/login	Realiza o login de um usuário	{ "email": "string", "password": "string" }	Token de autenticação e refresh token	Não
POST	/auth/forgot-password	Solicita um código para	{ "email": "string" }	Mensagem de sucesso e envio	Não

Fonte: Autoria Própria

Figura 55: Principais Rotas 2

Método	Rota	Descrição	Parâmetros/Corpo da Requisição	Resposta	Autenticado
		recuperação de senha		de e-mail com código de recuperação	
POST	/auth/verify-code	Verifica o código de recuperação de senha	{ "email": "string", "code": "string" }	Mensagem de validação	Não
POST	/auth/reset-password	Verifica o código de recuperação de senha	{ "email": "string", "newPassword": "string" }	Mensagem de sucesso	Não
POST	/auth/refresh-token	Renova o token de autenticação	{ "refreshToken": "string" }	Novo token de acesso	Não
POST	/users/register	Cria um novo usuário	{ "name": "string", "email": "string", "password": "string" }	Novo token de acesso	Não
PATCH	/users/upload-photo	Faz o upload de uma foto de perfil	Arquivo no formato `File` enviado no corpo da requisição	Objeto com ID e link da foto	Sim

Fonte: Autoria Própria

Figura 56: Principais Rotas 3

GET	/users	Obtém os dados do usuário autenticado		Objeto do usuário	Sim
PATCH	/users	Atualiza os dados do usuário	{ "name": "string", "email": "string" }	Objeto atualizado	Sim
PATCH	/users	Altera a senha do	{ "currentPas	ID do usuário atualizado	Sim

Fonte: Autoria Própria

Figura 57: Principais Rotas 4

POST	/projects	Adiciona um novo projeto	{ "name": "string", "description": "string", "cep": "string", ... }	ID do projeto criado	Sim
GET	/projects	Lista todos os projetos do usuário		Lista de projetos	Sim
GET	/projects/:id	Obtém detalhes de um projeto específico		Detalhes do projeto	Sim
DELETE	/projects/:id	Remove um projeto	`id` (na URL)	Mensagem de sucesso	Sim
POST	/convenients	Adiciona um novo cômodo ao projeto	{ "type": "string", "image360Uri": "string", "coverUrl": "string", "projectId": "string" }	ID do cômodo criado	Sim
GET	/convenients/:projectId	Lista os cômodos de um projeto específico		Lista de cômodos	Sim
DELETE	/convenients/:id	Remove um cômodo	`id` (na URL)	Lista de cômodos	Sim
POST	/clients	Adiciona um novo cliente	{ "name": "string", "phoneNumber":	Lista de cômodos	Sim

Fonte: Autoria Própria

Figura 58: Principais Rotas 5

Método	Rota	Descrição	Parâmetros/Corpo da Requisição	Resposta	Autenticado
			"string", "clientImage": "File" }		
GET	/clients	Lista todos os clientes do usuário		Lista de clientes	Sim
DELETE	/clients/:id	Remove um cliente	`id` (na URL)	Mensagem de sucesso	Sim
POST	/subscription/stripe	Escolhe um plano para o usuário	{ "userId": "string", "choice": "boolean" }	Mensagem de sucesso	Sim
POST	/subscription/stripe/webhook	Recebe eventos do Stripe para processar mudanças no pagamento (ex: sucesso, falha, cancelamento).	Evento do Stripe	Mensagem de sucesso	Sim

Fonte: Autoria Própria

4.4.7 Deploy

O processo de deploy consiste em disponibilizar a API na nuvem, com o objetivo de torná-la acessível em qualquer rede por meio de um domínio único. No entanto, antes de realizar esse deploy, é necessário configurar uma série de serviços que a aplicação precisa para funcionar corretamente, como banco de dados e armazenamento de arquivos. Inicialmente, o banco de dados estava em um ambiente de desenvolvimento local, ou seja, era algo presente apenas na máquina de desenvolvimento, o que o tornava limitado e não escalável.

Por isso, foi necessário buscar uma plataforma que permitisse escalar e publicar o banco de dados na nuvem. Além disso, a aplicação precisava de uma solução para armazenar arquivos, como fotos de usuários, já que não é recomendável armazenar arquivos grandes e complexos diretamente no banco de dados. Assim, optou-se por utilizar um serviço de armazenamento online.

Esses serviços requerem o uso de chaves secretas para garantir a comunicação segura entre a aplicação e os sistemas externos. Para proteger essas chaves, foi criado um arquivo `env` na raiz do projeto, que isola de maneira segura todas as credenciais e evita que sejam expostas publicamente.

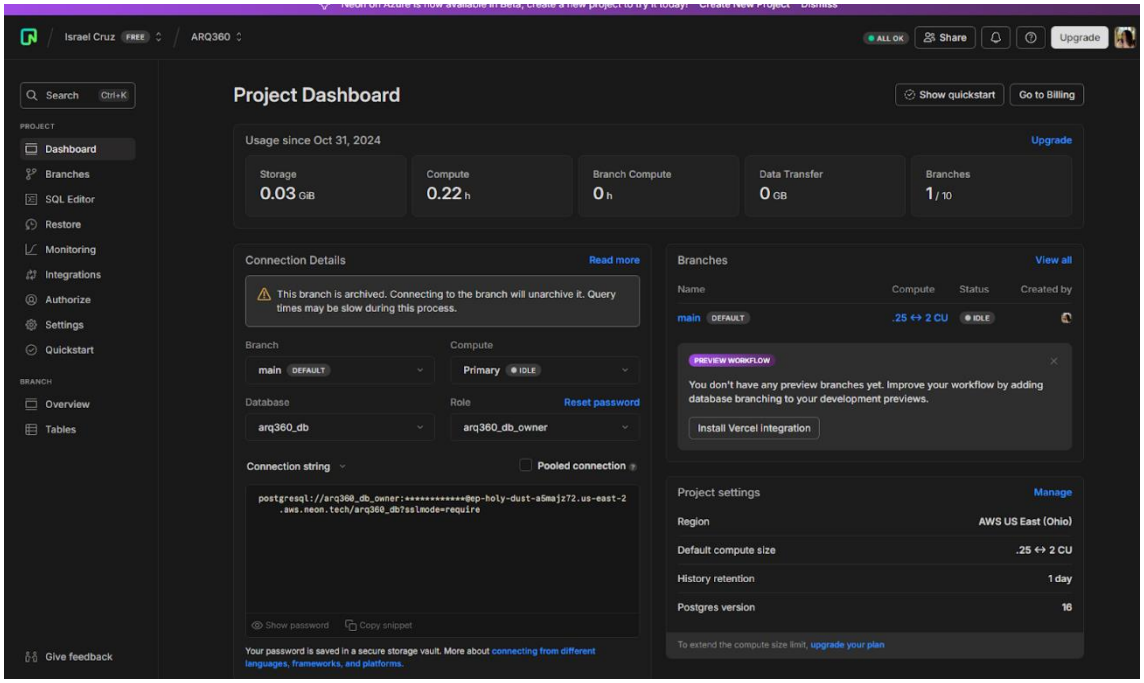
Para simplificar a integração, também foi incluído um arquivo de exemplo `.env`, orientando sobre quais valores devem ser configurados para cada serviço. Cada serviço também possui um módulo específico na pasta `shared`, contendo as funcionalidades necessárias para integração e uso geral.

Abaixo, foram detalhadas as plataformas escolhidas para atender a cada uma dessas necessidades:

- Banco de Dados: Para o banco de dados, foi escolhida a plataforma Neon, que oferece suporte ao PostgreSQL de forma gratuita, exatamente o banco utilizado pelo sistema. A migração dos dados para o Neon tornou possível ter um banco de dados escalável e hospedado na nuvem, garantindo a alta disponibilidade e desempenho necessário para a aplicação.

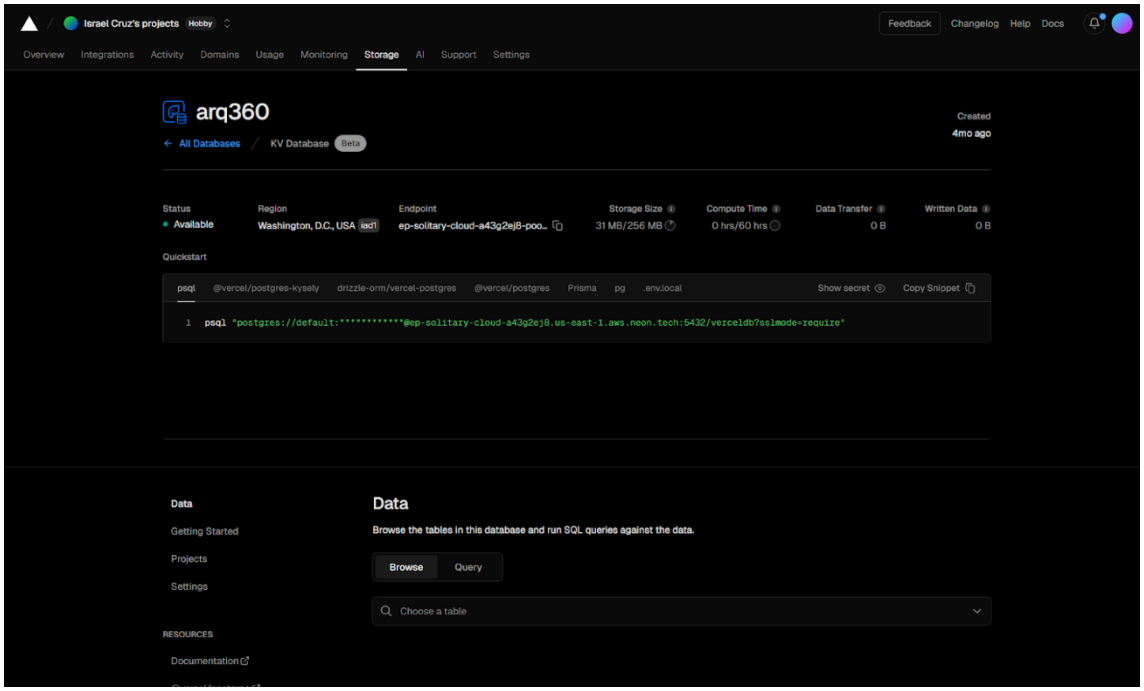
- Redis: Para o cache e gerenciamento de sessões, utilizamos a plataforma Vercel, que oferece integração com o Redis KV de forma gratuita. Esta solução simplifica a publicação e o gerenciamento do banco de dados Redis, permitindo que ele seja acessível de forma rápida e eficiente.

Figura 59: Neon



Fonte: https://console.neon.tech/app/projects.

Figura 60: Fluxo de Pagamentos Deploy

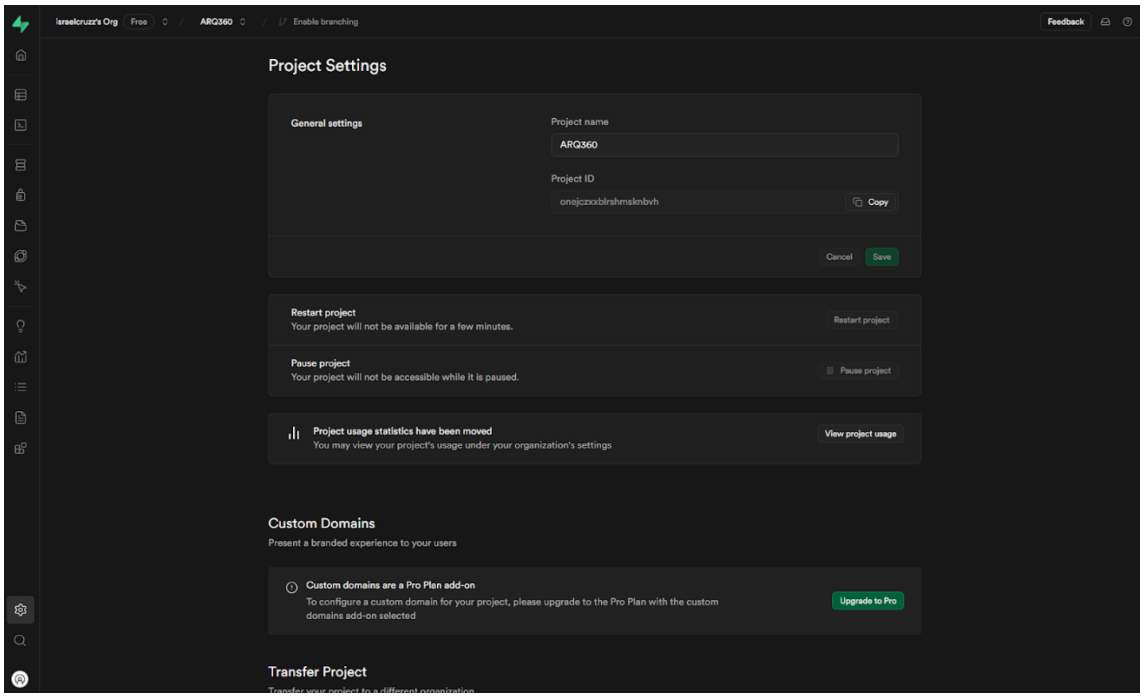


Fonte: https://vercel.com.

- Armazenamento de Arquivos: A escolha para o armazenamento de arquivos foi a plataforma Supabase, que não só oferece uma série de recursos de armazenamento, como também permite a criação de buckets

— contêineres na nuvem onde arquivos, como imagens e documentos, podem ser armazenados de forma organizada e escalável.

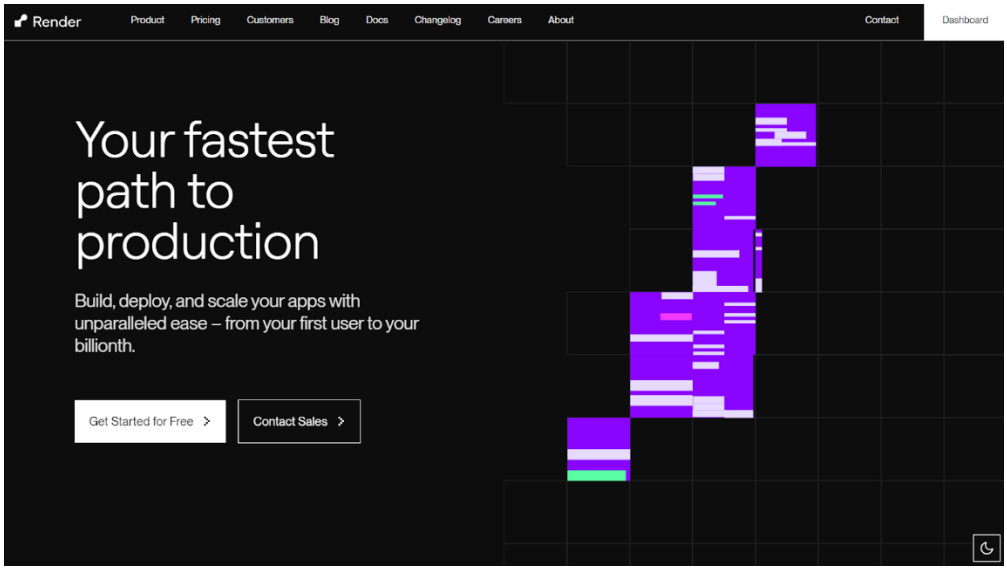
Figura 61: Supabase



Fonte: https://supabase.com/dashboard/project.

- Hospedagem da API: Para o deploy da API, a plataforma escolhida foi a Render, que oferece suporte gratuito e um alto limite de requisições por minuto. Isso possibilitou a criação de um domínio único para a API, tornando-a acessível na camada de apresentação.

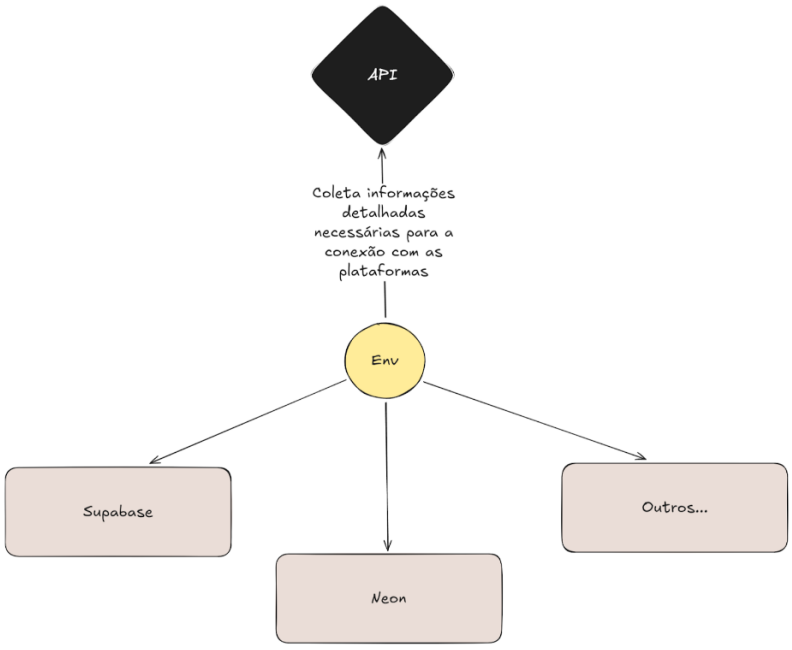
Figura 62: Fluxo de Pagamentos Render



Fonte: https://render.com/.

Por fim, o fluxo de integração entre esses serviços foi cuidadosamente desenhado para garantir que todos os componentes funcionem de maneira harmoniosa e escalável, como ilustrado abaixo.

Figura 63: Fluxo da Conexão da API com Plataformas Externas



Fonte: Autoria Própria.

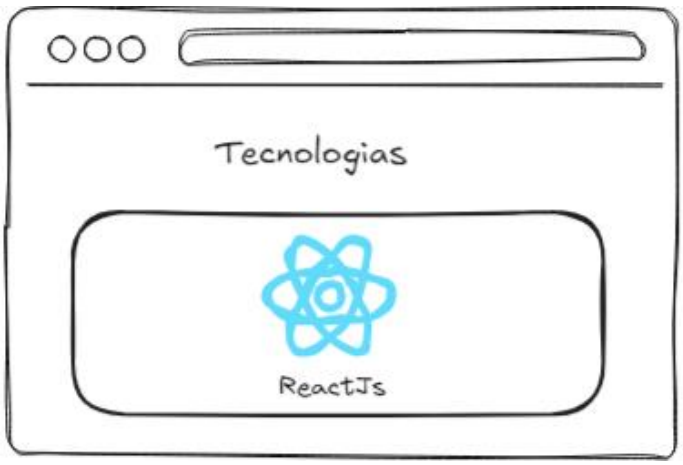
4.5 Camada de Apresentação

Ao contrário das outras camadas, nesta etapa serão desenvolvidos dois softwares distintos. O primeiro será um site, que servirá tanto como uma plataforma de divulgação quanto como uma forma de os usuários encontrarem e instalarem o aplicativo. Além disso, o site incluirá uma página específica que será carregada pelo aplicativo por meio de uma webview. O segundo software será o próprio aplicativo, que fornecerá a funcionalidade central do sistema. Para explicar melhor os detalhes de cada um desses softwares, serão apresentados tópicos específicos para cada um deles.

4.5.1 Criação de Website para divulgação do aplicativo

Para o desenvolvimento do website, optou-se por utilizar a tecnologia React.js, que facilitará o processo de criação. Seguindo o protótipo definido na camada de design, para garantir que o visual do website esteja alinhado com as especificações previamente estabelecidas.

Figura 64: Criação de Website para divulgação do aplicativo



Fonte: Autoria Própria.

4.5.1.2 Bibliotecas Utilizadas

Nesta seção, são apresentadas as bibliotecas adicionais utilizadas para acelerar o desenvolvimento e simplificar a implementação de funcionalidades, complementando as bibliotecas já fornecidas por padrão pelo React.js:

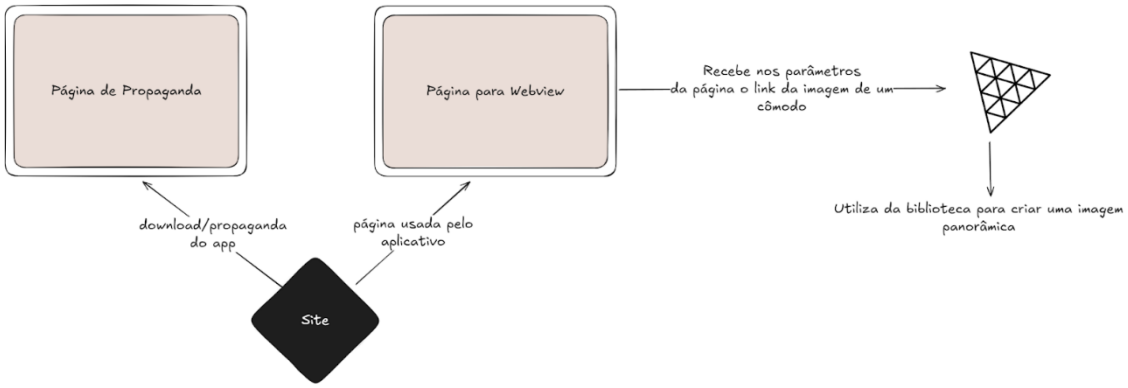
- React Three.js Fiber: Utilizada para facilitar a manipulação e renderização de elementos 3D no contexto do React, integrando a funcionalidade do Three.js de forma eficiente.

- Axios: Ferramenta para realizar requisições HTTP de forma simplificada e estruturada, permitindo comunicação com o backend e consumo de APIs.

4.5.1.3 Criação e Especificação do Website

O site é composto por duas páginas principais. A primeira é dedicada à promoção do aplicativo, apresentando informações gerais e disponibilizando o download do aplicativo. A segunda página é projetada para ser utilizada pelo aplicativo em uma webview, permitindo a visualização de um cômodo. Essa página foi desenvolvida com a biblioteca Three.js, mencionada anteriormente, e funciona de forma resumida ao receber, como parâmetro, o link da imagem do cômodo, criando uma visão panorâmica interativa da mesma.

Figura 65: Especificação da Lógica do Site



Fonte: Autoria Própria.

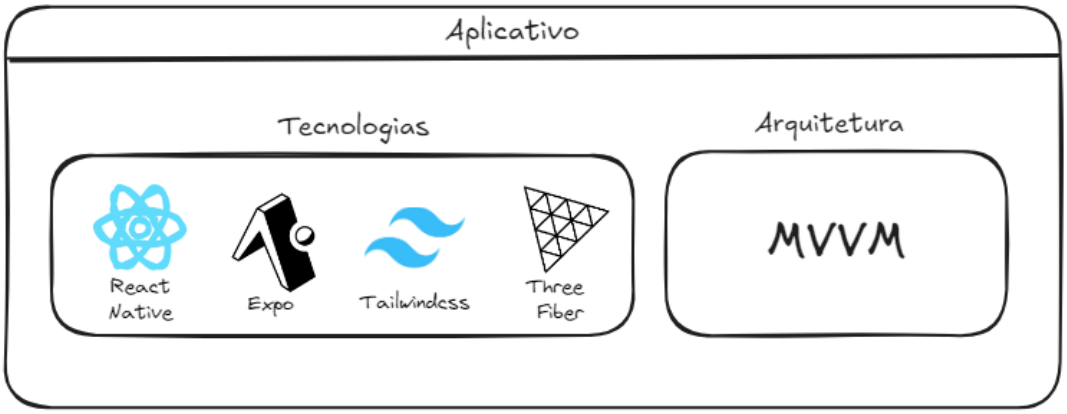
4.5.1.4 Deploy

O deploy foi realizado utilizando a plataforma Vercel, que oferece suporte tanto para o deploy da camada de sistema quanto para sites. Além de sua praticidade, a Vercel disponibiliza planos gratuitos, tornando-a uma solução acessível e eficiente para hospedar o projeto.

4.5.2 Desenvolvimento do Aplicativo

Para o desenvolvimento do aplicativo, optou-se por utilizar o React Native com Expo, uma ferramenta que oferece excelente suporte para desenvolvimento mobile. A linguagem escolhida foi TypeScript, que proporciona uma tipagem estática robusta e facilita a manutenção do código. Para a estilização, utilizamos o Tailwind CSS, conhecido por sua abordagem utilitária e eficiente. Além disso, para criar visualizações em 360 graus, empregamos o React Three Fiber, que integra o poder do Three.js com a simplicidade do React.

Figura 66: Detalhes Aplicativo



Fonte: Autoria Própria.

4.5.2.1 Arquitetura

O projeto adota uma arquitetura parcial, aplicando conceitos específicos apenas onde são realmente necessários. De forma geral, a estrutura do projeto não segue uma arquitetura fixa, mas há diretórios específicos onde foi implementada a arquitetura MVVM (Model-View-ViewModel), um padrão de design que promove a separação de responsabilidades dentro da aplicação.

No MVVM:

- Model: Representa a lógica de negócios e os dados do aplicativo, sendo responsável por gerenciar a recuperação e manipulação dos dados de forma independente da interface do usuário.

- View: Cuida da apresentação dos dados e da interação com o usuário. Ela exibe as informações fornecidas pelo Model e coleta as ações realizadas pelo usuário.

- ViewModel: Funciona como um intermediário entre o Model e a View. Ele formata e organiza os dados para a View e reage às interações do usuário, atualizando o Model quando necessário.

Essa arquitetura facilita a manutenção e escalabilidade do projeto, isolando a lógica de apresentação da lógica de negócios, o que resulta em um código mais limpo e modular.

Nas partes do projeto onde a arquitetura MVVM não foi aplicada, optamos por uma separação em camadas para organizar as responsabilidades de forma funcional. Essa abordagem será detalhada na próxima seção.

4.5.2.1.1 Estrutura das Pastas

A organização das pastas foi planejada para separar responsabilidades entre os diretórios, aplicando arquiteturas e padrões somente onde necessário. Abaixo, cada diretório é descrito em detalhes:

- Assets: Armazena todos os arquivos estáticos utilizados no aplicativo, como imagens, vídeos ou GIFs.

- Components: Os componentes agrupam blocos de código reutilizáveis, permitindo que sejam utilizados em diferentes partes do aplicativo sem necessidade de replicação. Isso facilita a manutenção e promove a reutilização do código. Este diretório organiza todos os componentes do software.

- Service: Contém toda a lógica de comunicação com a camada de sistema, incluindo funções e variáveis que são usadas para interagir com a API ou outras fontes de dados externas.

- Context: Este diretório armazena os contextos da aplicação, um conceito oriundo do React. Contextos funcionam como provedores de dados que podem ser acessados por qualquer parte do aplicativo. Exemplos incluem contextos para autenticação, temas e onboarding.

- Hooks: Guarda os hooks personalizados criados para o projeto. Hooks são funções do React que permitem utilizar estado ou outras funcionalidades em componentes funcionais. Neste diretório, organizamos hooks que encapsulam lógicas reutilizáveis específicas.

- Mocks: Armazena arquivos temporários usados para simular dados ou respostas de sistemas externos, como a simulação de respostas de uma API para testar funcionalidades com dados fictícios.

- Screens: Organiza todas as telas do aplicativo, divididas em três subdiretórios principais:

- Private: Telas acessíveis apenas para usuários logados.

- Public: Telas de acesso público, como login, cadastro ou recuperação de senha.

- Onboarding: Telas que apresentam o funcionamento do aplicativo antes do usuário entrar no fluxo principal.

Nesta seção, foi aplicada a arquitetura MVVM, separando responsabilidades de forma clara.

- Navigation: Centraliza o gerenciamento de navegação do aplicativo. Contém arquivos para organizar os fluxos de navegação de diferentes áreas, como telas privadas, públicas e de onboarding.

- Tokens: Este diretório define padrões de estilização da aplicação, como um arquivo que organiza todas as cores utilizadas. Ele ajuda a manter consistência visual no design do aplicativo.

- Utils: Agrupa funções auxiliares reutilizáveis em todo o aplicativo. Exemplos incluem utilitários para tratar erros vindos da camada de sistema ou outras funções genéricas.

4.5.2.2 Bibliotecas utilizadas

Abaixo estão listadas as bibliotecas adicionais utilizadas para facilitar a criação do aplicativo, além das que já vêm integradas como padrão no ambiente do React Native e Expo. Cada biblioteca possui um breve resumo de sua finalidade:

- @gorhom/bottom-sheet: Biblioteca para criar bottom sheets (folhas que deslizam de baixo para cima), com alta personalização e suporte a gestos.
- @hookform/resolvers: Integração com bibliotecas de validação (como Yup e Zod) para facilitar o uso de validações com React Hook Form.
- @react-native-async-storage/async-storage: Biblioteca de armazenamento local para dados persistentes, permitindo salvar dados entre sessões no dispositivo.
- @react-native-community/netinfo: Fornece informações sobre o status de conexão de rede do dispositivo, como tipo de rede e status online/offline.
- @react-native-picker/picker: Componente de seleção para criar listas suspensas em dispositivos móveis, com suporte nativo em iOS e Android.
- @react-navigation/bottom-tabs: Componente de navegação para criar abas na parte inferior do aplicativo, popular em aplicativos móveis para facilitar a navegação.
- @react-navigation/native: Biblioteca base para navegação em aplicativos React Native, oferecendo navegação de pilha, abas e outras funcionalidades.
- @react-navigation/stack: Proporciona navegação baseada em pilha, com transições de tela que simulam uma pilha de navegação nativa.
- axios: Cliente HTTP usado para fazer requisições de rede, popular por sua simplicidade e suporte a interceptores de requisição e resposta.
- deprecated-react-native-prop-types: Pacote que fornece definições de prop-types para componentes React Native, substituindo os prop-types depreciados do React Native.
- expo-image-picker: Biblioteca para acessar a câmera e galeria de imagens do dispositivo, permitindo selecionar ou capturar fotos e vídeos.
- expo-linear-gradient: Biblioteca para criar gradientes lineares em componentes, útil para adicionar efeitos visuais atraentes.
- expo-linking: Facilita o manuseio de links externos e URLs dentro do aplicativo, permitindo o deep linking.
- expo-status-bar: Componente de barra de status do Expo que permite o controle do estilo e da aparência da barra de status.

- lucide-react-native: Biblioteca de ícones para React Native, com uma coleção de ícones modernos e customizáveis.
- nativewind: Biblioteca que integra Tailwind CSS com React Native, permitindo estilos utilitários no desenvolvimento mobile.
- patch-package: Ferramenta que permite corrigir diretamente as dependências do Node_modules, facilitando ajustes em bibliotecas de terceiros.
- postinstall-postinstall: Executa comandos automaticamente após o postinstall, geralmente usado para configurar dependências após a instalação.
- react-native-gesture-handler: Biblioteca para manipulação de gestos no React Native, com suporte a gestos complexos como arrastar e deslizar.
- react-native-maps: Componente de mapas para React Native, com suporte a renderização de mapas nativos do Google Maps e MapKit.
- react-native-masked-text: Biblioteca para formatação de texto com máscara, usada para entrada de dados como CPF, telefone e valores monetários.
- react-native-picker-select: Componente de seleção que oferece uma interface de seleção customizável, usado para criar listas suspensas.
- react-native-reanimated: Biblioteca para animações altamente performáticas e nativas no React Native, com suporte a animações complexas.
- react-native-safe-area-context: Fornece informações sobre as áreas seguras da tela em dispositivos móveis, ajustando automaticamente a interface para evitar sobreposição de barras e entalhes.
- react-native-screens: Biblioteca para otimização de desempenho de navegação, tornando cada tela nativa e melhorando a experiência de navegação.
- react-native-snap-carousel: Componente de carrossel para React Native, com suporte a rolagem de elementos de forma interativa e visualmente atraente.
- @types/react-native-snap-carousel: Tipos TypeScript para o react-native-snap-carousel, garantindo tipagem robusta ao utilizar essa biblioteca.
- react-native-svg: Permite a utilização de SVGs como componentes React Native, facilitando a inclusão de gráficos e ícones vetoriais.
- react-native-toast-message: Biblioteca para exibir mensagens de notificação e alertas como toasts, com customização de estilos e duração.

- react-native-webview: Permite incorporar conteúdo da web em um aplicativo React Native, como exibir páginas da web dentro do app.
- yup: Biblioteca de validação de esquemas que permite definir e verificar esquemas de dados, garantindo a conformidade dos dados de entrada no app.
- expo-location: Biblioteca para acessar serviços de localização do dispositivo, permitindo coletar coordenadas GPS e trabalhar com geolocalização.

4.5.2.3 Componentes

Como mencionado anteriormente, os componentes representam os elementos fundamentais que serão utilizados no sistema. A seguir, exploraremos esses componentes em detalhes, apresentando uma tabela explicativa que descreve cada um deles e suas respectivas funções no sistema.

Tabela 15: Componentes

Nome do Componente	Função	Descrição	Imagem
Back Button	Voltar para a tela anterior	Este componente é um botão que, ao ser clicado pelo usuário, retorna para a tela anterior.	
Badge	Exibir informações relevantes e oferecer funcionalidades específicas alinhadas ao contexto em que é utilizado	Este componente é utilizado para exibir informações e executar diferentes funções conforme o contexto em que é aplicado. Por exemplo, no caso de gerenciamento de clientes, ele pode ser usado tanto para remover um cliente quanto para adicionar um novo.	
Button	Um botão projetado para executar uma função específica com base no contexto em que está inserido	Este componente é um botão genérico, projetado para ser utilizado em diversos contextos, executando uma função específica ao ser clicado.	
Card Project	Exibir os detalhes de um projeto e redirecionar para a tela correspondente ao projeto representado	Este componente é utilizado para exibir os detalhes de um projeto e, ao ser clicado, redireciona o usuário para a tela com informações completas do projeto.	

Fonte: Autoria Própria

Tabela 16: Componentes 2

Nome do Componente	Função	Descrição	Imagem
Carousel	Exibe os projetos de um usuário de forma dinâmica.	O carrossel exibe os projetos do usuário de forma dinâmica, permitindo navegar entre eles com facilidade, exibindo informações resumidas de cada projeto.	
Client Line	Exibe as informações de um cliente com a opção de removê-lo.	Este componente é responsável por exibir as informações de um cliente, como nome e imagem, e, quando necessário, inclui um badge para permitir sua remoção.	
Convenient Line	Lista informações do cômodo e permite visualizar em 360 graus através de um botão.	Este componente é responsável por listar os dados de um cômodo, como nome e imagem, e oferece a opção de navegar para a tela de visualização em 360 graus ao clicar em um botão.	
Description Text	Padronização de texto para descrições	Este componente é responsável por padronizar a exibição de textos, como descrições, aplicando uma estilização consistente.	
Heading Text	Padronização de texto para títulos	Este componente é responsável por padronizar a exibição de textos, como títulos, aplicando uma estilização consistente.	

Fonte: Autoria Própria

Tabela 17: Componentes 3

Nome do Componente	Função	Descrição	Imagem
Input	Entrada de dados	Este componente tem como objetivo padronizar todos os inputs, que são os meios de entrada de dados utilizados pelos usuários, garantindo consistência e uniformidade na interface.	
Label	Descrever o propósito de um input	Este componente tem como objetivo descrever claramente o propósito de um input, como, por exemplo, quando a entrada de dados é destinada ao preenchimento de um e-mail.	
Text Link	Texto que direciona para outra tela	Este componente exibe um texto que, ao ser clicado, redireciona o usuário para uma tela específica.	

Fonte: Autoria Própria

4.5.2.6.1 Estilização

A estilização do aplicativo foi realizada utilizando a biblioteca StyleSheet, nativa do React Native. Para garantir a padronização e facilitar a manutenção, foi criado um diretório styles dentro da pasta token, que contém os principais parâmetros de estilização utilizados ao longo do projeto. Abaixo, uma tabela exemplifica alguns dos estilos definidos nesta estrutura. stilização

Figura 67: Estilização

Tipo	Propriedade	Valor
Cor	Primary	#6200ee
Cor	Secondary	#03dac6
Cor	Background	#f5f5f5
Cor	Text Primary	#000000
Tipografia	Font Family	Roboto-Bold
Tamanhos	Small	24px

Fonte: Autoria Própria.

4.5.2.6.1 Integração com a camada de sistema

Como mencionado anteriormente, o aplicativo possui um diretório chamado service, onde é utilizada a biblioteca Axios, previamente citada. Esse diretório contém um arquivo principal, ApiServiceConfig, que usa uma funcionalidade do Axios para criar um objeto configurado com a URL base da API da camada de sistema, evitando a necessidade de constantes alterações manuais. Além disso, interceptadores são configurados para processar as respostas da API e tratar possíveis erros, como o caso de um erro de autorização.

Há também o arquivo ApiService, que implementa uma classe com todas as funcionalidades necessárias, como a obtenção de todos os projetos de um usuário. Dessa forma, todo o sistema acessa a camada de sistema de forma consistente, minimizando bugs e garantindo uniformidade nos padrões.

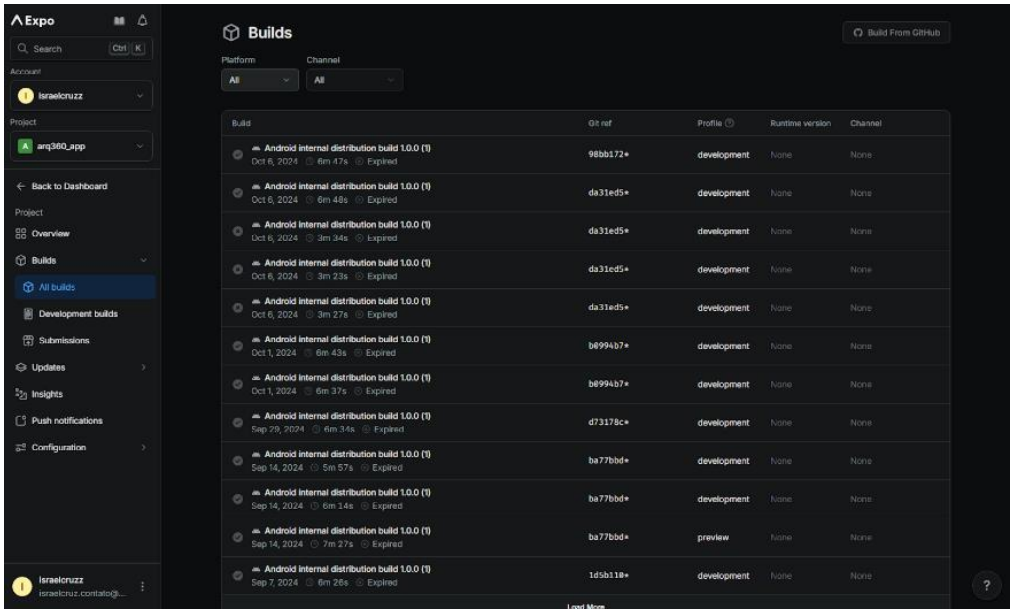
4.5.2.8 Deploy

O processo de deploy do aplicativo será realizado utilizando o EAS (Expo Application Services), uma ferramenta do Expo que facilita a distribuição do aplicativo sem a necessidade de publicá-lo nas lojas oficiais de aplicativos. O EAS permite gerar builds nativos do aplicativo tanto para Android quanto para iOS.

O processo de deploy envolve a geração dos builds nativos do aplicativo, que são os arquivos executáveis (APK para Android e IPA para iOS). Esses

arquivos serão disponibilizados diretamente no site criado para o aplicativo, permitindo que os usuários façam o download e instalem o aplicativo em seus dispositivos sem precisar passar pelas lojas de aplicativos tradicionais. Assim, o aplicativo pode ser instalado manualmente, seja em dispositivos Android ou iOS, garantindo uma distribuição direta e sem a necessidade de aprovação ou revisão por parte das lojas oficiais.

Figura 68: EAS



Fonte: https://expo.dev/accounts/israelcruzz/projects/arq360_app/builds.

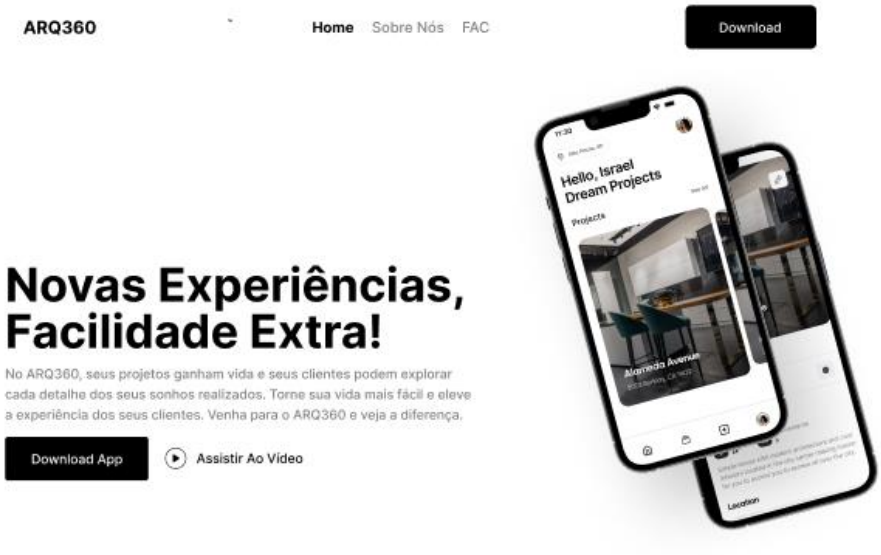
5. APRESENTAÇÃO DO PROJETO

A seguir, serão apresentados os resultados após a finalização de todas as camadas de desenvolvimento, resultando na entrega dos dois softwares previstos para o projeto. Serão detalhados os dois softwares, destacando suas funcionalidades.

5.1 Apresentação do Website

A seguir serão detalhadas todas as telas presentes no site: ARQ360, que tem foco

Figura 69: Tela 1 Site



Fonte: Autoria Própria

A tela principal conta com uma breve introdução do que se trata o ARQ360 e já traz as opções de download explicitas ao usuário para que ele consiga visualizar e já obter seu aplicativo.

Figura 70: Tela 2 Site

Fonte: Autoria Própria

Figura 71: Tela 3 Site

Fonte: Autoria Própria

As telas acima apresentam os benefícios e qualidades do ARQ360 para o usuário saber quais os pontos positivos e conhecer melhor a plataforma com imagens de seu funcionamento.

As telas abaixo tem como objetivo mostrar ao usuário críticas positivas já feitas ao produto, para conquistar a confiança do usuário de que tanto o site quanto o aplicativo são seguros.

Figura 72: Tela 4 Site

Fonte: Autoria Própria

Figura 73: Tela 5 Site

Fonte: Autoria Própria

Figura 74: Tela 6 Site

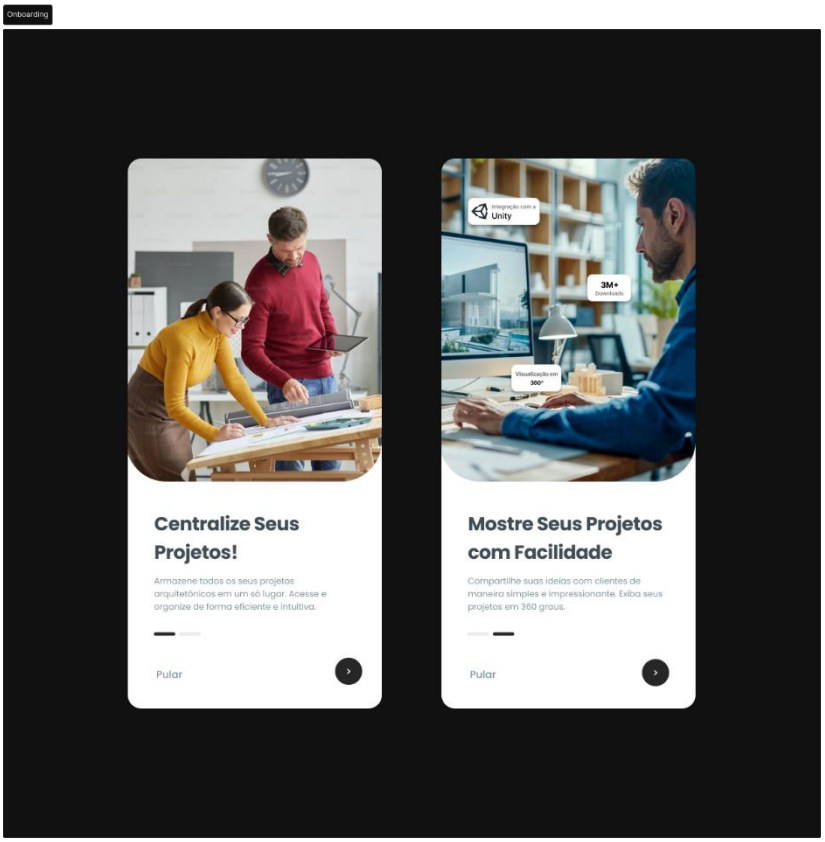
Fonte: Autoria Própria

A tela acima está localizada no final do site e retoma o botão de download para relembrar o usuário e no rodapé temos as fontes de contato para caso de dúvidas não solucionadas.

5.2 Apresentação do Aplicativo

A seguir serão detalhadas todas as telas presentes no aplicativo: ARQ360

Figura 75: Telas Iniciais

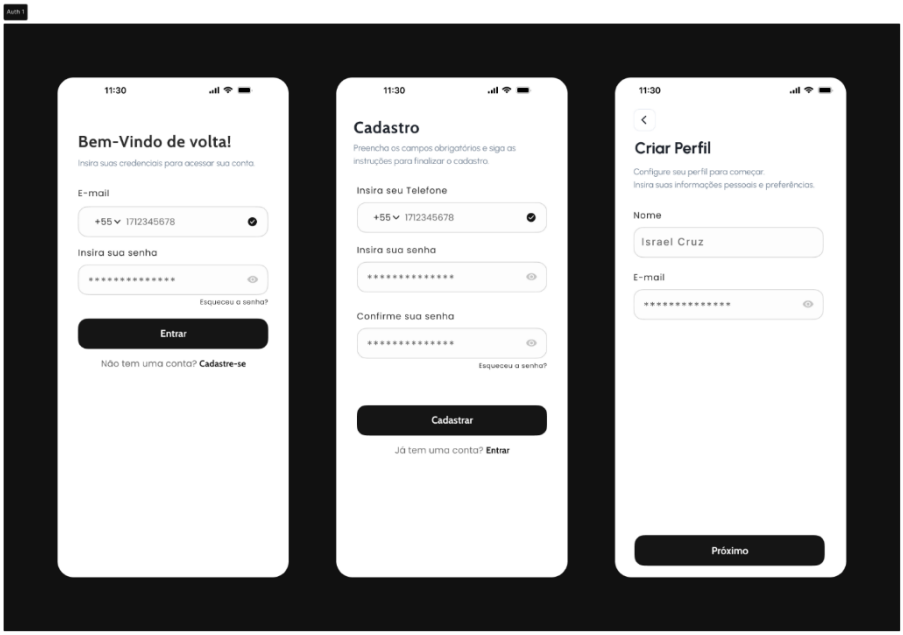


Fonte: Autoria Própria

As telas iniciais são a base para o usuário se orientar e receber dicas do que está por vir no aplicativo.

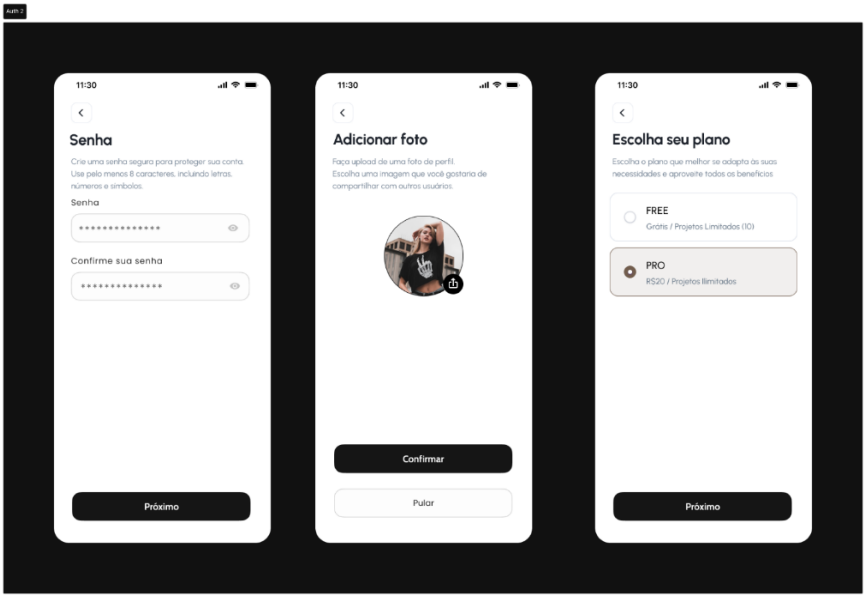
As telas de login foram desenvolvidas para permitir que o usuário acesse sua conta a qualquer momento, sem perder seus projetos e progressos. A interface tem om intuito de fazer o cadastro e login o mais rápido e fácil possível para não gerar estresse ao usuário.

Figura 76: Login e Cadastro



Fonte: Autoria Própria

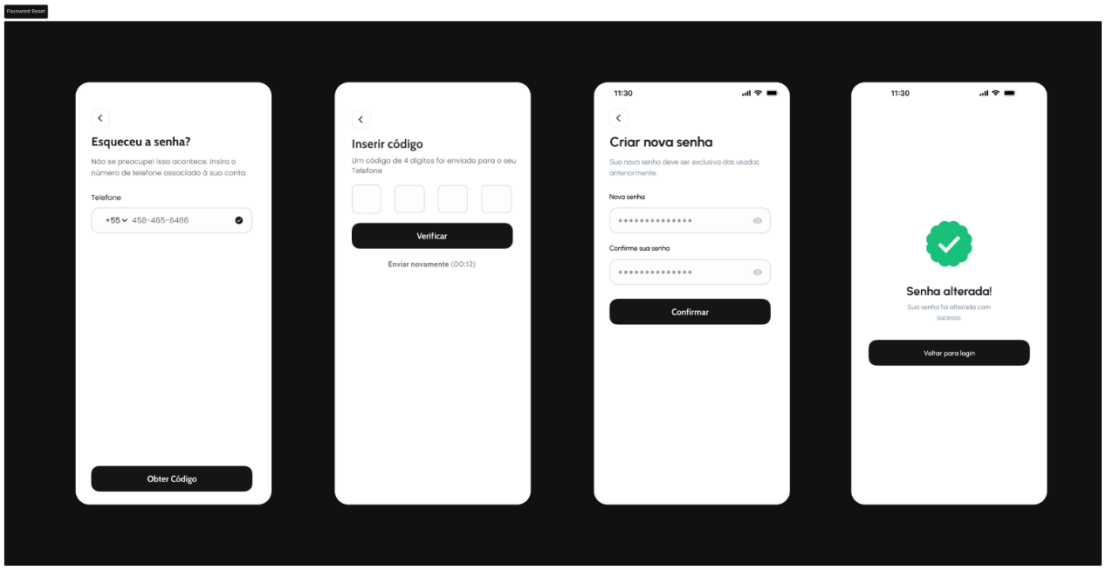
Figura 77: Login e Cadastro 2



Fonte: Autoria Própria

As telas de recuperação de senha são fundamentais para garantir que o usuário possa recuperar o acesso à sua conta em caso de esquecimento. A interface deve enfatizar claramente as informações que o usuário precisa fornecer, além de oferecer confirmações visuais do sucesso nas alterações. Nesse contexto, a sugestão de dados a serem inseridos, acompanhada por um selo verde com um ícone de 'check', ajuda o usuário a compreender as instruções e confirma que as etapas foram concluídas com sucesso.

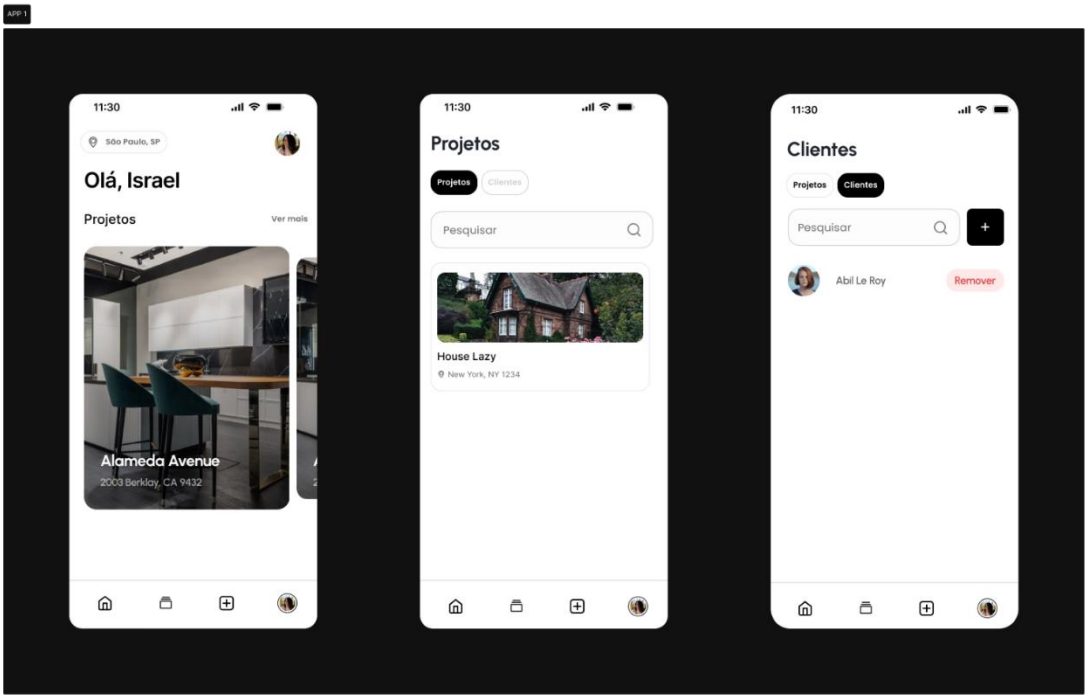
Figura 78: Resetar Senha



Fonte: Autoria Própria

As telas do menu principal e outras essenciais do aplicativo foram projetadas para fornecer as informações necessárias de forma clara e objetiva, utilizando um design minimalista que evita a sobrecarga cognitiva. O menu principal é composto pelas seguintes abas: Home, Projetos, Criar Projeto e Perfil.

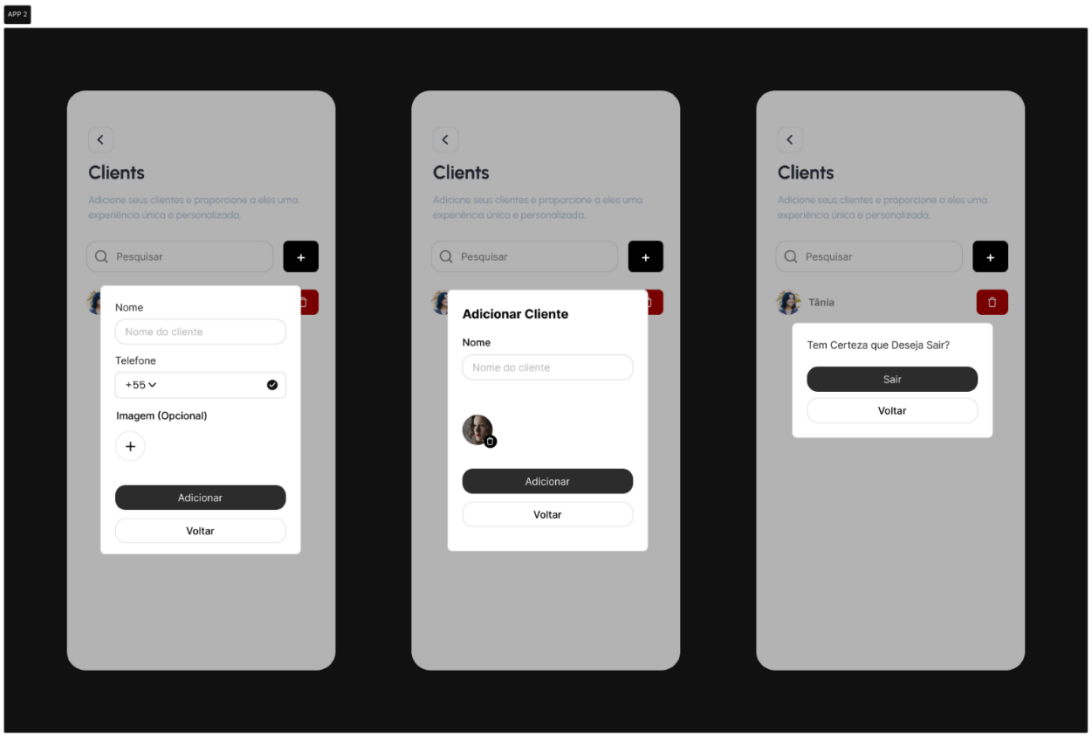
Figura 79: Home/Projetos/Clientes



Fonte: Autoria Própria

Nas seguintes telas, é mostrado que é possível adicionar seus clientes à plataforma para compartilhar seus projetos com eles. As telas oferecem opções para adicionar informações sobre cada cliente, além de permitir sua remoção quando necessário

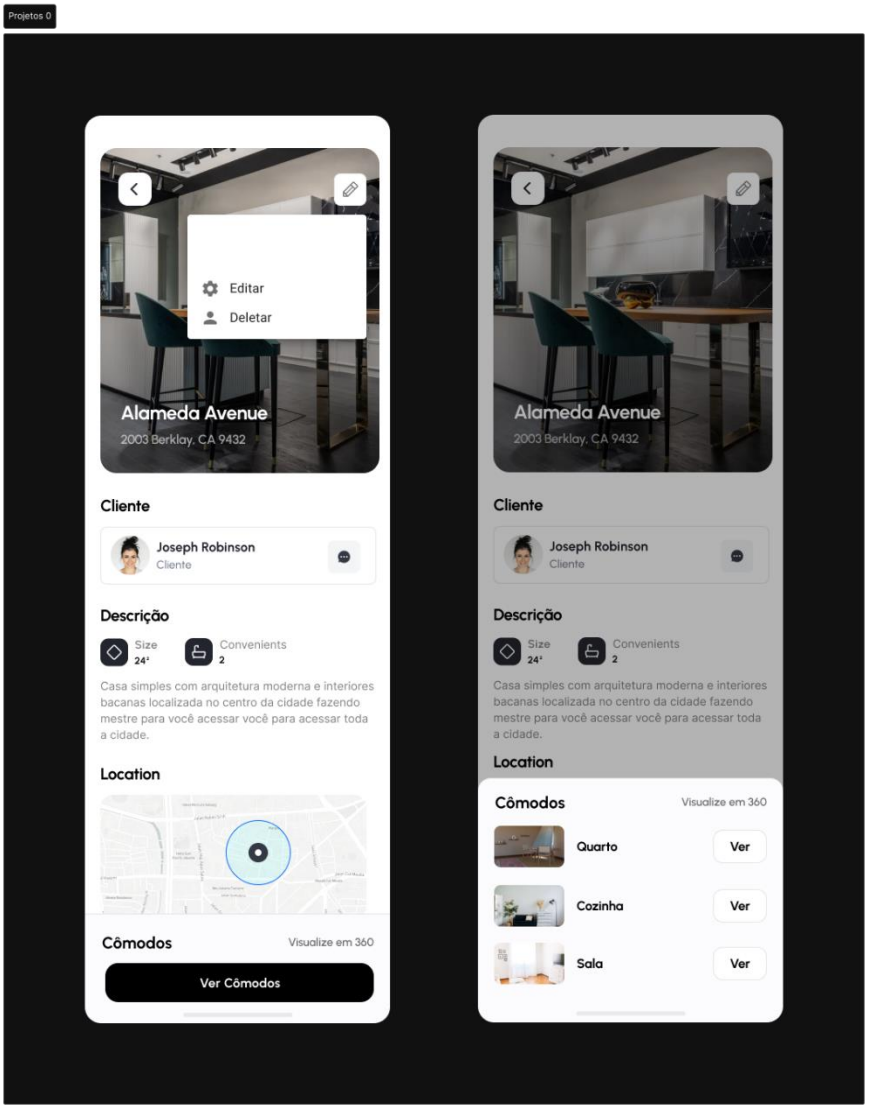
Figura 80: Adicionar Clientes



Fonte: Autoria Própria

As telas a seguir ilustram como o usuário pode visualizar seus projetos, apresentando todas as informações essenciais, incluindo uma descrição detalhada do imóvel, localização e a divisão por cômodos.

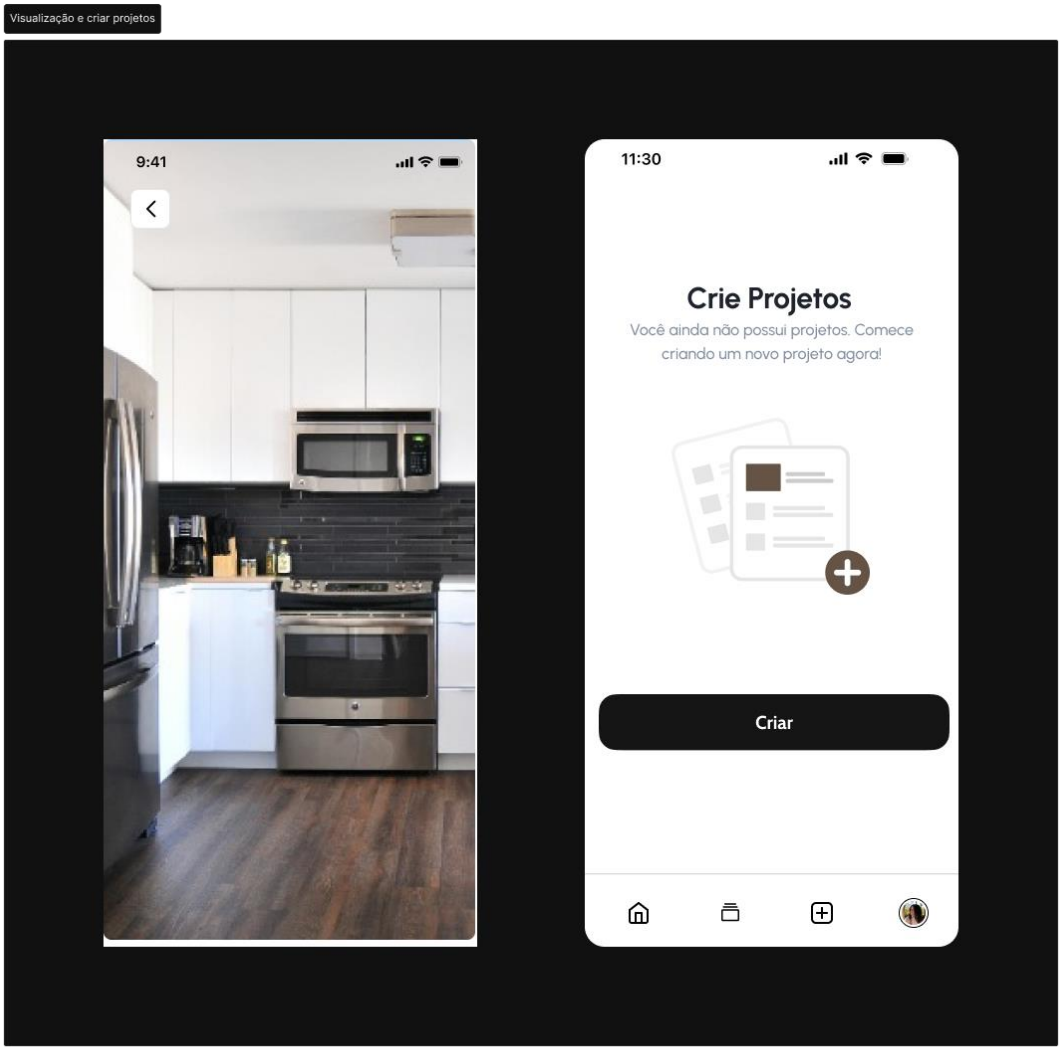
Figura 81: Projetos



Fonte: Autoria Própria

Nas próximas telas, é possível observar a implementação da visualização em 360 graus nos projetos, assim como a aparência da aba 'Projetos' quando o usuário acessa o aplicativo pela primeira vez.

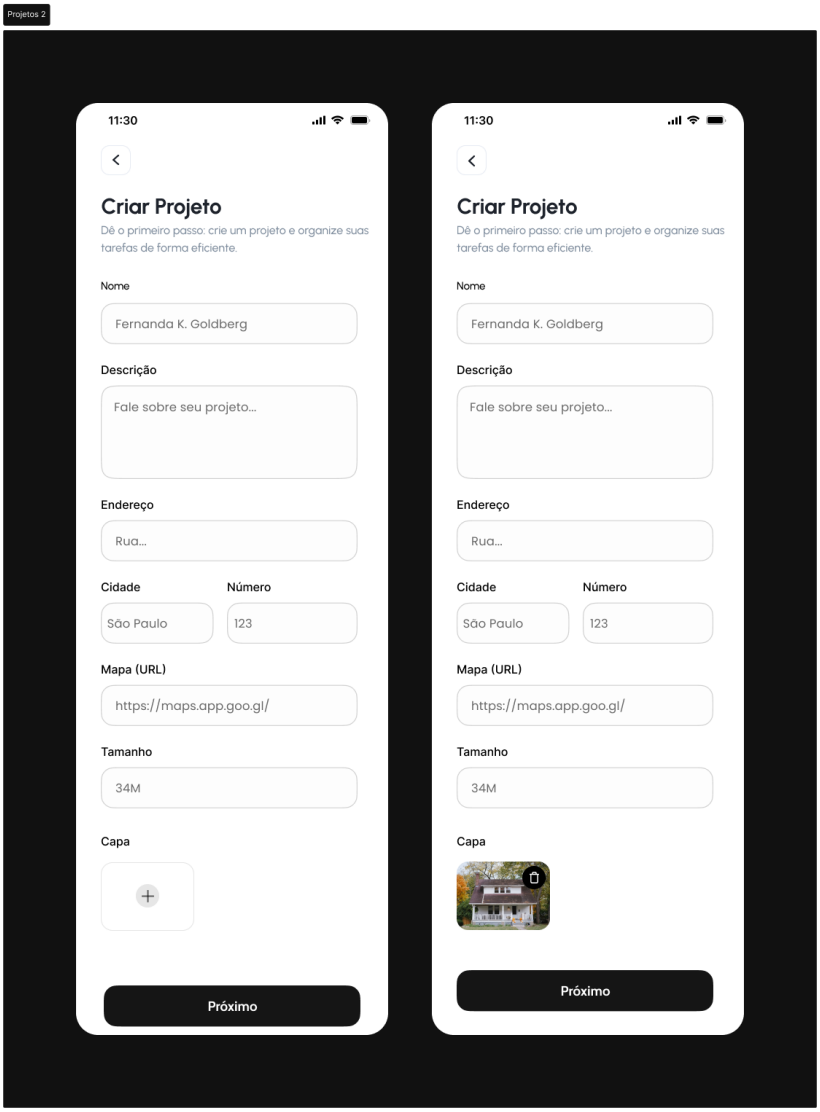
Figura 82: Projeto em 360 e Criar Projeto



Fonte: Autoria Própria

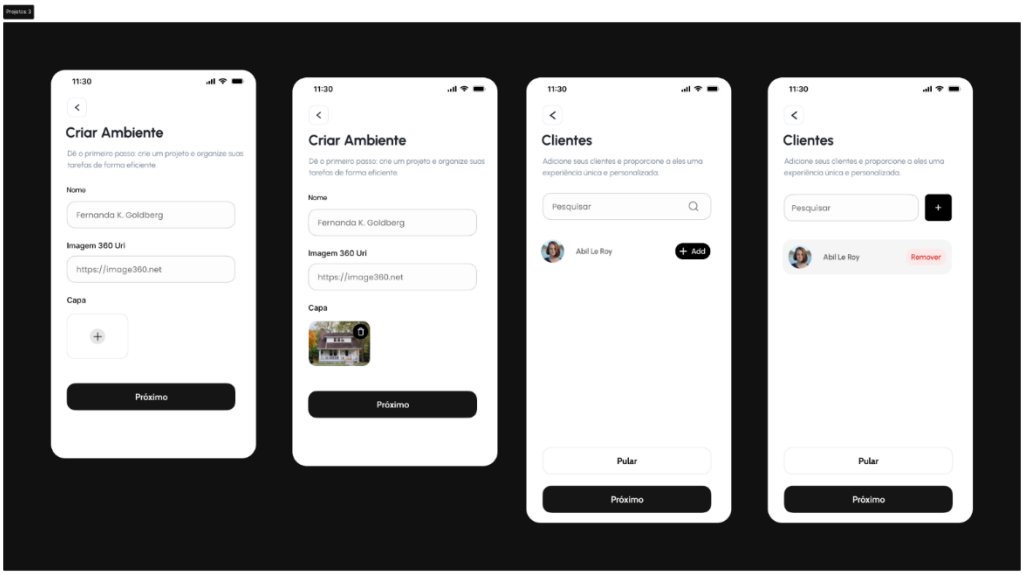
As telas de criação de projeto contêm todas as informações que o usuário precisa inserir para armazenar seus projetos no aplicativo. E o usuário pode adicionar seu cliente a seu projeto.

Figura 83: Criar Projetos



Fonte: Autoria Própria

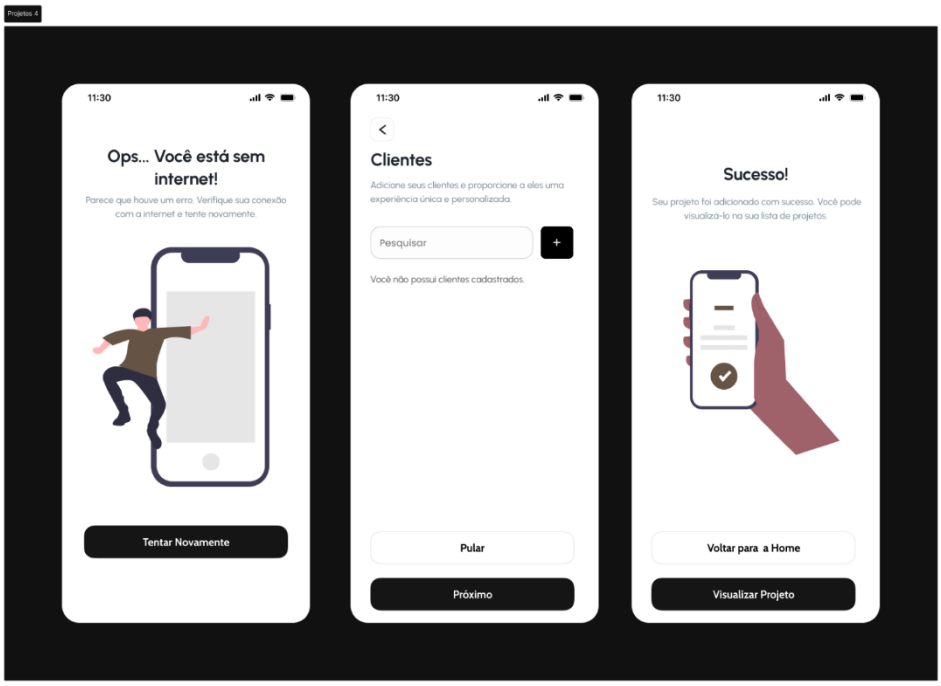
Figura 84: Criar Projetos 2



Fonte: Autoria Própria

As telas a seguir são essenciais para informar ao usuário que uma etapa foi concluída com sucesso, alertá-lo sobre a ausência de conexão com a internet e exibir uma mensagem de retorno caso ele ainda não tenha adicionado clientes.

Figura 85: Telas Extras



Fonte: Autoria Própria

6. CONCLUSÃO

A conclusão deste trabalho reafirma a relevância e o impacto positivo do ARQ360 no cenário atual, evidenciando sua importância para atender a uma lacuna existente na sociedade. O projeto demonstrou eficiência ao integrar tecnologia e arquitetura, alcançando um elevado índice de aceitação por parte dos usuários. O feedback recebido destacou a praticidade e a funcionalidade da plataforma, confirmando o sucesso da proposta.

Os resultados obtidos foram excepcionais, validando as hipóteses levantadas e cumprindo todos os objetivos estabelecidos. O cronograma foi rigorosamente seguido, garantindo entregas dentro dos prazos e assegurando a qualidade do produto final. A aplicação dos recursos adquiridos na instituição de ensino foi fundamental, possibilitando a execução de um projeto alinhado às demandas reais do mercado.

Além disso, o ARQ360 proporcionou uma experiência diferenciada para arquitetos e seus clientes, promovendo maior eficiência, organização e facilidade de uso. A integração de funcionalidades como visualização em 360 graus e armazenamento em um único ambiente trouxe avanços significativos na prática profissional.

Por fim, o projeto, mesmo em constante evolução, já demonstra seu potencial como ferramenta indispensável para o setor. Com melhorias contínuas, o ARQ360 busca oferecer experiências cada vez mais completas e satisfatórias, reafirmando seu compromisso com a inovação e a qualidade.

REFERÊNCIAS

ABNT. Associação Brasileira de Normas Técnicas, 2022, Rio de Janeiro.

ARCHTRENDS PORTOBELLO. Realidade Virtual na Arquitetura. Archtrends, 2020. Disponível em: https://blog.archtrends.com/realidade-virtual-na-arquitetura/. Acesso em: 29 ago. 2024.

CLIQUE ARQUITETURA. Realidade Virtual na Arquitetura. Clique Arquitetura, [s.d.]. Disponível em: https://www.cliquearquitetura.com.br/artigo/realidade-virtual-na-arquitetura.html. Acesso em: 29 ago. 2024.

COMMERCIAL ARCHITECTURE. The impact of virtual reality on architectural design. Commercial Architecture, 13 mar. 2023. Disponível em: https://www.commercialarchitecturemagazine.com/the-impact-of-virtual-reality-on-architectural-design/. Acesso em: 22 ago. 2024.

CURSOS PM3. App design. Cursos PM3, 1 de março de 2024. Disponível em: https://www.cursospm3.com.br/blog/app-design/. Acesso em: 05 ago. 2024 às 10:47.

DIANA, Daniela. O que é Arquitetura**?** Toda Matéria, [s.d.]. Disponível em: https://www.todamateria.com.br/o-que-e-arquitetura/. Acesso em: 29 ago. 2024.

HIGHLANDER FULLSTACK. Dominando NestJS: liberando o poder da arquitetura limpa e DDD no desenvolvimento de comércio. Medium, 21 jun. 2023. Disponível em: https://medium.com/@highlanderfullstack/dominando-nestjs-liberando-o-poder-da-arquitetura-limpa-e-ddd-no-desenvolvimento-de-com%C3%A9rcio-48c64b82d0e7. Acesso em: 19 ago. 2024.

MACDESIGN. Entenda como é a influência da tecnologia na arquitetura. MacDesign, 05 de fev. de 2018. Disponível em: http://macdesign.com.br/blog/entenda-como-e-a-influencia-da-tecnologia-na-arquitetura/. Acesso em: 04 ago. 2024.

MEU PASSEIO VIRTUAL. Como usar imagens 360° para apresentar seus projetos. Meu Passeio Virtual, 17 mai. 2022. Disponível em: https://blog.meupasseiovirtual.com/2022/05/17/como-usar-imagens-360-para-apresentar-seus-projetos/. Acesso em: 19 ago. 2024.

MIRO. O que é brainstorming? Miro**,** 2024. Disponível em: https://miro.com/pt/brainstorming/o-que-e-brainstorming/#o-que-%C3%A9-brainstorming. Acesso em: 13 ago. 2024.

RE-THINKING THE FUTURE. Methods of communication for architects. Rethinking The Future, 2024. Disponível em: https://www.re-thinkingthefuture.com/architectural-community/a6013-methods-of-communication-for-architects/#google_vignette. Acesso em: 22 ago. 2024.

LIVROS DE MARKETING. Livro "Não me faça pensar". Disponível em: https://www.livrosdemarketing.com.br/marketing/livro-nao-me-faca-pensar/#:~:text=Etapa%201%20%E2%80%93%20Princ%C3%ADpios%20B%C3%A1sicos.%20Cap%C3%ADtulo%201. Acesso em: 7 out. 2024.

ALURA. 10 heurísticas de Nielsen: uma fórmula pra evitar erros básicos de usabilidade. Disponível em: https://www.alura.com.br/artigos/10-heuristicas-de-nielsen-uma-formula-pra-evitar-erros-basicos-de-usabilidade. Acesso em: 7 out. 2024.

TUTORIALIA. (n.d.). Teoria das cores: o que é e como surgiu. Disponível em: https://tutorialia.com.br/teoria-das-cores-o-que-e-como-surgiu/. Acesso em: 8 out. 2024

ALURA. HTTP: Entendendo a web por baixo dos panos. Alura, 2023. Disponível em: https://www.alura.com.br/artigos/http. Acesso em: 15 mar. 2024.

ALURA. Design Systems na prática: exemplos e cases de sucesso. Alura, 2023. Disponível em: https://www.alura.com.br/artigos/design-systems-exemplos-praticos. Acesso em: 23 jun. 2024.

ATLASSIAN. Guia do Scrum. Atlassian, 2023. Disponível em: https://www.atlassian.com/br/agile/scrum. Acesso em: 07 ago. 2024.

AUTH0. Refresh Tokens: Quando e como utilizá-los. Auth0 Blog, 2023. Disponível em: https://auth0.com/blog/pt-refresh-tokens-what-are-they-and-when-to-use-them/. Acesso em: 12 mai. 2024.

BLIP. Como documentar uma API REST. Blip Blog, 2023. Disponível em: https://www.blip.ai/blog/tecnologia/documentar-api/. Acesso em: 28 fev. 2024.

BRIAN, K. Creating a Shared Module in NestJS: Benefits and Use Cases. Medium, 2023. Disponível em: https://medium.com/@briankworld/creating-a-shared-module-in-nestjs-benefits-and-use-cases-6292a1dcd200. Acesso em: 19 set. 2024.

COOPERSYSTEM. Requisitos funcionais e não funcionais: o que são e qual é a diferença? Coopersystem, 2023. Disponível em: https://www.coopersystem.com.br/requisitos-funcionais-e-nao-funcionais-o-que-sao-e-qual-e-a-diferenca/. Acesso em: 03 jul. 2024.

CUBOS ACADEMY. UX/UI Design: Guia Completo para Iniciantes. Cubos Academy Blog, 2023. Disponível em: https://blog.cubos.academy/ux-ui-design-guia-completo/. Acesso em: 14 abr. 2024.

DEVMEDIA. Modelagem de Software com UML. DevMedia, 2023. Disponível em: https://www.devmedia.com.br/modelagem-de-software-com-uml/20140. Acesso em: 25 out. 2024.

EWALLY. O que é Backend? Guia completo para iniciantes. Ewally Blog, 2023. Disponível em: https://www.ewally.com.br/blog/ajudando-sua-empresa/backend. Acesso em: 09 mar. 2024.

FREECODEMAP. Como Personalizar Arquivos .env do Node.js. FreeCodeCamp, 2023. Disponível em: https://www.freecodecamp.org/portuguese/news/como-personalizar-arquivos-envdo-node-js-para-diferentes-estagios-do-ambiente/. Acesso em: 17 nov. 2024.

GEOVAN, E. Documentação do NestJS: Organização de Pastas, Padrões de Nomes e Padrões de Projeto. Medium, 2023. Disponível em: https://medium.com/@geovanent/documentação-do-nestjs-organização-de-pastas-padrões-de-nomes-e-padrões-de-projeto-cae91b8af304. Acesso em: 30 ago. 2024.

IBM. O que é API? IBM, 2023. Disponível em: https://www.ibm.com/br-pt/topics/api. Acesso em: 22 mai. 2024.

JAO DEV. Docker para Iniciantes: Criando Containers de Bancos de Dados. Dev.to, 2023. Disponível em: https://dev.to/jao_dev/docker-para-iniciantes-criando-containers-de-bancos-de-dados-291i. Acesso em: 11 jun. 2024.

LINHA DE CÓDIGO. UML - Unified Modeling Language: Atores, Atividades e Componentes. Linha de Código, 2023. Disponível em: http://www.linhadecodigo.com.br/artigo/853/uml-unified-modeling-language-atores-atividades-e-componentes.aspx. Acesso em: 16 set. 2024.

LUCIDCHART. O que é UML? Lucidchart, 2023. Disponível em: https://www.lucidchart.com/pages/pt/o-que-e-uml. Acesso em: 28 jul. 2024.

LUMIS. Metodologias Ágeis: O que são e como aplicar. Lumis, 2023. Disponível em: https://www.lumis.com.br/a-lumis/blog/metodos-ageis.htm. Acesso em: 05 fev. 2024.

MESTRES DA WEB. Requisitos Funcionais e Não Funcionais: O que são? Mestres da Web, 2023. Disponível em: https://www.mestresdaweb.com.br/tecnologias/requisitos-funcionais-e-nao-funcionais-o-que-sao. Acesso em: 13 out. 2024.

MICROSOFT. Padrão MVVM (Model-View-ViewModel). Microsoft Learn, 2023. Disponível em: https://learn.microsoft.com/pt-br/dotnet/architecture/maui/mvvm. Acesso em: 20 mar. 2024.

MIRO. O que é Diagrama Entidade Relacionamento? Miro, 2023. Disponível em: https://miro.com/pt/diagrama/o-que-e-diagrama-entidade-relacionamento/. Acesso em: 08 nov. 2024.

NEON. Documentation. Neon Tech, 2023. Disponível em: https://console.neon.tech/. Acesso em: 14 ago. 2024.

NESTJS. Documentation. NestJS, 2023. Disponível em: https://docs.nestjs.com/. Acesso em: 26 mai. 2024.

NIELSEN, J. 10 Heurísticas de Nielsen para o Design de Interface. UX Design Brasil, 2023. Disponível em: https://brasil.uxdesign.cc/10-heurísticas-de-nielsen-para-o-design-de-interface-58d782821840. Acesso em: 02 abr. 2024.

NODEJS. Documentation. Node.js, 2023. Disponível em: https://nodejs.org/en. Acesso em: 19 jul. 2024.

OPERACIONAL TI. UML: Diagrama de Casos de Uso. Medium, 2023. Disponível em: https://medium.com/operacionalti/uml-diagrama-de-casos-de-uso-29f4358ce4d5. Acesso em: 07 mar. 2024.

PRISMA. Documentation. Prisma, 2023. Disponível em: https://www.prisma.io/docs. Acesso em: 21 set. 2024.

REACT. Documentation. React, 2023. Disponível em: https://react.dev/learn. Acesso em: 15 jun. 2024.

REACT NATIVE. StyleSheet. React Native Documentation, 2023. Disponível em: https://reactnative.dev/docs/stylesheet. Acesso em: 29 ago. 2024.

REACT. Componentes e Props. React Documentation, 2023. Disponível em: https://pt-br.legacy.reactjs.org/docs/components-and-props.html. Acesso em: 04 out. 2024.

ROCK CONTENT. O que é Banco de Dados? Rock Content Blog, 2023. Disponível em: https://rockcontent.com/br/blog/banco-de-dados/. Acesso em: 18 fev. 2024.

ROCKETSEAT. Módulos no NestJS. Rocketseat Blog, 2023. Disponível em: https://blog.rocketseat.com.br/modulos-no-nestjs/. Acesso em: 10 nov. 2024.

RYAN, K. Estrutura de Pastas para React Native. Medium, 2023. Disponível em: https://medium.com/@ryannnkl/estrutura-de-pastas-para-react-native-c2f2d50bd09. Acesso em: 01 mai. 2024.

SIGNIFICADOS. Teoria das Cores. Significados, 2023. Disponível em: https://www.significados.com.br/teoria-das-cores/. Acesso em: 24 abr. 2024.

STRIPE. Documentation. Stripe, 2023. Disponível em: https://docs.stripe.com/. Acesso em: 06 set. 2024.

SUPABASE. Documentation. Supabase, 2023. Disponível em: https://supabase.com/docs. Acesso em: 12 ago. 2024.

TREINAWEB. Fluxo de autenticação baseado em JWT. TreinaWeb Blog, 2023. Disponível em: https://www.treinaweb.com.br/blog/fluxo-de-autenticacao-baseado-em-jwt. Acesso em: 27 jul. 2024.

VERCEL. Documentation. Vercel, 2023. Disponível em: https://vercel.com/docs. Acesso em: 31 mar. 2024.

WANNY, W. Levantamento de Requisitos. Medium, 2023. Disponível em: https://medium.com/@wonderwanny/levantamento-de-requisitos-bda17b972d58. Acesso em: 09 out. 2024.

WIKIPEDIA. Design. Wikipédia, 2023. Disponível em: https://pt.wikipedia.org/wiki/Design. Acesso em: 16 mai. 2024.