Documentação do Projeto Sistema de Transferência Bancária - *Ceub Banking*

Curso: Engenharia de *Software*

Disciplina: Engenharia de *Software*

Professor: Elias Do Nascimento Melo Filho

Turno: Vespertino

Integrantes da equipe:

- Gabriella Mendes
- Lara Geovana Elias Silva
- Leticia Lafetá Naves
- Maria Luiza Mendonça Wanderlei

Introdução

Inicialmente, é importante destacar que esse trabalho foi desenvolvido nos moldes de um piloto, já que foi o primeiro desenvolvido em Modelo Cascata pelas partes envolvidas. A definição do tema foi encaixada nos seguintes parâmetros primários:

1) Deveria ser de fácil execução, ou seja, a ideia deveria ser simples.

2) Deveria requerer pouco conhecimento prévio acerca de: *design* gráfico, programação em HTML, manutenção de *software* e uso da plataforma usada *Figma.*

3) Deveria ter rápida execução, para que a aprendizagem e a entrega das características do projeto fossem resolutas no prazo postado pelo professor (1 mês).

Nesse sentido, se propõe os seguintes objetivos:

1) Criar um *design* pronto e ideal para a execução de um *backend*, que não ocorrerá no projeto, mas será constantemente visado e planejado.
2) Utilizar ferramentas que atendam os parâmetros acima e que sejam de utilização variada ao longo da área de engenharia de *software*.

3) Aprender a respeito do Modelo Cascata por meio da sua implementação no trabalho.

Diante disso, após todas as discussões tomarem lugar no *WhatsApp* e de ouvir a opinião do professor da disciplina, foi consensual de que seria possível a realização do *Sistema de Transferência Bancária,* uma vez que atenderia os pré-requisitos listados acima e, também, expandiria os limites das alunas. Sendo assim, é justo salientar quais e para quê servem as ferramentas foram usadas nesse arquivo:

- ***Modelo Cascata***: modelo pioneiro para a construção de produtos digitais que alinhassem as demandas de um cliente com código e interfaces. Dentre as suas principais características estão: detalhada documentação e as suas etapas convencionadas - requisitos, design, implementação, verificação, implantação, verificação e manutenção.
- ***Modelo ABNT***: conjunto de normas da Associação Brasileira de Normas Técnicas (ABNT) que padroniza a formatação de trabalhos acadêmicos e científicos no Brasil. Essa documentação será apresentada conforme tais convenções.
- ***Figma***: plataforma digital destinada ao desenvolvimento de interfaces UI e de prototipagem interativa, que foi usada para exportar e acrescentar os estágios finais da documentação e criar todas as interfaces do trabalho.
- ***Dribbble***: comunidade digital destinada ao compartilhamento de diversas ideias de *design*, que foi usada para orientar e inspirar os designs logo apresentados.
- ***Word***: plataforma destinada a criação de textos usada para os estágios iniciais dessa documentação.
- ***Krita***: *software* de edição e criação de imagens e desenhos. Foi utilizado no auxílio de imagens e logotipos do design.
- ***DB Designer***: ferramenta utilizada para diagramar o modelo de Banco de Dados idealizado para esse trabalho e criar uma modelagem de dados fiel ao MER (Modelo Entidade-Relacionamento).
- ***WhatsApp***: rede social de conversas remotas e instantâneas que sediou todas as decisões sobre os requisitos e os processos pré-concluintes do produto.

- ***Youtube***: rede social usada para pesquisa de conhecimentos que atendessem as demandas dessa atividade.

Logo, inicia-se a documentação progressiva desta via Modelo Cascata.

1) Primeira Etapa

Requisitos e Metas

Análise e definição de requisitos. Os serviços, restrições e metas do sistema são estabelecidos por meio de consulta aos usuários. Em seguida, são definidos em detalhes e funcionam como uma especificação do sistema.

Sumário de Requisitos

1) Quanto às funções principais do *app*, que definem o seu propósito, que é a transferência de valores entre uma conta e outras – que não sejam necessariamente do mesmo banco.

1.1) Função PAGAR:

Essa função consta na inserção manual do usuário de um valor específico – à sua escolha – para um destinatário específico sistema e deverá ser exibida toda e qualquer quantia paga no campo de ENTRADAS E SAÍDAS.

- Envio de um valor específico:

  - O próprio usuário inserirá a quantia desejada em uma caixa destinada a retê-la no sistema.

- Forma de envio ao destinatário:

  - Chave Pix, que corresponda a:

    - Número de celular
    - *E-mail*
    - CPF

1.2) Função RECEBER:

Essa função será operada pelo próprio sistema e deverá ser exibida toda e qualquer quantia recebida no campo de ENTRADAS E SAÍDAS.

2) Quanto às funções secundárias:

2.1) Função CADASTRO:

Essa função efetua a entrada dos dados do usuário no Banco de Dados e permite o seu acesso a todas as funcionalidades da plataforma.

2.2) Função LOG IN:

Essa operação acessará o Armazenamento de Dados à procura dos dados de acesso do usuário. Uma vez encontrados, libera acesso ao aplicativo novamente**.**

2.3) Função VERIFICAÇÃO DO STATUS do pagamento:

A Verificação do *status*

3) Quanto ao *design*, que confere a identidade visual solicitada pelo cliente ao produto.

3.1) O protótipo de *design* será realizado na plataforma de desenvolvimento UI e UX Figma. Além disso, para o auxílio em determinados elementos do *design* foi utilizado o Krita.

3.2) O *design* deve ser inspirado no *website* da faculdade UniCEUB (Centro Universitário de Brasília) – de endereço *www.uniceub.com*, que orientará as desenvolvedoras quanto a seleção de quais cores, *layouts* de texto e ícones, como ilustrações, devem ser utilizadas. Obrigatoriamente, essas escolhas devem estar dentro de parâmetros que inegavelmente conferem identidade a instituição.

3.3) O *design* consiste em:

- *Layout* de texto
  - Tamanho do texto;
  - Fonte do texto (*Inter*, em sua maioria);
  - Cor do texto;
  - Espaçamento entre as letras;
  - Espaçamento entre duas ou mais caixas de texto;
  - Alinhamento de texto
  - Posição do texto dentro do plano cartesiano – referência de localização usada na plataforma *Figma*.

        - Opacidade do texto.
    - Caixas informacionais
        - Taxa de arredondamento das “quinas” da caixa.
        - Formato da caixa.
        - Função da caixa:
            - Botão;
            - Implantação de texto;
            - Implantação de imagens;
            - Inserção de dados, que e controlada pela imputação do usuário.
    - Imagens de formato .png importadas do *site* da instituição UniCEUB e do endereço Google Imagens. Submetidas a:
        - Alteração de tamanho;
        - Alteração de opacidade;
        - Recortes em suas extensões;
        - Rotação;
        - Pintura
    - Layout de página
        - O layout da página será feito nos moldes de um dispositivo IOS (Iphone 13 Mini).

Considerações:

I. Foi consensual entre as partes do desenvolvimento do *software* que, para diminuir o tempo de realização e os custos, especialmente com fins de agradar o cliente, boa parte do design será reaproveitada das bibliotecas do aplicativo *Figma*, uma vez que já possuem um padrão de qualidade testado e o novo produto poderá se beneficiar de manutenções realizadas no protótipo original.

II. Acerca das funções executáveis por botões, estão: submeter informações digitadas pelo usuário, redirecionar para outra interface existente e confirmar uma ação oferecida pelo *software*. Diante disso, e importante ressaltar que essas atividades são acionadas na Visão do *software*, ou seja, de controle do usuário.

4) Quanto ao código, teste e manutenção:

Novamente, foi consensual entre as partes do desenvolvimento do *software* que, para diminuir o tempo de realização e os custos, especialmente com fins de agradar o cliente em rapidez e praticidade, boa parte do código, dos testes e da manutenção será reaproveitada das bibliotecas do aplicativo *Figma*, uma vez que já possuem um padrão de qualidade testado e o novo produto poderá se beneficiar de manutenções realizadas no protótipo original.

5) Informações gerais a respeito do *software*:

I. Deve ser chamado de "*Ceub Banking*" e deve conter a identidade visual da marca UniCEUB.

II. Deve ser feito para o programa IOS e deve ser adequado para o dispositivo Iphone 13 Mini.

III. Público-alvo:

- Jovens Adultos (18-24 anos): Este grupo pode estar interessado em contas bancárias básicas, cartões de débito e crédito, e produtos de poupança.
- Adultos (25-40 anos): Focam em produtos como financiamentos, empréstimos pessoais, investimentos e previdência.
- Meia-Idade (41-60 anos): Podem buscar soluções de investimento, planejamento de aposentadoria e seguros.
- Idosos (acima de 60 anos): Geralmente interessam-se por investimentos seguros, produtos de aposentadoria e gerenciamento de patrimônio.

IV. Deve possuir Banco de Dados – para que haja armazenamento de dados e eliminação de inconsistências no sistema.

2) Segunda Etapa

Projeto de sistema e software

O processo de projeto de sistemas aloca os requisitos tanto para sistemas de *hardware* como para sistemas de *software*, por meio da definição de uma arquitetura geral do sistema. O projeto de *software* envolve identificação e descrição das abstrações fundamentais do sistema de *software* e seus relacionamentos.

1) PROJETO DE SISTEMA

Visão geral da arquitetura do sistema

1.1) Quanto ao tipo de arquitetura

Arquitetura monolítica.

1.2) Quanto ao diagrama arquitetural de alto nível

[Cliente] <-> [Frontend (HTML, CSS, JS)] <-> [Backend (NestJs, C++)] <-> [Banco de Dados (MySQL)]

|--> [Cache (Redis)]

|--> [Fila de Processamento (RabbitMQ)]

|--> [Gateway Bancário (PIX)]

Observação: algumas características destacadas a seguir fazem parte do plano de implementações do projeto – ou seja, estão ilustradas apenas para iluminar a possível estrutura final do produto para o solicitante. Assim sendo, ainda não há resolução quanto aos tópicos: Cache, Sistema de Gerenciamento de Banco de Dados (SGBD) e Fila de Processamento.

2) PROJETO DE *SOFTWARE: DESIGN*

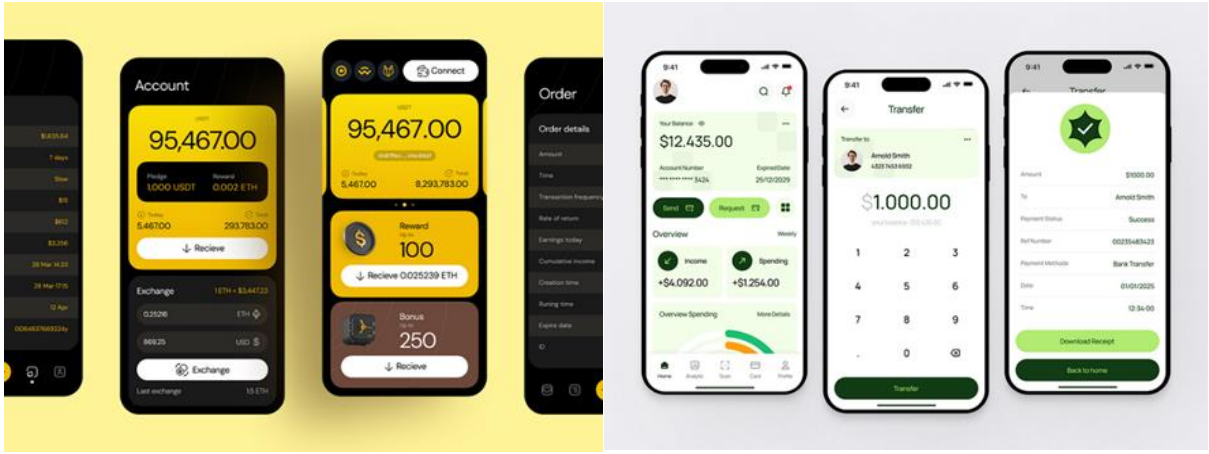
Visão das interfaces e diagramação de funções principais e secundárias

A Visão das interfaces desempenha as funções de I. conferir IDENTIDADE VISUAL ao produto, II. oferecer a estrutura necessária para que o usuário PAGUE o valor desejado ao destinatário de preferência e RECEBA valores de supostos remetentes e III. CADASTRE-SE no Banco de Dados, faça *LOG IN* no aplicativo e VERIFIQUE O STATUS de pagamento realizado.

I. Quanto às inspirações estudadas para definição do design

Dribbble

- A plataforma foi usada com o objetivo de avaliar como desenvolvedores UX ao redor do mundo percebem bancos de transferência monetária. Definitivamente, foi um ponto orientador para a determinação do "rosto" do aplicativo. Vide algumas das amostras estudadas:



Imagens retiradas diretamente da plataforma Dribbble

II. Quanto a identidade visual do *app Ceub Banking*

Logotipo e Variantes

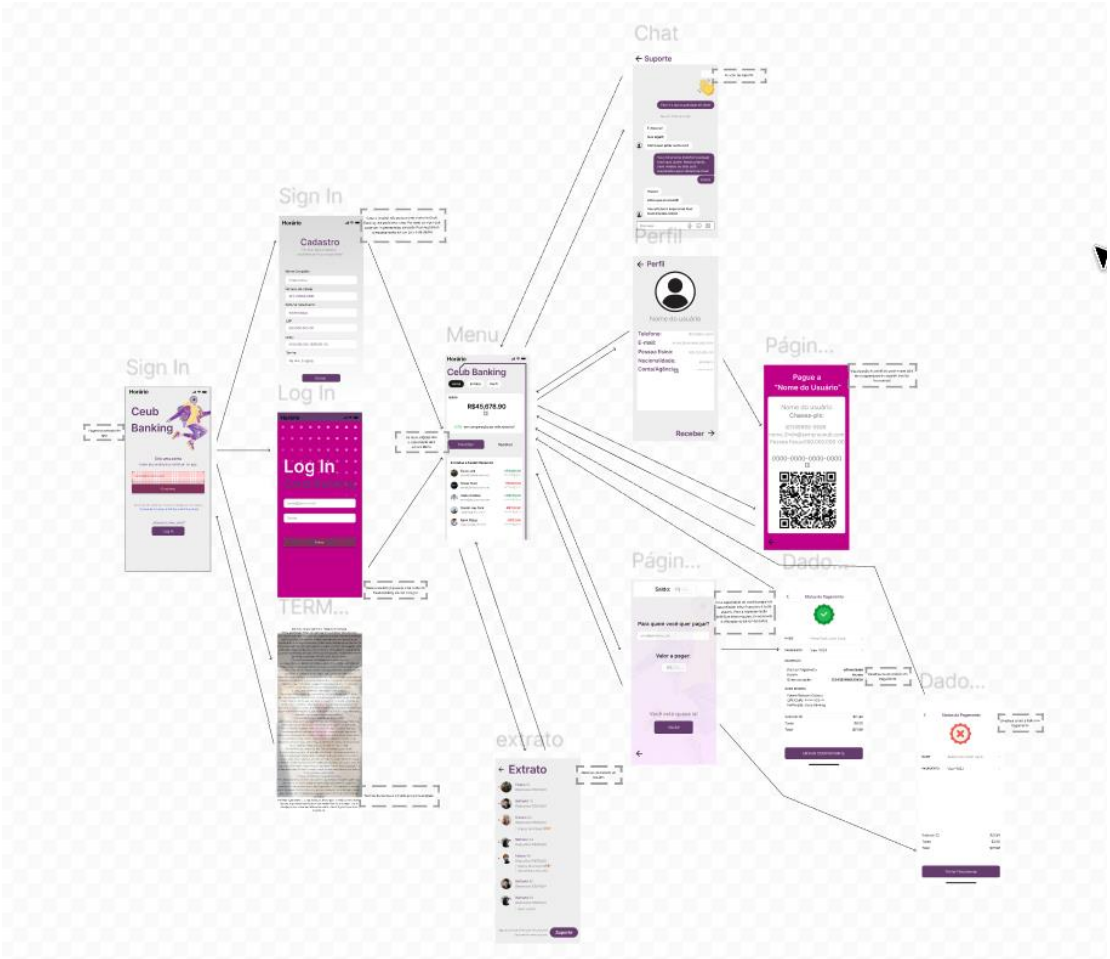
- O logotipo utilizado ao longo do *design* que identifica como semelhando ao da universidade é a caixa de texto contendo "Ceub" em determinadas regras:
- **Tais regras de uso são**:
  - Fonte Inter
  - Fundo branco (FFFFFF) ou levemente cinza (EEEEEE)

- Cor do texto sendo roxa tipicamente com a da universidade (663B6F)
- Com variações de tamanho de texto diferentes

Ícones e Elementos Gráficos

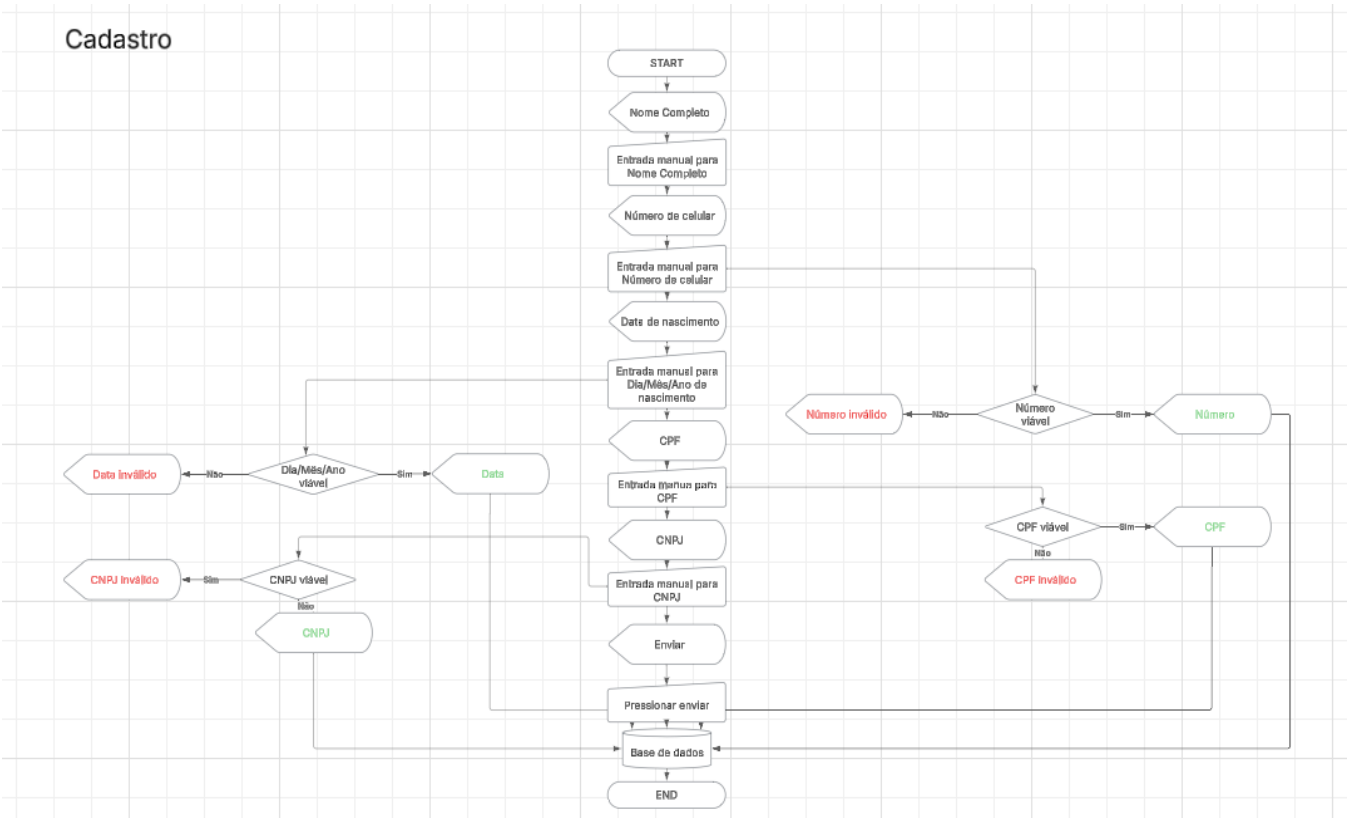
- Ao todo, foi utilizado apenas um ícone, retirado de redes públicas, que se refere a foto vazia do usuário
- De ícones personalizados foi criado uma imagem, com auxílio do Krita, que se assemelha com uma das imagens encontradas no site da universidade. Tal elemento foi feito e mantido de forma caricata, para que apenas houvesse a semelhança entre os designs da universidade e do protótipo.
- Os botões do design foram decididos por serem roxos, apesar disso, há interações diferentes entre eles.
- Os inputs foram representados com uma caixa branca ou levemente cinza, de fácil identificação, e descrita no documento do Figma.
- Nenhum filtro foi aplicado a nenhuma página do design, foram utilizados apenas mudanças de cores ao longo dos elementos.

As interfaces foram desenvolvidas na plataforma Figma, que se mostrou o aplicativo ideal para as habilidades da equipe de UX design. Para promover praticidade nesse documento, foi decidido que a visualização completa do design permanecerá sendo acessado exclusivamente na plataforma, porém de acesso público por via da URL disposta no rodapé desse exemplar. Segue uma amostra sem prototipagem.



I. Quanto à estrutura necessária para que o usuário PAGUE o valor desejado ao destinatário de preferência e RECEBA valores de supostos remetentes.
II. Quanto à estrutura necessária para CADASTRO no Banco de Dados, faça *LOG IN* no aplicativo, VERIFICAÇÃO DE STATUS de pagamento.
    - Quanto ao Cadastro:

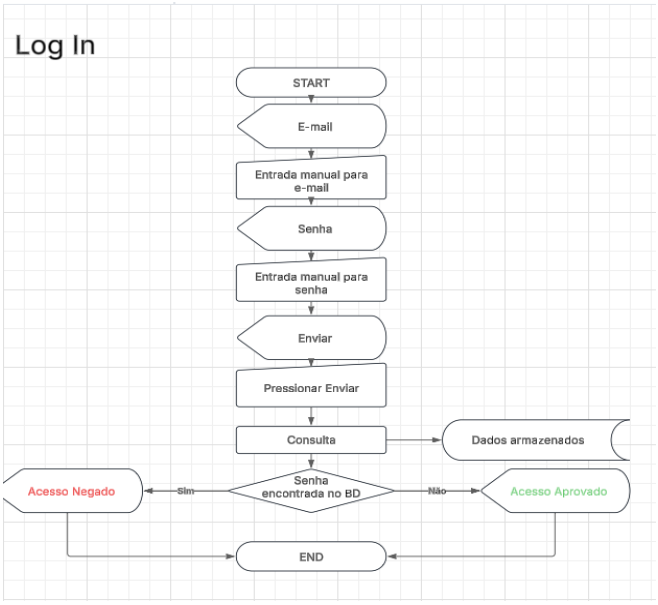
Essa função se resume em colocar todos os dados necessários para a criação de um perfil interessante para a empresa. Os dados são: Nome completo, número de celular, data de nascimento, *e-mail*, CPF e CNPJ.



Fluxograma da função de cadastro.

- Quanto ao *Log In:*

O *Log In* é realizado apenas com *e-mail* e senha. Caso a senha já tenha sido registrada com sucesso no Banco de Dados do *app* e foi inserida corretamente no campo designado, o usuário terá livre acesso à sua conta. Caso contrário, a página de *Log In* permanecerá indicando “senha inválida” na entrada dedicada à senha.

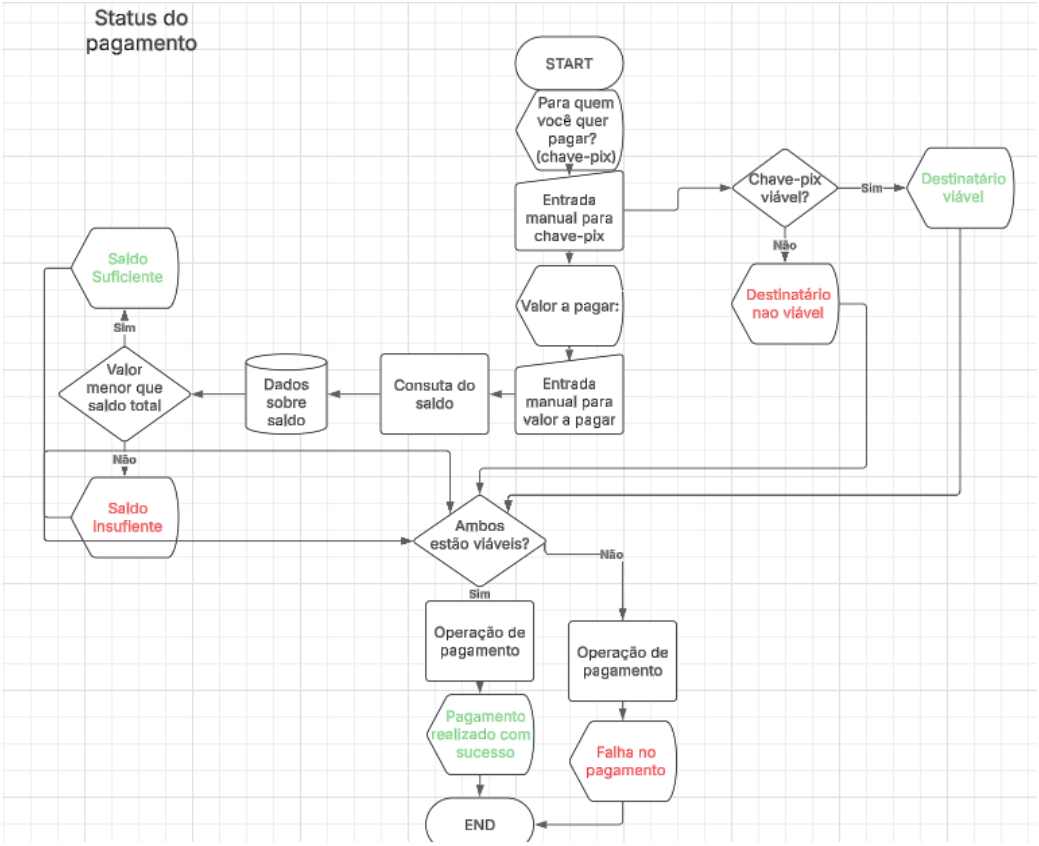


Fluxograma da função de Log In.

- Quanto à Verificação de *Status* do pagamento

Essa análise deve coletar dados sobre saldo suficiente, qual será o valor que o usuário deseja transferir e a viabilidade do remetente. Obrigatoriamente, o valor a ser transferido deve ser menor que o saldo total no instante de efetuar tal transferência e a

chave-pix deve ser viável para que o pagamento seja realizado com sucesso.



Fluxograma da função Status do Pagamento

- Agradecimento ao Prof. Wenderson Alexandre de Souza Silva por prestar ajuda às alunas durante o desenvolvimento desse tópico.

1) PROJETO DE SISTEMA: ARMAZENAMENTO DE DADOS

O armazenamento de dados será realizado pela sincronização de dados inseridos manualmente pelo usuário com o Banco de Dados projetado pela equipe de desenvolvimento.

Portanto, tornou-se essencial ressaltar que, devido às limitações técnicas da equipe, que cursa o primeiro semestre do curso de Engenharia de *Software*, a captação de informação pelo Banco de Dados ainda não é funcional. Segundo esclarecimentos apresentados pelo orientador da disciplina, a equipe compreendeu que não havia necessidade de configurar tal funcionalidade até o prazo estipulado e que essa automação será mandatória apenas em próximas atividades.

Consoante a essas observações, foi preparado um modelo de Banco de Dados, que possui um *Modelo Lógico*. O Modelo Lógico, de James Martin, é uma representação estruturada de dados que foca em **entidades, atributos** e **relacionamentos**. Diante do exposto, o diagrama Entidade-Relacionamento (DER) - correspondente ao vide projeto - terá as seguintes características:

1. Quanto às entidades e seus atributos
    - Entidade forte: tabela com dados acerca do cliente **(*tb_cliente*)**
        - Nome Completo
        - E-mail
        - Número de celular móvel com DDD +55
        - Data de nascimento
        - CPF, que identificará se o representado é uma *pessoa física* (PF).
        - CNPJ, que identificará se o representado é uma *pessoa jurídica* (PJ).
    - Entidade fraca: tabela com dados acerca do destinatário da transação - que não deve ser o cliente analisado (*tb_destinatario_nao_cliente*)
        - Nome completo do então destinatário
        - Banco do então destinatário
    - Entidade fraca: tabela com dados acerca do remetente da transação - que não deve ser o cliente analisado (*tb_remetente_nao_cliente*)
        - Nome completo do então remetente
        - Banco do então remetente
    - Entidade forte: tabela com dados acerca da transação bancária realizada pelo cliente analisado (*tb_transacao_c_d*)
        - Data da transação
        - Valor da transação
        - Remetente (cliente analisado)
        - Destinatário “não cliente”
        - Indicador de caráter da transação: crédito ou débito
    - Entidade forte: tabela com dados acerca da transação bancária destinada ao cliente analisado (*tb_transacao_r_c*)
        - Data da transação
        - Valor da transação
        - Remetente “não cliente”
        - Destinatário (cliente analisado)

    - Indicador de caráter da transação: crédito ou débito
- Entidade fraca: tabela com dados acerca de CPF (*tb_CPF*)
    - Banco de dados importado da Receita Federal do Brasil.
- Entidade fraca: tabela com dados acerca de tipo de pessoa (*tb_tipo_pessoa*)
    - Pessoa Física
    - Pessoa Jurídica
- Entidade fraca: tabela com dados acerca da movimentação da transação (*tb_cred_deb*)
    - Crédito
    - Débito

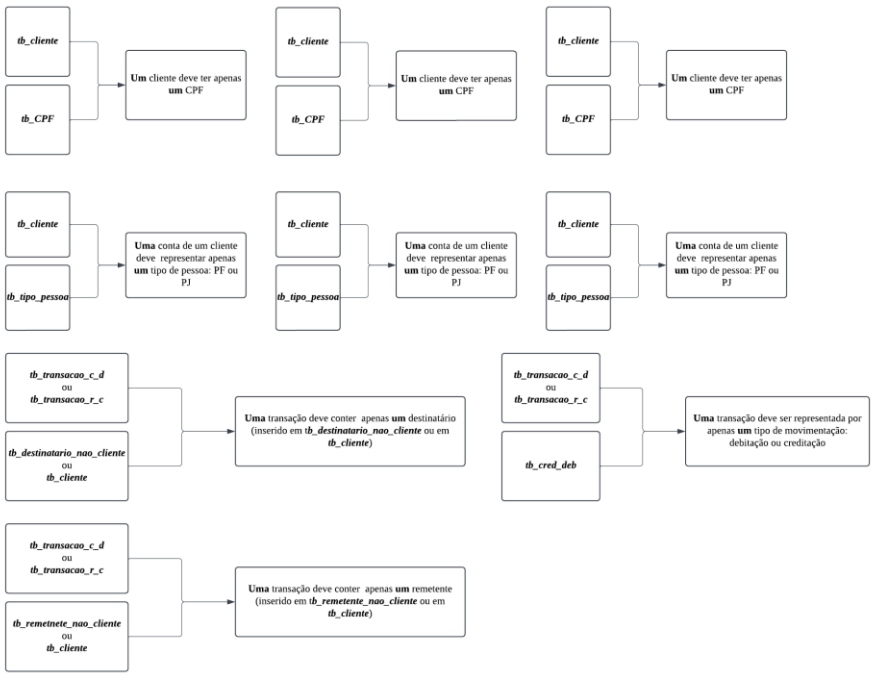
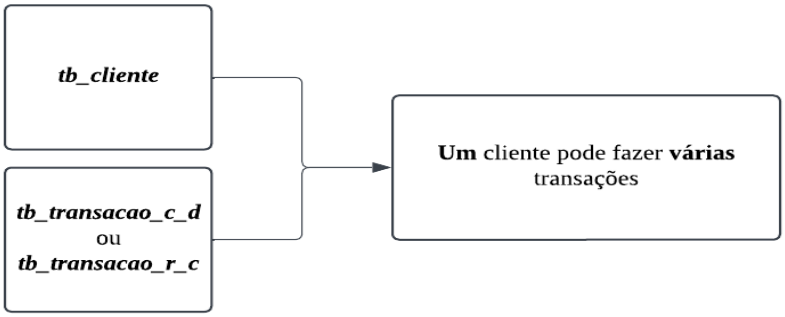


Ilustração gráfica do Banco de Dados

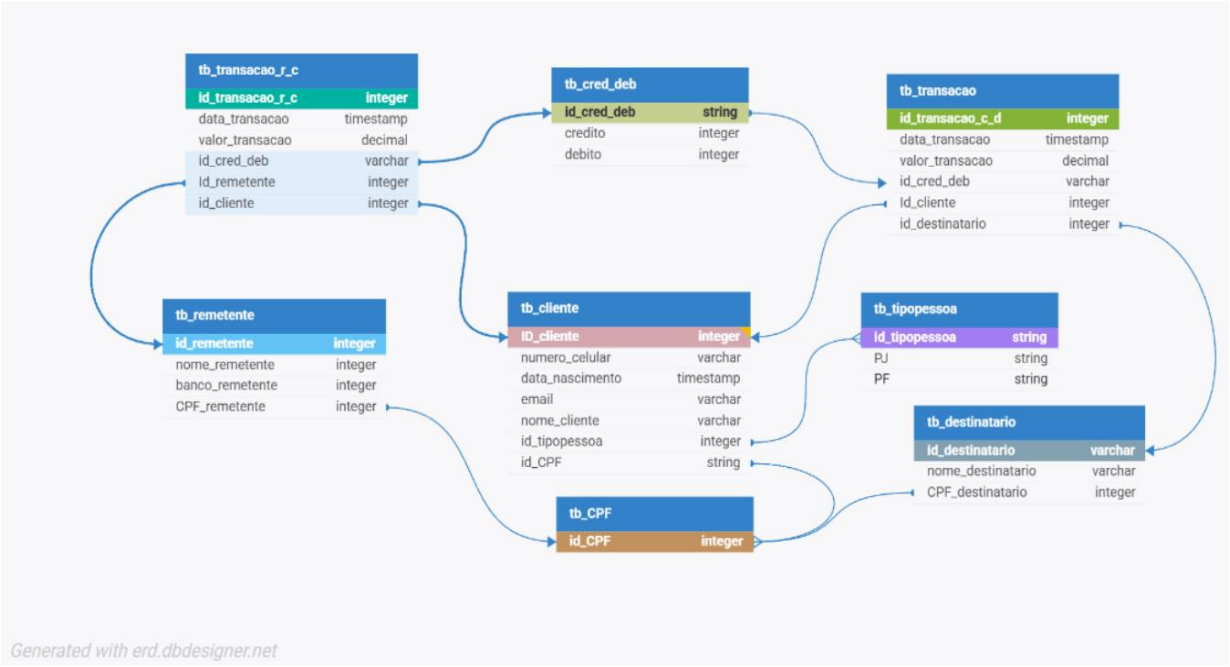
2. Quanto aos relacionamentos:
    - Relacionamento 1:1 (Um para Um)
        - Em uma relação 1:1, cada registro em uma tabela está relacionado a, no máximo, um registro em outra tabela.
    - Relacionamento 1:N (Um para Vários)

- Em uma relação 1:N, um registro em uma tabela pode estar relacionado a múltiplos registros em outra tabela, mas cada registro na segunda tabela está relacionado a apenas um registro na primeira.



Relacionamentos no Banco de Dados

- Modelo Lógico – segundo James Martin



Considerações Finais sobre o Modelo de Banco de Dados:

- As caixas coloridas correspondem a chaves primárias.
- As caixas que possuem em si a ponta da seta acoplada correspondem a chaves estrangeiras.
- As setas podem ser diferenciadas em dois tipos: as simples e as com pé de galinha.

- As setas simples correspondem a relacionamento 1:1.
- As setas com pé de galinha correspondem a relacionamentos 1:N.

❖ Agradecimento ao Prof. José Wellington Cunha da Silva por prestar ajuda às alunas durante o desenvolvimento desse tópico.

2) PROJETO DE *SOFTWARE*: CÓDIGO.

- *Framework Backend* (reaproveitado do próprio *Figma*): C++, *WebSockets*, AWS (*Amazon Web Services*), *WebAssembly* (WASM) e CRDTs (*Conflict-free Replicated Data Types*).
- Banco de Dados **(projeto de implementações**): MySQL
- Tecnologias aplicadas ao *Frontend* (reaproveitado do próprio Figma): HTML, CSS, TypeScript, JavaScript, WebGL, Canvas API e WebAssembly (WASM)
- Autenticação (reaproveitada do próprio *Figma*): *Firebase Authentication.*
- Testes (reaproveitados do próprio *Figma*): *Jest* e *PlayWright.*

1. Quanto à codificação de autoria da equipe

Durante o desenvolvimento do nosso sistema, iniciamos o processo de construção da interface assistindo a vídeos sobre HTML. No entanto, ao colocar o conhecimento em prática, percebemos que o *design* não estava ficando conforme o planejado, pois estávamos utilizando um modelo simples de cadastro.

Diante desse problema, optamos por extrair o código diretamente do Figma, acreditando que isso facilitaria a implementação do *design* desejado. Porém, ao tentar utilizar esse código, verificamos que ele não funcionava corretamente. Além disso, ao realizar edições no *design*, o código apresentava erros que não conseguíamos solucionar.

Isso ocorreu porque o *Figma* gera um código específico para a estrutura de design dele, mas qualquer modificação alterava a formatação, resultando em

falhas na implementação. Buscamos a orientação de professores, que nos explicaram que transformar o *design* gerado no *Figma* em um código funcional era um processo bem mais complexo do que imaginávamos.

Para resolver os problemas encontrados, realizamos diversas tentativas:

- Editamos manualmente o código gerado pelo *Figma;*

• Fizemos ajustes na estrutura do HTML e CSS para tentar compatibilizá-los com o design planejado;

• Testamos diferentes abordagens para corrigir os erros apresentados.

No entanto, mesmo com esses esforços, não conseguimos fazer com que o código funcionasse corretamente.

Com essa experiência, percebemos que a conversão direta do design do *Figma* para código não é um processo simples e exige conhecimentos mais aprofundados de desenvolvimento *web*.

Desde o início do projeto, utilizamos o *Visual Studio Code* (VSCode) como nossa ferramenta essencial para edição e desenvolvimento do código. Através dele, conseguimos visualizar e modificar os arquivos em HTML e CSS, além de testar as implementações realizadas.

O *VSCode* nos permitiu:

- Criar e organizar os arquivos do projeto de maneira eficiente;
- Testar e visualizar as mudanças no código em tempo real;
- Identificar erros e falhas na estrutura do HTML e CSS com mais facilidade;
- Integrar extensões úteis para facilitar a escrita e depuração do código.

Mesmo com todas essas facilidades, enfrentamos desafios ao tentar transformar o *design* do projeto em código funcional.

Contudo, ao tentar integrar o código do Figma no nosso projeto pelo VSCode, enfrentamos diversos problemas:

- O código não funcionava corretamente na estrutura do nosso projeto;
- Ao editar o design no *Figma*, o código gerado mudava e apresentava erros;
- Não conseguimos identificar e corrigir alguns dos erros apresentados.

Essas dificuldades nos levaram a buscar orientações externas para entender melhor como transformar um *design* estático em um código funcional.

Após entender melhor as dificuldades, tentamos corrigir o código de diversas formas:

- Edição manual do HTML e CSS no *VSCode* para ajustar os problemas de *layout*;
- Refatoração do código para melhorar a organização e evitar erros estruturais;
- Testes constantes para identificar quais mudanças estavam causando problemas.

Mesmo com essas tentativas, enfrentamos desafios para tornar o código funcional de acordo com o *design* esperado.

O uso do *VSCode* foi essencial durante todo o processo, proporcionando um ambiente completo para edição e teste do código. No entanto, percebemos que transformar um *design* em código funcional exige mais do que apenas copiar e colar trechos gerados por ferramentas como o *Figma.*

Quanto ao Python, inicialmente, pensamos em desenvolver o sistema bancário usando Python e, por isso, nos dedicamos ao estudo da linguagem. No entanto, ao avançarmos no projeto, percebemos que não precisaríamos utilizá-la. Mesmo assim, o aprendizado foi válido e agregou o conhecimento nessa linguagem.

Quanto ao que é publicizado pelo Figma sobre seus códigos:

> As próximas informações foram extraídas de diferentes artigos encontrados no site www.figma.com, que estarão disponíveis no rodapé desse documento.

1.1) Usar trechos de código no *Dev Mode*

Este recurso permite que desenvolvedores acessem trechos de código diretamente no *Dev Mode* do Figma, agilizando a implementação e garantindo consistência entre o design e o produto final. Ao inspecionar elementos no *Dev Mode*, os desenvolvedores podem visualizar e copiar trechos de código correspondentes, facilitando a tradução precisa dos designs para código.

1.2) Guia para o *Dev Mode*

O *Dev Mode* é uma interface voltada para desenvolvedores dentro do Figma que oferece ferramentas avançadas de inspeção e geração de código. Ele permite que desenvolvedores naveguem por arquivos de design, acessem especificações atualizadas, comparem versões de frames e integrem-se com plataformas como JIRA, *Storybook* e GitHub. Além disso, o *Dev Mode* permite explorar todas as variantes em um conjunto de componentes sem a necessidade de editar o arquivo, vinculando designs a tickets, documentação e componentes de código.

1.2) *Code Connect*

O *Code Connect* serve como uma ponte entre a base de código dos componentes e o Figma. Com ele, é possível levar o código dos componentes do sistema de design diretamente para o *Dev Mode* do Figma, permitindo que os desenvolvedores visualizem exemplos de componentes que refletem a estrutura do código de produção. Isso resulta em trechos de código reais definidos pelo sistema de design, em vez de trechos gerados automaticamente, garantindo maior precisão e consistência na implementação.

3) Terceira Etapa

Implementação e teste unitário

Durante esse estágio, o projeto do *software* é desenvolvido como um conjunto de programas ou unidades de programa. O teste unitário envolve a verificação de que cada unidade atenda a sua especificação. As unidades individuais do programa ou

programas são integradas e testadas como um sistema completo para assegurar que os requisitos do *software* tenham sido atendidos.

1) Quanto ao teste do código gerado pelo *Figma*

Atualmente, o *Figma* não disponibiliza informações públicas detalhadas sobre seus processos internos de teste de *software*. Para obter testagem sobre as estruturas oferecidas, a plataforma possui uma espécie de exame controlado feito pelo próprio usuário, que consiste na fase de prototipagem.

Nessa fase, a ferramenta permite testar a interatividade e a navegabilidade das interfaces e permite que os *designers* identifiquem problemas de *design*, validem hipóteses e refinem a experiência com base em *feedback* concreto. A integração do *Figma* com *UserTesting* oferece a implementação de clipes de vídeo sobre as criações, o que facilita a visualização e a aplicação do molde final do aplicativo. A URL do vídeo de demonstração do protótipo estará na Lista de Referências - ao final do documento.

Futuras Implementações

Nosso objetivo principal para as próximas etapas do sistema bancário é torná-lo totalmente funcional, garantindo uma experiência boa e segura para os usuários. Para isso, planejamos implementar diversas funcionalidades essenciais e melhorias que facilitarão o uso da plataforma.

Principais Funcionalidades a Serem Implementadas

1) **Cadastro e Acesso Seguro**

- Implementação de um sistema de *Sign Up* (cadastro) e *Log In* seguro, com autenticação em dois fatores (2FA) para maior proteção dos usuários.
- Recuperação de senha por *e-mail* ou SMS.

2) **Criação de senha**

- Implementação de segurança que solicita palavra-passe do usuário para que acesse às suas informações de forma privada e protegida. Essa função deverá

oferecer opções de senhas seguras geradas pelo próprio sistema ou, se for de preferência do cliente criar a sua própria, que avalie a força da senha segundo parâmetros pré-definidos de segurança.

3) **Pagamentos e Transferências**

- Transferências entre contas dentro do sistema, com validação de saldo e autenticação do usuário.
- Pagamentos de contas e boletos através de *QR Code* diretamente pela plataforma.
- Opção de agendamento de pagamentos e transferências, permitindo que os usuários programem transações futuras.

4) **Segurança e Confirmação de Transações**

- Implementação de notificações por *e-mail* ou SMS sempre que uma transação for realizada.
- Criação de um código PIN ou autenticação biométrica para confirmar pagamentos e transferências.

5) **Facilidade de Uso e Experiência do Usuário**

- Interface intuitiva e responsiva, acessível tanto em dispositivos móveis quanto em computadores.
- Histórico detalhado de transações, permitindo que o usuário acompanhe todas as suas movimentações financeiras.
- Suporte via *chat* ou assistente virtual para dúvidas e suporte técnico.

6) **Integração com Outras Plataformas**

- Possibilidade de integração com bancos externos, permitindo que os usuários realizem transferências para contas fora do sistema.

- Integração com métodos de pagamento digitais, como Pix, carteiras digitais e cartões de crédito.
- Integração com Banco de Dados a partir do uso de Sistemas de Gerenciamento de Banco de Dados, como o MySQL..

Com essas melhorias, o sistema bancário se tornará mais prático, seguro e eficiente, proporcionando uma experiência completa para os usuários.

4) Quarta Etapa

Operação e manutenção

Normalmente (embora não necessariamente), essa é a fase mais longa do ciclo de vida. O sistema é instalado e colocado em uso. A manutenção envolve a correção de erros que não foram descobertos em estágios iniciais do ciclo de vida, com melhora da implementação das unidades do sistema e ampliação de seus serviços em resposta às descobertas de novos requisitos.

Atualmente, o *Figma* não disponibiliza artigos específicos sobre manutenção de código. No entanto, oferece recursos que facilitam a colaboração entre *designers* e desenvolvedores, auxiliando na manutenção e implementação do código. Embora não haja artigos específicos sobre manutenção de código, esses recursos do *Figma* são fundamentais para facilitar a colaboração entre *design* e desenvolvimento, contribuindo para a manutenção eficaz do código. Dois desses recursos são o *Dev Mode* e o *Code Connect*.

1. *Dev Mode*

‘ O *Dev Mode* é uma interface no *Figma* concebida para aprimorar a manutenção do código, proporcionando aos desenvolvedores acesso direto a trechos de código a partir do *design.* Essa funcionalidade assegura maior alinhamento entre a interface projetada e sua implementação, minimizando retrabalho e simplificando futuras adaptações. Com essa abordagem, o código mantém-se mais estruturado, coerente e sustentável ao longo do tempo.

2. *Code Connect*

O *Code Connect* estabelece uma ligação entre a base de código dos componentes e o *Figma*, permitindo a incorporação direta do código do sistema de design no *Dev Mode*. Dessa forma, os desenvolvedores podem visualizar exemplos que reproduzem com fidelidade a estrutura do código em produção, facilitando a manutenção e evolução do projeto. Esse mecanismo reforça a precisão e a uniformidade na implementação, tornando o código mais organizado e sustentável. Informações complementares podem ser encontradas no artigo "*Code Connect*".

- ❖ Agradecimento ao Prof. Elias do Nascimento Melo Filho por prestar ajuda às alunas durante o desenvolvimento desse trabalho. Obrigada pela oportunidade!

Lista de Referências

- Quanto aos links informativos sobre o modo de operação do Figma:

1) O código reaproveitado possui as seguintes políticas:

I. https://help.figma.com/hc/pt-br/articles/15023202277399-Usar-trechos-de-c%C3%B3digo-no-Dev-Mode

II. https://help.figma.com/hc/en-us/articles/15023124644247-Guide-to-Dev-Mode

III. https://help.figma.com/hc/pt-br/articles/23920389749655-Code-Connect

2) Instruções relativas a teste de código:

I. https://help.figma.com/hc/en-us/articles/19790203466263-Test-your-prototypes-with-UserTesting

II. https://www.figma.com/resource-library/how-to-design-an-app/#run-tests-and-implement-feedback

3) Instruções relativas a manutenção de código Figma:

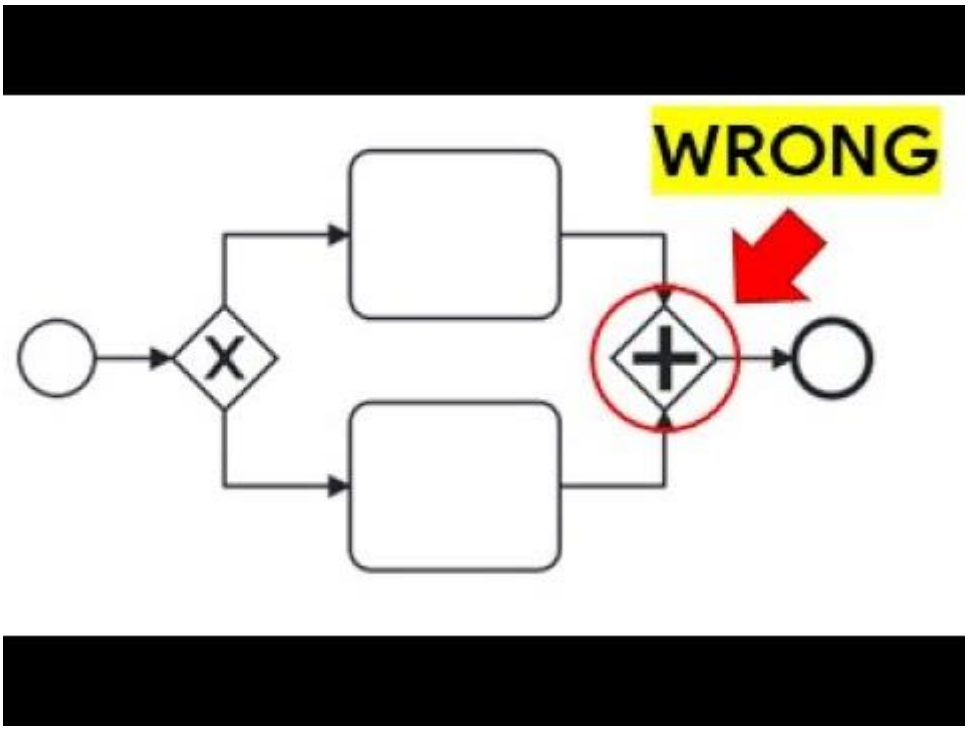
I. https://help.figma.com/hc/pt-br/articles/360043196274-O-que-acontece-durante-a-manuten%C3%A7%C3%A3o-programada

II. https://help.figma.com/hc/pt-br/articles/26781702258583-Status-e-notifica%C3%A7%C3%B5es-do-Dev-Mode

III. https://help.figma.com/hc/en-us/articles/19790203466263-Test-your-prototypes-with-UserTesting

IV. https://youtube.com/playlist?list=PLHz_AreHm4dlKP6QQCekuIPky1CiwmdI6&si=wKKrVtnrhCY_T0jw

- Para fins de obter uma visão geral sobre a estrutura do trabalho:



- Para fins de aprender a sincronizar a plataforma Figma com a VSCode:

Também os vídeos:

https://www.youtube.com/watch?v=Mp0vhMDI7fA&list=PLHz_AreHm4dlKP6QQCekuIPky1CiwmdI6&index=2

https://www.youtube.com/watch?v=VuKvR1J2LQE&list=PLHz_AreHm4dlKP6QQCekuIPky1CiwmdI6&index=3

https://www.youtube.com/watch?v=31llNGKWDdo&list=PLHz_AreHm4dlKP6QQCekuIPky1CiwmdI6&index=4

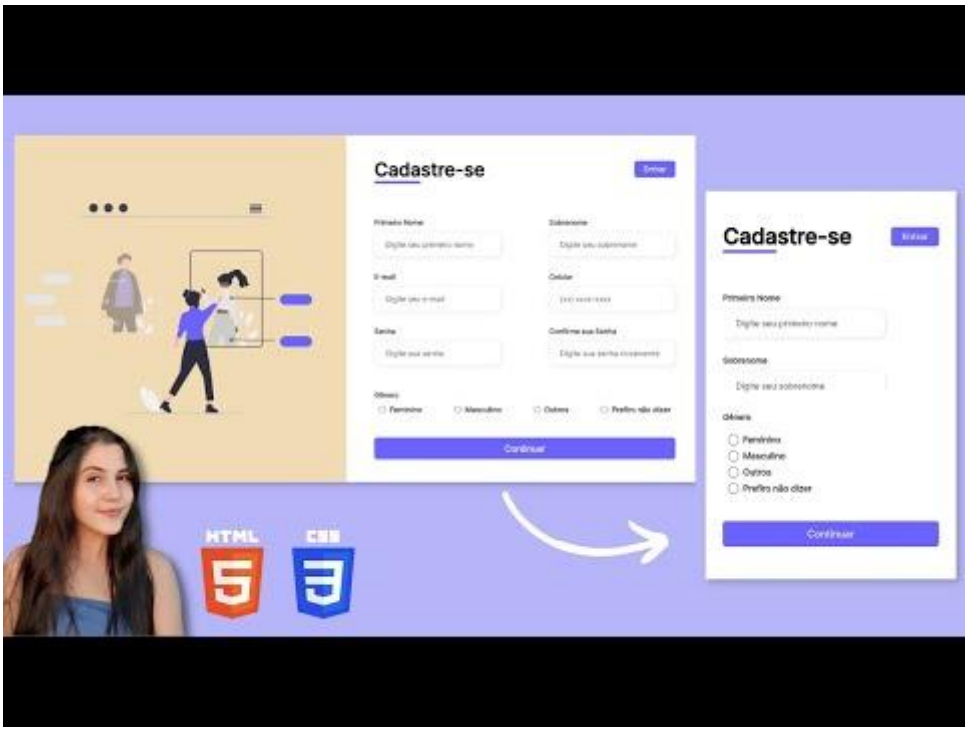
https://www.youtube.com/watch?v=ElRd0cbXIv4&list=PLHz_AreHm4dlKP6QQCekuIPky1CiwmdI6&index=6

https://www.youtube.com/watch?v=nIHq1MtJaKs&list=PLHz_AreHm4dlKP6QQCekuIPky1CiwmdI6&index=7

https://www.youtube.com/watch?v=FNqdV5Zb_5Q&list=PLHz_AreHm4dlKP6QQCekuIPky1CiwmdI6&index=8

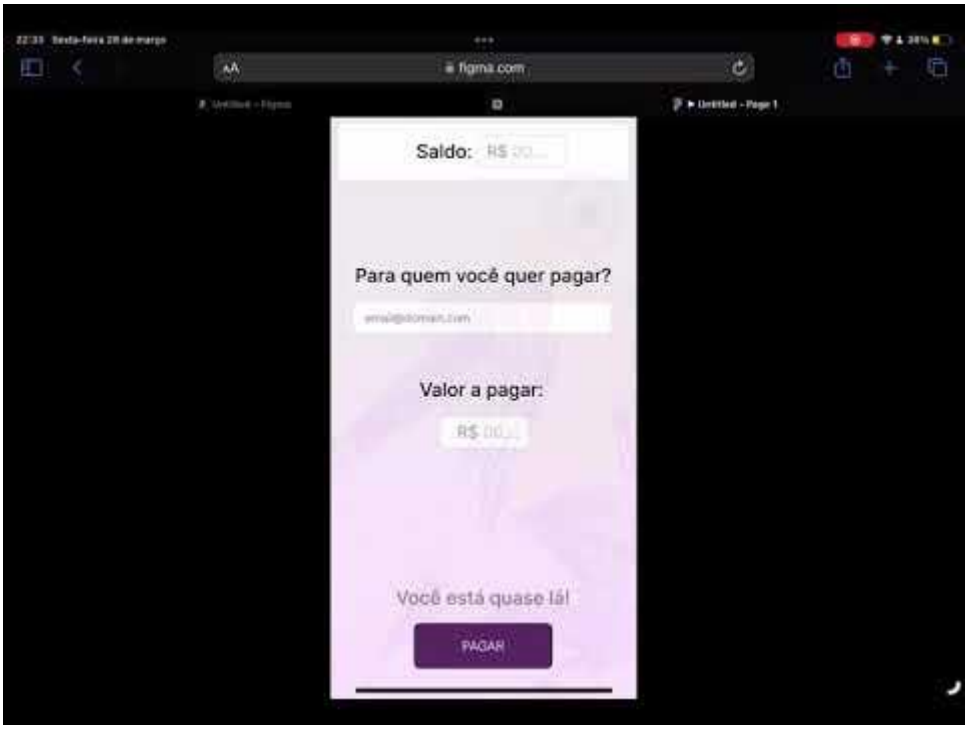
- Para fins de aprender a sincronizar a plataforma Figma com a VSCode:

- Para fins de aprender a como codificar e criar um Formulário de Cadastro funcional:



- Para fins de aprendizagem da linguagem HTML:

- URL para vídeo de demonstração do protótipo do *app*:



- Link para o Figma (o professor está na lista de acesso desse protótipo): https://www.figma.com/design/ZjyEGzmwj9GqhkgVuarTsK/Untitled?node-id=0-1&p=f&t=0Xn70l83RjJJh4mc-0