Projeto Interdisciplinar II - B

Tema: Sistema de Conta Digital.

4k Bank$

Integrantes do Projeto

- Rafael Santana Rgm - 31907288
- Lucas Alexandre Rgm - 31055435
- Victor Arouca Rgm - 31281893
- Ryan Godoy Rgm - 31158307
- Hendrik Mariano Rgm - 131175163

Indice:

Cronograma:

1. Estimativa de entrega.
1. Iniciou em 03/03/23 para entrega 15/05/23

Fase – 01 – Dialogo e ideias iniciais

1. Formação da equipe.
2. Pequenas reuniões sobre o tema.
3. Ferramentas de desenvolvimento a serem usadas.

Fase – 02 - Definição de objetivos de escopo.

1. Identificação dos requisitos e necessidades.
2. Estabelecimento dos objetivos do projeto.
3. Definição do escopo.

Fase – 03 – Pesquisa e levantamento.

1. Realização de pesquisas e levantamento de informações.

Fase – 04 – Análise e planejamento.

1. Analise das informações coletadas
2. Elaboração do plano detalhado.

Fase – 05 – Desenvolvimento do Projeto.

1. Início do desenvolvimento das etapas de acordo com o plano.
2. Acompanhamento do progresso e controle de prazos.

Fase – 06 – Revisão e Ajustes.

1. Avaliação das atividades geral concluídas.
2. Identificação de Problemas.
3. Realização de ajustes necessários envolvendo toda infra estrutura do Projeto.

Fase – 07 – Finalização.

1. Conclusão das atividades restantes.
2. Revisão final do projeto realizado.
3. Preparação para entrega.

Especificações do Escopo

Motivação:

- A motivação é oferecer aos clientes um serviço bancário digital moderno e eficiente, que possa ser acessado facilmente a qualquer hora e lugar, através de dispositivos móveis ou computadores. Além disso, a demanda por serviços bancários mais acessíveis e com menor custo, e um sistema bancário digital que pode atender a todas as classes sociais.

Objetivo do Projeto:

- O objetivo desse projeto é desenvolver um sistema bancário digital que permite aos usuários realizar operações financeiras com facilidade e rapidez, tais como consultas de saldo, transferências, pagamentos, realizar transações via PIX e depósitos.
- Garantir a segurança das transações financeiras, protegendo os dados dos usuários contra possíveis ameaças e ataques cibernéticos.
- Proporcionar uma experiência de usuário amigável e intuitiva, de forma a atrair e fidelizar clientes.
- Oferecer uma plataforma escalável, capaz de suportar um grande número de usuários e transações simultâneas, sem comprometer o desempenho e a estabilidade do sistema.
- Cumprir as regulamentações e exigências do mercado financeiro, garantindo a conformidade legal e regulatória do sistema bancário digital.

Objetivos do Negócio:

- O objetivo principal deste negócio é fornecer uma solução bancária digital moderna e inovadora que atenda às necessidades dos usuários e empresas de todos os perfis e segmentos, em todo o território nacional. Além disso, busca-se oferecer um serviço diferenciado e de qualidade, que seja referência no mercado financeiro, contribuindo para a inclusão financeira e a democratização do acesso aos serviços bancários.
- Com esse sistema, busca-se oferecer um serviço diferenciado e confiável, que permita a realização de transações financeiras com facilidade e praticidade, garantindo a segurança e a privacidade das informações dos usuários.

Riscos do Projeto:

- Risco de segurança: como se trata de um sistema bancário digital, a segurança é uns dos principais fatores a serem considerados. Qualquer falha de segurança pode levar a perda de informações confidenciais dos clientes e comprometer a reputação do banco.
- Risco de Implementação: o desenvolvimento de um sistema bancário digital complexo pode apresentar desafios técnicos que podem atrasar a entrega do projeto. Além disso, mudanças nos requisitos ou problemas com os fornecedores de tecnologia podem atrasar ainda mais a implementação do projeto.
- Risco Regulatório: o setor financeiro é altamente regulamentado e qualquer mudança nas leis e regulamentos pode afetar o projeto. É importante monitorar as mudanças regulatórias e se adaptar a elas.
- Risco de Mercado: o mercado financeiro é altamente competitivo e o sucesso do projeto dependerá da aceitação dos usuários. Se o mercado não adotar o sistema bancário digital, o projeto pode fracassar.

Especificação do Sistema proposto:

Objetivo.

- O objetivo desse projeto, é proporcionar uma melhor experiência e adaptação para o usuário. Dentre os objetivos deste documento estão o de listar da forma mais clara possível, os requisitos funcionais e não funcionais do nosso sistema. O usuário cliente tomara conhecimento do que o sistema fara e os desenvolvedores de como poderão implementar as funcionalidades

Escopo.

- O presente documento visa de forma abrangente e clara para que possa ser utilizado por todos os desenvolvedores

Visão Geral.

- "O objetivo do sistema é proporcionar uma experiência completa para o usuário, com diversas funcionalidades disponíveis e com suporte de 24 horas por dia".

Descrição Geral:

Atributos de qualidade.

- A aplicação será completamente voltada a velocidade para melhor desempenho do usuário na mesma. Será realizado diversas verificações para ter ciência sobre, se o aplicativo irá precisar de manutenção, e também a pedido dos usuários colher algumas informações para que possamos realizar alguns ajustes, e porventura processo de reajuste do aplicativo.

Obtenção e Especificação de requisitos:

Requisitos Funcionais

1. O usuário deve fornecer seu nome de CPF e senha corretos para acessar sua conta
2. O sistema deve permitir que os clientes consultem seus saldos de conta bancária.
3. O sistema deve permitir que os usuários façam saques de suas contas bancárias.
4. O sistema deve permitir que os usuários depositem dinheiro em suas contas bancárias.
5. O sistema deve permitir que os clientes façam transferências bancárias entre contas do mesmo banco.
6. O sistema deve permitir que os clientes realizem pagamentos de contas de serviços públicos e de outras empresas
7. O sistema deve permitir que os clientes realizem transações financeiras usando o sistema Pix.
8. O sistema deve permitir que os clientes gerenciam suas informações pessoais e de conta bancária
9. O sistema deve permitir que os funcionários do banco gerenciem as contas dos clientes, aprovem solicitações de crédito e gerem relatórios financeiros.

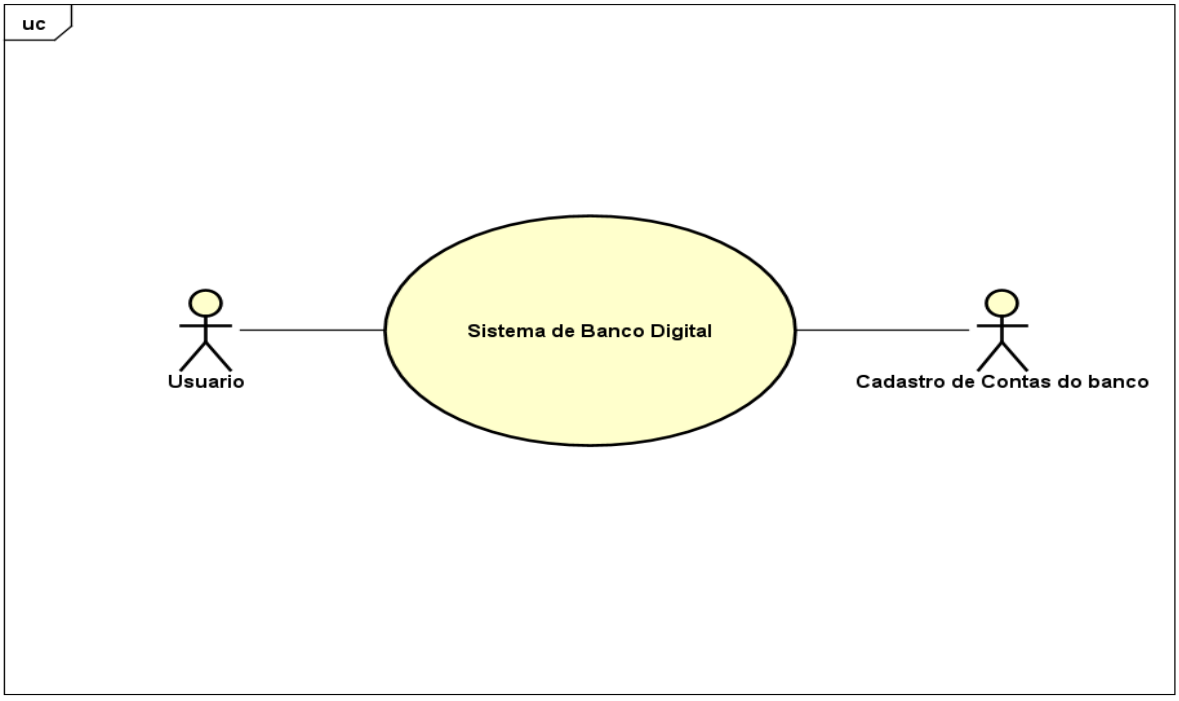
Requisitos não Funcionais

1. O sistema deve ser capaz de lidar com uma grande quantidade de usuários simultâneos sem ficar indisponível
2. O sistema deve ser capaz de lidar com um grande número de transações simultâneas.
3. O sistema deve ter tempos de resposta rápidos para evitar atrasos nas transações
4. O sistema deve ser seguro e proteger as informações financeiras dos
5. O sistema deve cumprir as regulamentações governamentais e as políticas internas do banco.
6. O sistema deve ter alta disponibilidade e estar disponível em todos os momentos.
7. O sistema deve ter uma interface de usuário amigável e fácil de usar.
8. O sistema deve ser escalável e permitir a adição de novos recursos e funcionalidades no futuro.
9. O sistema deve ter backup e recuperação de desastres em caso de falha do sistema.

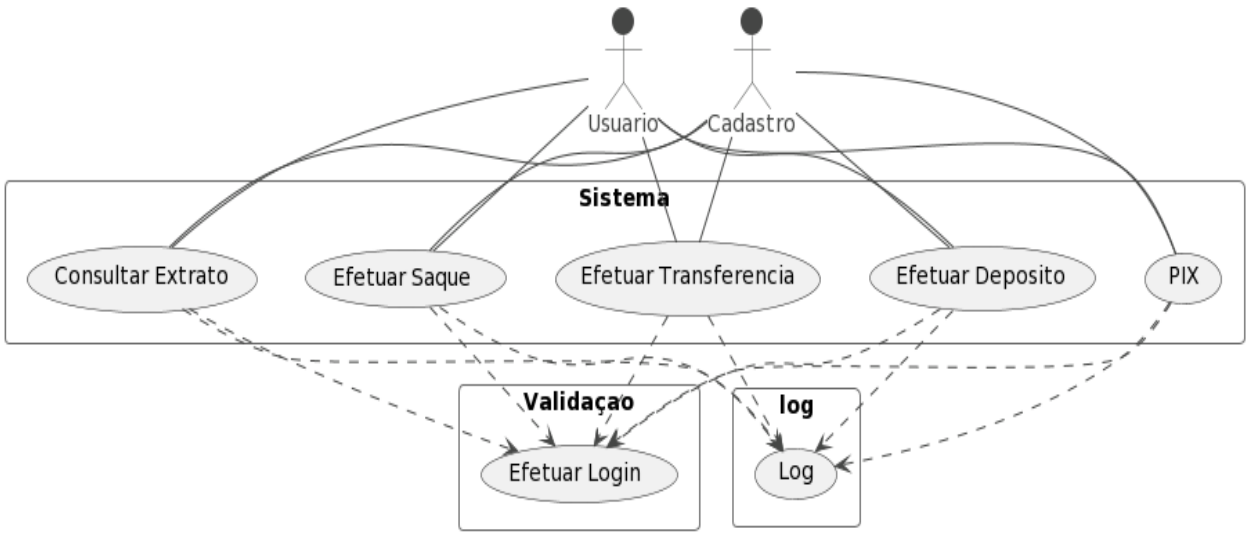
Regras de Negócio:

- Os usuários devem ter uma conta bancária ativa para usar o sistema.
- As transações financeiras realizadas pelo sistema devem ser legais e cumprir as regulamentações governamentais.
- As informações financeiras dos clientes devem ser protegidas e confidenciais.
- O banco deve realizar verificações de crédito antes de aprovar um empréstimo para um cliente.
- As taxas bancárias para cada tipo de transação devem ser claramente definidas e transparentes para os clientes.
- O banco deve notificar os clientes sobre qualquer alteração em suas contas bancárias ou transações financeiras.
- O sistema deve registrar todas as transações financeiras realizadas pelos clientes e fornece relatórios financeiros para os funcionários do banco.
- O sistema deve ter uma opção de suporte ao cliente para ajudar os clientes com suas

Modelo de Contexto (caso de uso nível 0)



Modelo de Casos de Uso:



Detalhamento:

Caso de Uso: Consultar Saldo

- Breve Descrição: O Usuário, já autenticado, escolhe a opção "Consultar Extrato" e o sistema apresenta seu saldo
- Atores: Usuário, Cadastro de Contas do Banco
- Pré- condição: A conta deve estar ativa e o cliente já deve ter sido autenticado junto ao sistema, através do caso de uso Efetuar Login

Caso de Uso: Consultar Saque

- Breve Descrição: O Usuário, já autenticado, escolhe a opção "Efetuar Saque", informa a quantia desejada e, caso o saldo da conta seja suficiente a quantia e liberada
- Atores: Usuário, Cadastro de Contas do Banco
- Pré-condição: O usuário deve estar logado no sistema, através do caso de uso "Efetuar Login". Além disso, a conta deve estar ativa e o valor a debitar deve ser maior que zero e não pode ser superior ao sado da conta nem a quantidade de dinheiro disponível no caixa.
- Pós-condição: O valor ser sacado e subtraído do saldo da conta e do total disponível no caixa eletrônico e a quantia solicitada e fornecida ao usuário.

Caso de Uso: Consultar Deposito(Transferencia)

- Breve Descrição: O usuário, já autenticado, escolhe a opção “Efetuar Deposito”, informa a quantia desejada e, a conta que deseja enviar o dinheiro
- Atores: Usuário, Cadastro de Contas do Banco.
- Pré-condição: O usuário deve estar logado no sistema, através do caso de uso "Efetuar Login".
- Pós-condição: O valor é depositado é adicionado ao saldo da conta.

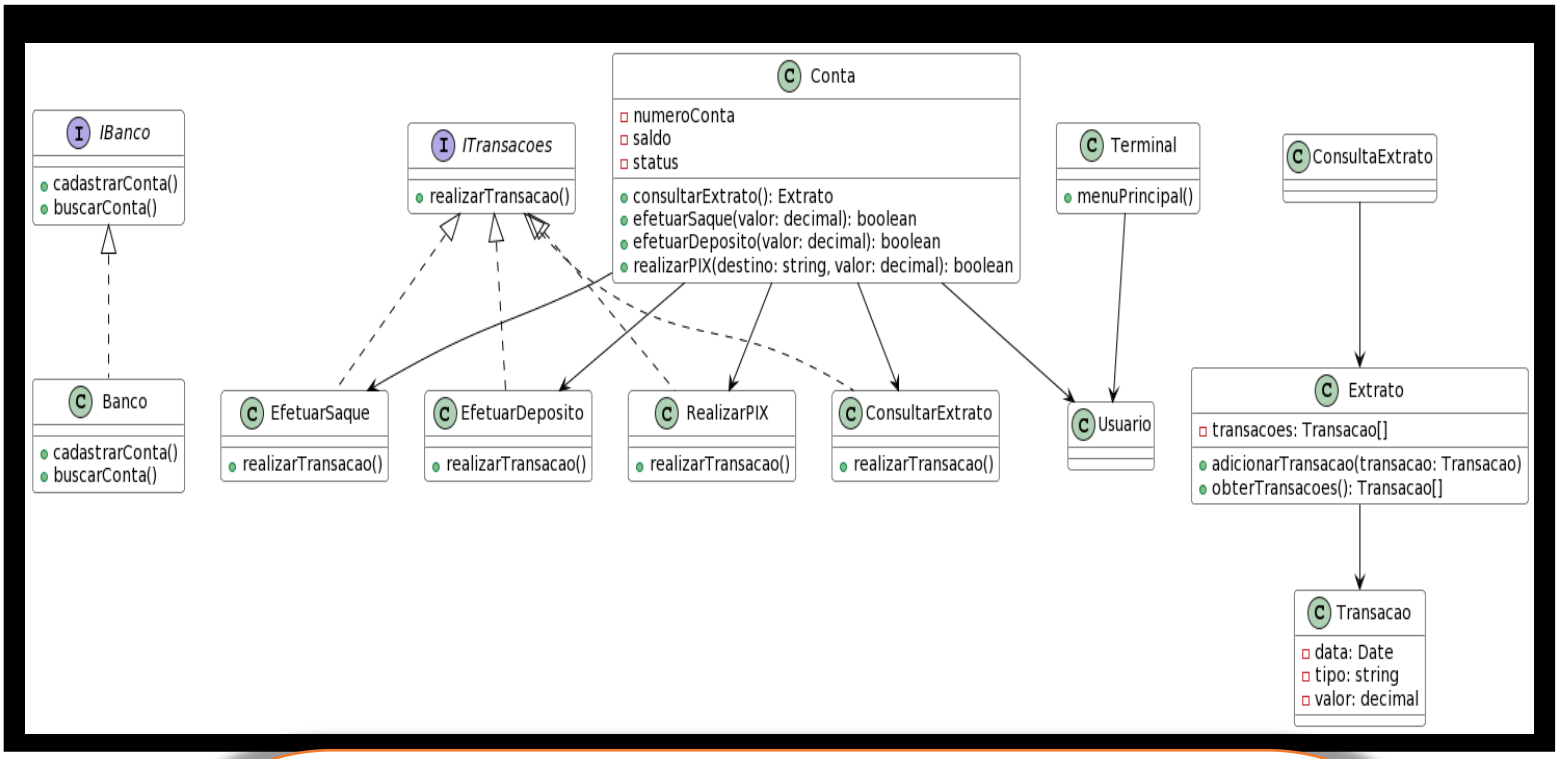
Caso de Uso: Consultar PIX

- Breve Descrição: O usuário, já autenticado, escolhe a opção "PIX" no menu principal para realizar uma transferência instantânea de valores para outro usuário, através dos dados fornecido (telefone, e-mail, CPF ou telefone)
- Atores: Usuário, Cadastro de Contas do Banco.
- Pré – condição: O usuário, já autenticado escolhe a opção “PIX”, informe qual a forma de transferir o valor ex: (fone ou CPF)
- Pós – condição: O valor e transferido instantaneamente para conta de destino.

Caso de Uso: Consultar Login

- Breve Descrição: O usuário deve fornecer o numero da conta e senha, essa informação deve ser autenticada pelo “Cadastro de Contas(usuário)”
- Atores: Usuário, Cadastro de Contas(usuário)
- Pré-condição: nenhuma
- Pós-condição: Após uma autenticação bem realizada, o usuário esta apto a operar o sistema de caixa eletrônico tanto pelo celular ou computador

Diagrama de Classes:



Interface IBanco: Essa interface define os métodos para cadastrar uma conta e buscar uma conta.

Interface Transações: Define o método realizar Transação, que será implementado pelas classes que lidam com diferentes tipos de transações.

Classe Banco: Implementa a interface IBanco e fornece a funcionalidade de cadastrar e buscar contas no banco.

Classe Conta: Representa uma conta bancaria e possui atributos como numero da conta, saldo e status. Ela tem métodos como consultarExtrato, efetuarSaque, efetuarDeposito e realizarPIX para realizar diferentes operações na conta. A classe Conta também possui uma associação com a classe Extrato, que contém as transações relacionadas a conta.

- *As respectivas "**classes**" que "**implementa**" a interface "**ITransaçoes**"*

1. ConsultarExtrato >Realizar a transação de consultar o extrato.
2. EfetuarSaque > Transação de sacar
3. EfetuarDeposito > Efetuar o deposito
4. RealizarPIX > Responsável por realizar as transações via PIX

Classe terminal: Representa o terminal do banco e possui um método menuPrincipal para exibir o me de opções disponível

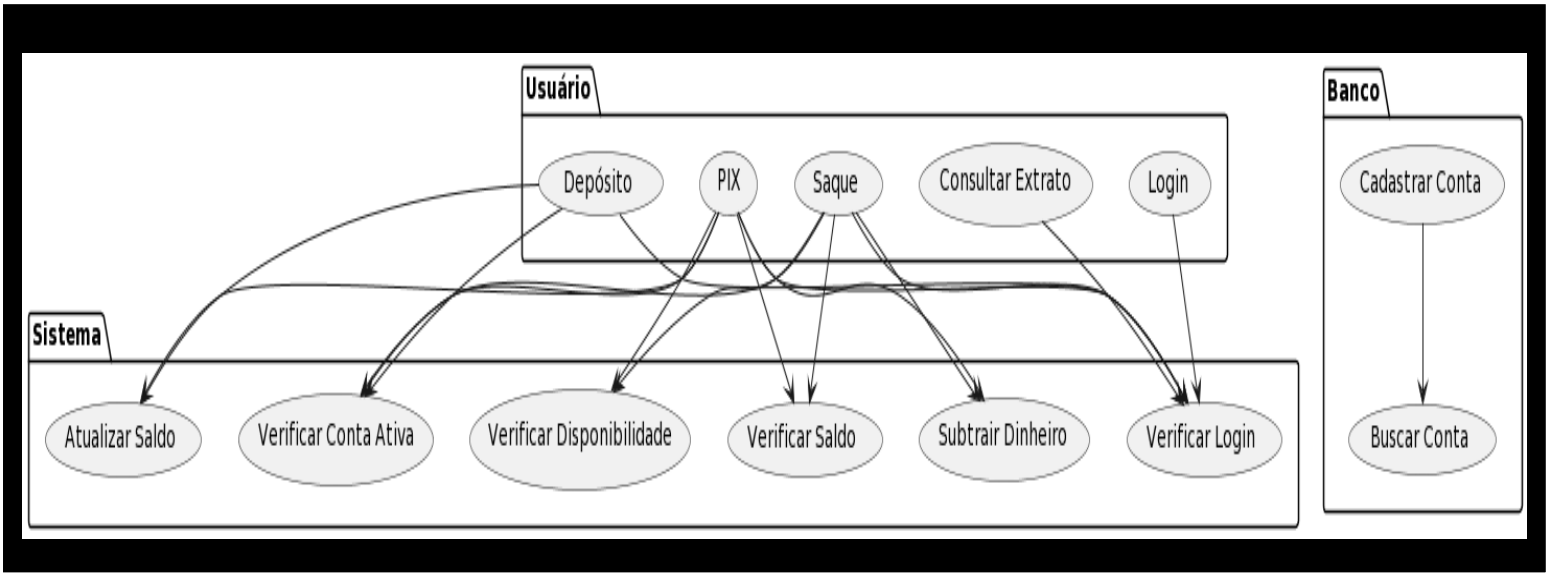
Classe Usuário: Representa um usuário do banco e é associada com classe Conta

Classe Extrato: Representa o extrato da conta, e registra todo tipo de transação.

Classe Transação: representa uma transação e contem informação como data, tipo e valor

As associações entre as classes estão representadas pelas setas no diagrama, indicando a relação entre as entidades

Diagrama de análise de fluxo de Trabalho:



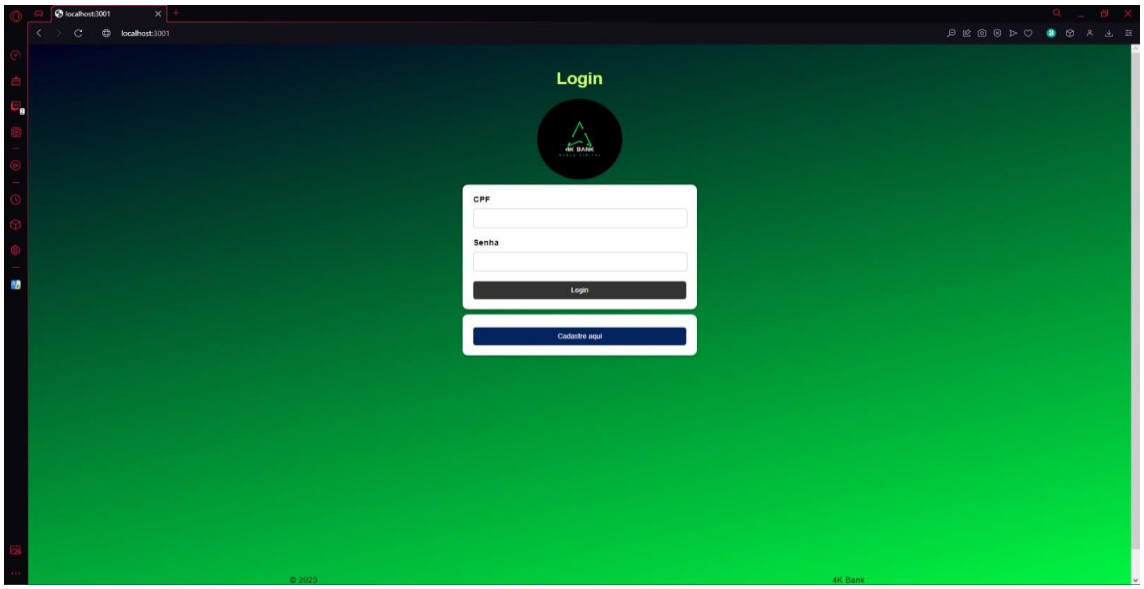
O diagrama acima representa o fluxo de trabalho de um sistema bancário, mostrando as intenções entre o usuário, o banco e o sistema. Ele ilustra as principais atividades, como login, consulta de extrato, saque, deposito e PIX, além de verificações realizadas pelo sistema, como verificações de login, conta ativa, saldo e disponibilidade de dinheiro.

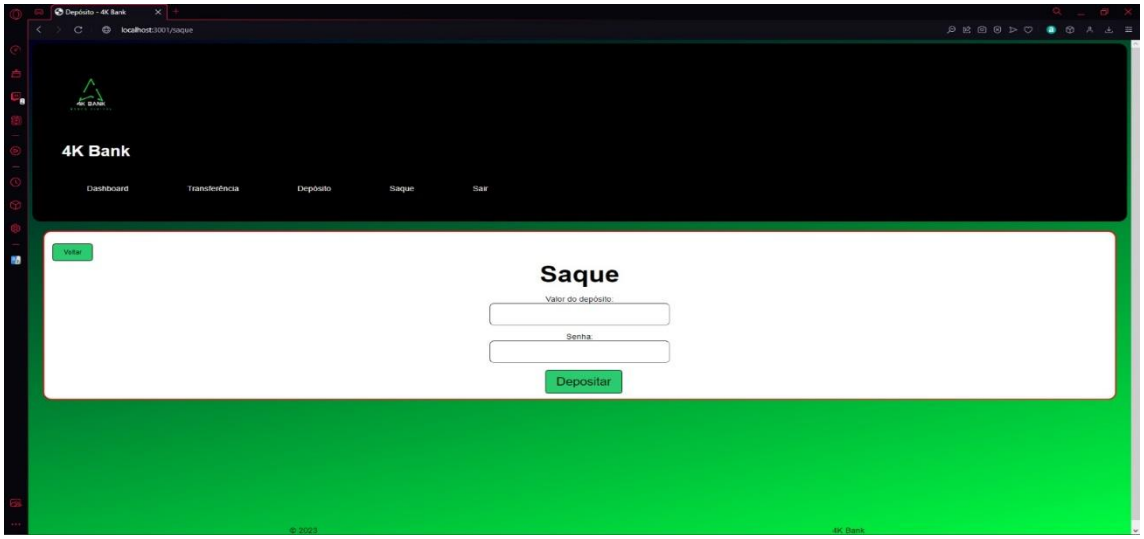
Esse diagrama ajuda a visualizar de forma simplificada e mais claras em etapas e interações envolvidas no processo bancário.

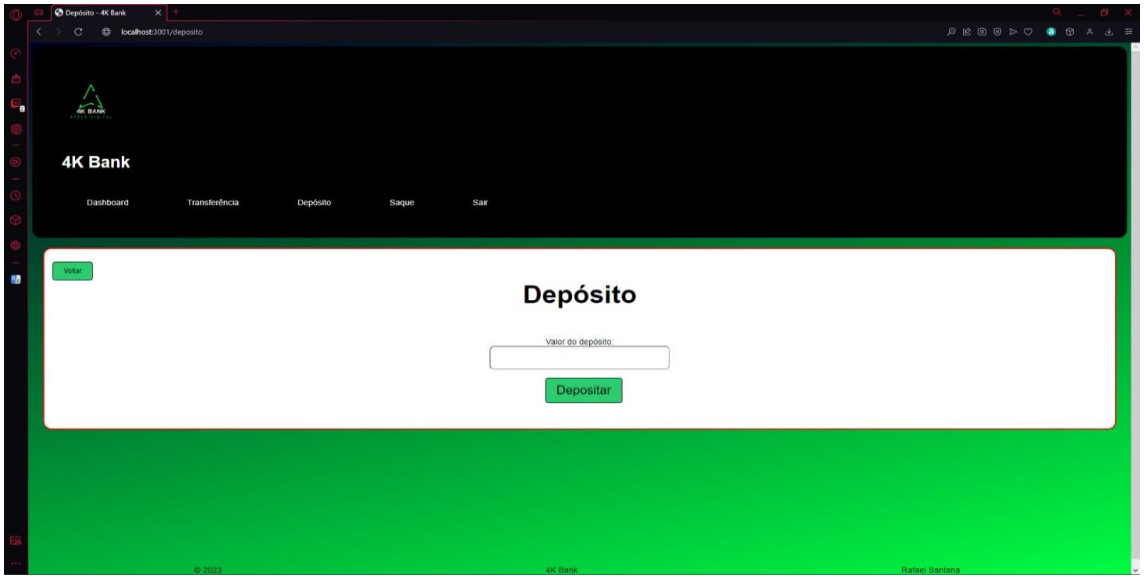
Ao final dos requisitos

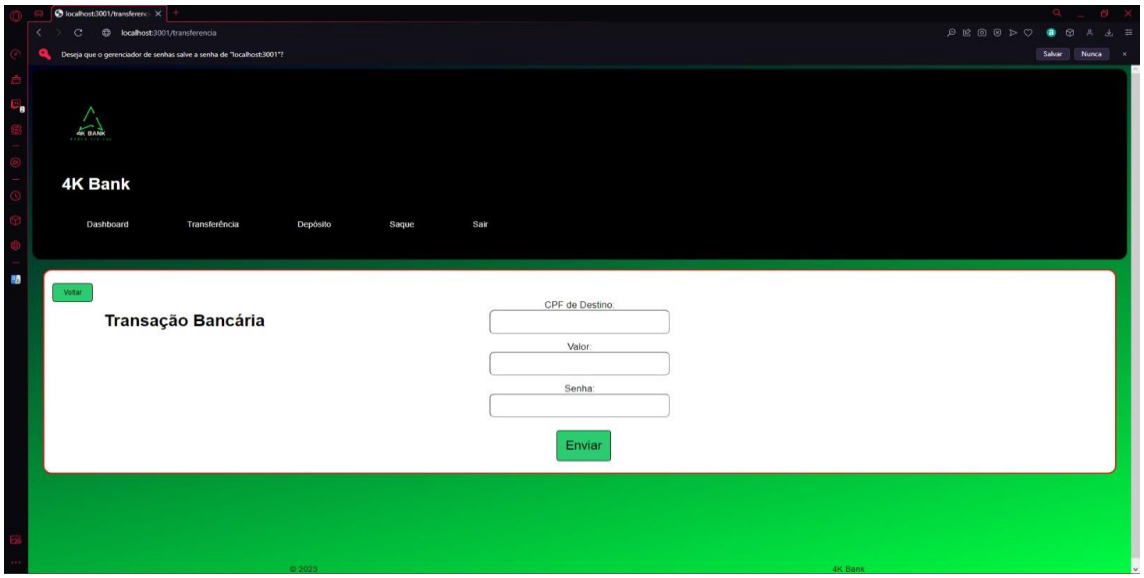
- *Realizar uma proposta*
- *Estimativa de custos*
- *Definir prioridade aos Requisitos levantados*
- *Analisar os riscos esperados*

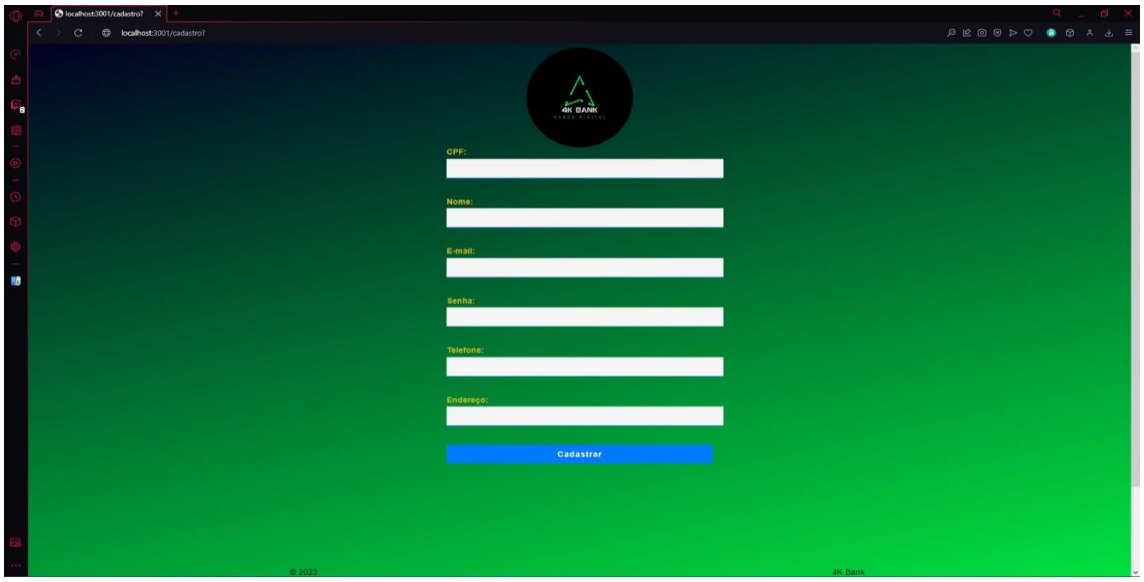
Mockups: Funcionamento do Programa, e os códigos de desenvolvimento.








localhost:3001/transferencia
Deseja que o gerenciador de senhas salve a senha de "localhost:3001"?
Salvar
Nunca
4K Bank
Dashboard
Transferência
Depósito
Saque
Sair
Voltar
Transação Bancária
CPF de Destino:
Valor:
Senha:
Enviar
© 2023
4K Bank


localhost:3001/cadastro?
CPF:
Nome:
E-mail:
Senha:
Telefone:
Endereço:
Cadastrar
© 2023
4K Bank

Rotas do funcionamento:

Rota: login.

```
// Inicio
app.get('/', (req, res) => {
  res.render('login')
})

// Rota de Login
app.post('/login', async (req, res) => {
  localStorage.clear()
  const cpf = req.body.cpf
  const senha = req.body.password

  const query = db.execute(`select u.cpf, u.senha, c.num_conta from usuarios u, contas c where cpf = ? and senha = ?`, [cpf, senha], (err, result) => {
    if (err) throw err
    console.log(result)
    if (!result.length == 0) {
      if (result[0].cpf == cpf) {
        if (result[0].senha == senha) {
          console.log('logado!')
          localStorage.setItem('cpf', cpf)
          res.redirect('/dashboard')
        }
      }
    } else {
      console.log("Credenciais inválidas!")
      res.redirect('/')
    }
  })
  console.log("cheguei aqui")
})

// Rota do dashboard
app.get('/dashboard', async (req, res) => {
  const conta = localStorage.getItem('cpf')
  var valorTela
  await db.execute('SELECT saldo from contas WHERE usuario_id = (SELECT id FROM usuarios WHERE cpf = ?)', [conta], (err, result) => {
    valorTela = result[0].saldo
    res.render('dashboard', { cssPath: '/public/CSS/dashboard.css', saldo: valorTela })
  })
})
```

Rota: cadastro.

```
// Rota de Cadastro
app.get('/cadastro', (req, res) => {
  res.render('cadastro')
})

// Rota de Registro
app.post('/registro', async (req, res) => {
  if (req.body.CPF.length == 11) {
    try {
      const params = [null, req.body.CPF, req.body.Nome, req.body.Email, req.body.Senha, req.body.Telefone, req.body.Endereco];
      if (params.some(param => param === undefined)) {
        throw new Error('Um ou mais parâmetros de ligação são undefined');
      }
      const result = await db.promise().execute('INSERT INTO usuarios(id,cpf, nome, email, senha, telefone, endereco) values(?,?,?,?,?,?,?)', params);
      const id_usuario = result[0].insertId;
      const num_conta = Math.floor(Math.random() * 900000) + 100000;
      const params2 = [id_usuario, 'Corrente', 0.0, num_conta];
      if (params2.some(param => param === undefined)) {
        throw new Error('Um ou mais parâmetros de ligação são undefined');
      }
      const resultado = await db.promise().execute("INSERT INTO contas(usuario_id,tipo,saldo,num_conta) VALUES (?,?,?,?)", params2);
      console.log(resultado);
      res.redirect('/');
    } catch (err) {
      res.redirect('/cadastro');
    }
  } else {
    res.redirect('/cadastro');
  }
});
```

Rota: Logica de transferencias.

```
// Logica de Transferencia
app.post('/logTransferencia', async (req, res) => {
  console.log(req.body)
  const cpf = localStorage.getItem('cpf')
  const cpfDest = req.body.cpfDest;
  const valorT = req.body.valor;
  const data = moment().format('DD/MM/YYYY HH:mm:ss')
  console.log(data)
  let [rows, fileds] = await db.promise().query('select saldo from contas where usuario_id = (select id from usuarios where cpf = ?)', [cpf])
  let [row, field] = await db.promise().query('select saldo from contas where usuario_id = (select id from usuarios where cpf = ?)', [cpfDest])
  let valor1 = parseInt(rows[0].saldo) - parseInt(valorT);
  let valorDest = parseInt(row[0].saldo) + parseInt(valorT);
  await db.execute('update contas set saldo = ? where usuario_id = (select id from usuarios where cpf = ?)', [valor1, cpf])
  await db.execute('update contas set saldo = ? where usuario_id = (select id from usuarios where cpf = ?)', [valorDest, cpfDest])
  await db.execute('Insert into transferencia(conta_origem,conta_destino, valor, data)values(?,?,?,?)',[cpf,cpfDest,valorT,data])
  let [r,f] = await db.promise().query('select nome from usuarios where cpf = ?',[cpf])
  let [ro,fi] = await db.promise().query('select nome from usuarios where cpf = ?',[cpfDest])
```

Rota: Deposito.

```
// rota de deposito
app.get('/deposito', (req, res) => {
  res.render('deposito')
})

// Logica de Deposito
app.post('/logDeposito', async (req, res) => {
  const body = req.body.valor
  const cpf = localStorage.getItem('cpf')
  var [rows, fields] = await db.promise().query('select saldo from contas where usuario_id = (select id from usuarios where cpf = ?)', [cpf])
  const saldo = rows[0].saldo
  const valor = parseInt(saldo) + parseInt(body)
  await db.execute('update contas set saldo = ? where usuario_id = (select id from usuarios where cpf = ?)', [valor, cpf])
  res.redirect('/dashboard')
})
```

Rota: Saque.

```
//  Rota de Saque
app.get('/saque', (req, res) => {
  res.render('saque')
})

// Logica de Saque
app.post('/logSaque', async (req, res) => {
  const body = req.body.valor
  const cpf = localStorage.getItem('cpf')
  var [rows, fields] = await db.promise().query('select saldo from contas where usuario_id = (select id from usuarios where cpf = ?)', [cpf])
  const saldo = rows[0].saldo
  const valor = parseInt(saldo) - parseInt(body)
  await db.execute('update contas set saldo = ? where usuario_id = (select id from usuarios where cpf = ?)', [valor, cpf])
  res.redirect('/dashboard')
})
```

Rota: Transferencia

```
// rota de tranferencia
app.get('/transferencia', (req, res) => {
  res.render('transferencia')
})
```

Logica: Comprovante.

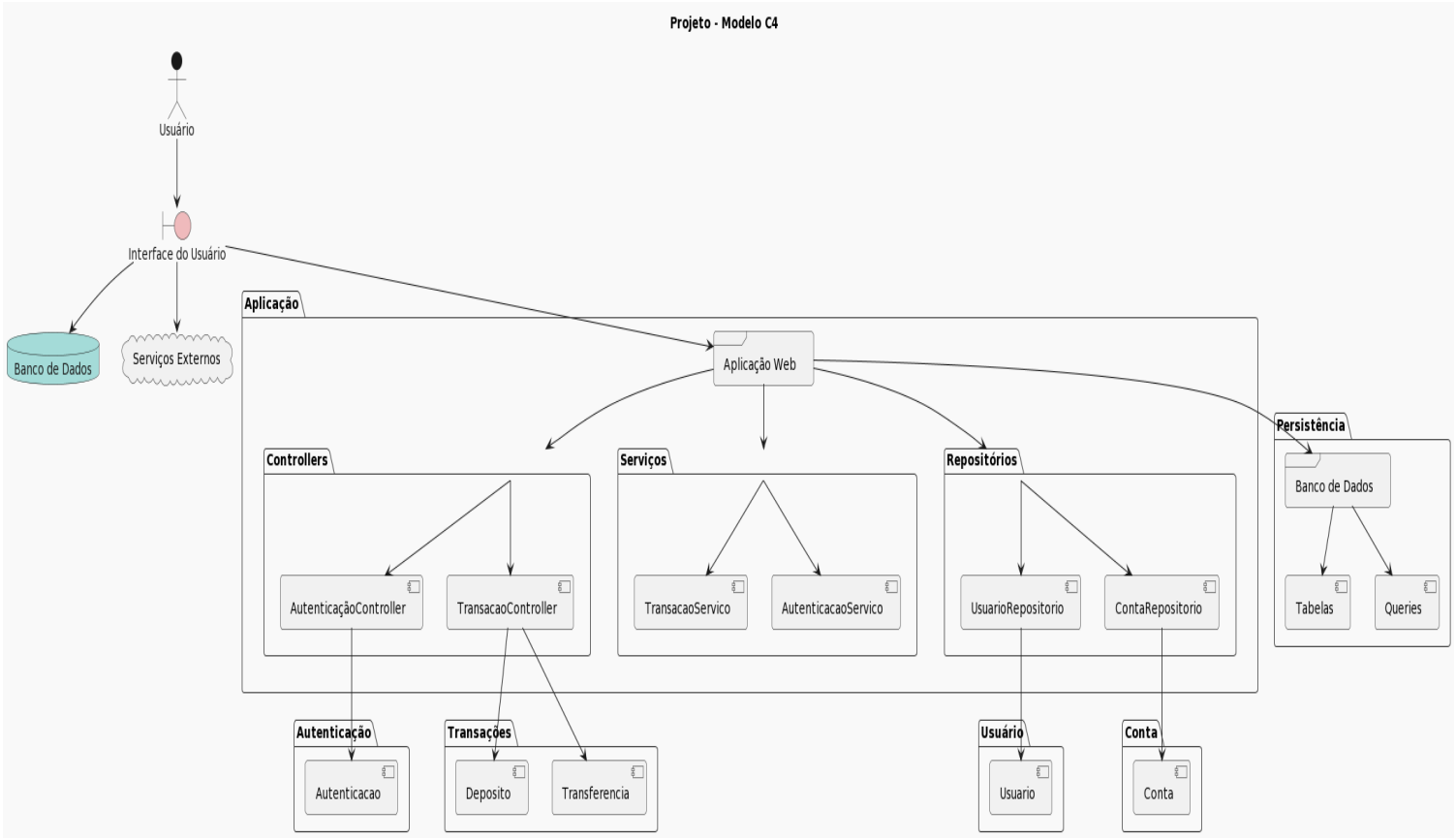
```
  try {
    // Criar o comprovante em PDF
    const doc = new PDFDocument();
    const writeStream = fs.createWriteStream('comprovante.pdf');

    doc.pipe(writeStream);
    doc.image('public/Imagens/4KBank.png', {
      width: 100,
      height: 100,
      x: doc.page.width / 2 - 50,
      y: 50,
      align: 'center'
    });
    doc.moveDown();
    doc.moveDown();
    doc.moveDown();
    doc.moveDown();
    doc.moveDown();
    doc.moveDown();
    doc.moveDown();
    doc.fontSize(17)

    doc.text('COMPROVANTE DE TRANSFERÊNCIA', { align: 'center' });
    doc.moveDown();
    doc.moveDown();
    doc.text(`CPF de origem: ${cpf}`, { indent: 20 });
    doc.text(`Nome: ${r[0].nome}`,{indent:20})
    doc.moveDown();
    doc.moveDown();
    doc.text(`CPF de destino: ${cpfDest}`, { indent: 20 });
    doc.text(`Nome: ${ro[0].nome}`,{indent:20})
    doc.moveDown();
    doc.moveDown();
    doc.text(`Valor transferido: R$ ${valorT}`, { indent: 20 });
    doc.text(`Data da Tranferencia: ${data}`, {indent: 20})
    doc.moveDown();

    doc.end();
    // Enviar o comprovante em PDF como resposta da requisição
    writeStream.on('finish', () => {
      res.setHeader('Content-Disposition', 'attachment; filename=comprovante.pdf');
      res.contentType('application/pdf');
      fs.createReadStream('comprovante.pdf').pipe(res);
    });
  } catch (error) {
    console.log(error);
  }
})
```

Modelo C4



Modelo C4: O modelo fornece uma estrutura para representar a arquitetura de software, promovendo a comunicação efetiva, clareza e compreensão, além de fornecer uma base consistente para documentação

Irei detalhar algumas de suas vantagens ao implantar o modelo C4:

1. Comunicação efetiva: O modelo C4 utiliza diagramas visuais simples e intuitivos, o que facilita a comunicação entre equipes técnicas e não técnicas. Ele fornece uma linguagem comum para discutir a arquitetura de software.
2. Clareza e compreensão: O modelo C4 permite uma representação clara da arquitetura, dividindo-a em contextos, contêineres, componentes e código. Isso ajuda a entender como os diferentes elementos se relacionam e interagem entre si.
3. Foco na estrutura: O modelo C4 concentra-se na estrutura estática e nas relações entre os elementos arquiteturais, permitindo uma visão abrangente da arquitetura sem se aprofundar nos detalhes de implementação.
4. Escalabilidade: O modelo C4 é escalável, o que significa que pode ser aplicado em diferentes níveis de granularidade, desde sistemas complexos até componentes individuais. Ele suporta uma representação hierárquica que facilita a decomposição e a compreensão dos sistemas.
5. Documentação consistente: O modelo C4 fornece uma estrutura consistente para documentar a arquitetura de software. Isso torna mais fácil criar documentação clara, organizada e padronizada, facilitando a manutenção e a colaboração.

Considerações Finais do Projeto

- Durante o desenvolvimento do projeto de um banco digital, enfrentamos desafios relacionados à criação dos diagramas, implementação do código, definição dos requisitos e documentação do software. No entanto, com trabalho em equipe e o uso de programas específicos, conseguimos superar esses obstáculos e alcançar nossos objetivos.
- Ao criar os diagramas, tivemos dificuldades em representar a estrutura e interações do sistema de forma clara e precisa. Foi necessário um bom entendimento das funcionalidades e dos relacionamentos entre as classes e componentes para criar diagramas coerentes e compreensíveis. Utilizamos ferramentas como o PlantUML e o Astah para auxiliar na criação dos diagramas e facilitar a comunicação entre a equipe.
- Na implementação do código do programa, enfrentamos desafios técnicos específicos da linguagem de programação escolhida, JavaScript. Lidar com conceitos como classes, herança, polimorfismo, tratamento de eventos e manipulação do DOM exigiu um conhecimento sólido da linguagem e boas práticas de programação. Além disso, a integração de bibliotecas e frameworks também trouxe suas próprias complexidades.
- Os requisitos do software desempenharam um papel crucial no projeto. Compreender as necessidades dos usuários, suas expectativas e os requisitos funcionais e não funcionais do sistema foi fundamental para o sucesso do projeto.
- A documentação do software foi uma etapa importante e demandou um esforço significativo. Criar documentação clara, concisa e abrangente, que incluísse requisitos, arquitetura, diagramas, fluxo de trabalho e instruções de uso, foi essencial para facilitar a compreensão do sistema e permitir sua manutenção e evolução futura. Utilizamos ferramentas de documentação para criar documentos de qualidade profissional.
- No trabalho em equipe, foi necessário um bom gerenciamento de tarefas, comunicação eficiente e colaboração entre os membros da equipe. A divisão de responsabilidades, definição de prazos e monitoramento do progresso foram aspectos essenciais para garantir a conclusão bem-sucedida do projeto. A troca de conhecimento, apoio mútuo e busca constante pela qualidade foram valores fundamentais para o trabalho em equipe.
- Destacamos o uso do Astah e do PlantUML para a criação dos diagramas, o Visual Studio Code como ambiente de desenvolvimento, e a utilização de bibliotecas e frameworks JavaScript relevantes para o projeto do banco digital.
- Em geral, o desenvolvimento do projeto do banco digital exigiu esforço, dedicação e trabalho em equipe, mas com perseverança e colaboração, conseguimos superar os desafios e entregar um software de qualidade.
- Esse projeto agregou muito, pois nos fez a se interessar profundamente mais na arquitetura de software, e descobrimos que podemos desenvolver uma documentação clara e eficiente,
- E se buscarmos novas fronteiras teremos uma visão de quanto é importante toda essa engenharia e arquitetura ao iniciar o desenvolvimento de um software.