PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS

ESCOLA DE CIÊNCIAS EXATAS E DA COMPUTAÇÃO

GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

SMART BANK: UMA PROPOSTA PARA AUTOMAÇÃO DE ANÁLISE DE CRÉDITO EM BANCO DIGITAIS

RONILSON DE SOUSA SILVA

GOIÂNIA - GO

Junho de 2021

RONILSON DE SOUSA SILVA

SMART BANK: UMA PROPOSTA PARA AUTOMAÇÃO DE ANÁLISE DE CRÉDITO EM BANCO DIGITAIS

Trabalho de Conclusão de Curso, apresentado à Escola de Ciências Exatas e da Computação, da Pontifícia Universidade Católica de Goiás, como parte dos requisitos para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Me. André Luiz Alves

GOIÂNIA - GO

Junho de 2021

RONILSON DE SOUSA SILVA

SMART BANK: UMA PROPOSTA PARA AUTOMAÇÃO DE ANÁLISE DE CRÉDITO EM BANCO DIGITAIS

Este Trabalho de conclusão de Curso julgado adequado para obtenção o título de Bacharel em Ciência da Computação, e aprovado em sua forma final pela Escola de Ciências Exatas e da Computação, da Pontifícia Universidade Católica de Goiás, em ____/____/_____.

_____

Prof. Ma. Ludmilla Reis Pinheiro

Coordenadora de Trabalho de Conclusão de Curso

Banca examinadora:

_____

Me. André Luiz Alves

_____

Me. Gilcimar Divino de Deus

_____

Me. Joriver Rodrigues Canêdo

Dedico este trabalho a minha mãe e meu pai, pelo exemplo de vida que deram a mim, com muita sinceridade e amor ensinaram-me conceitos fundamentais para a base de minha vida: fé, justiça própria e com o próximo, esperança, sinceridade, respeito, autovalores.

“Quero conhecer os pensamentos de Deus... O resto é detalhe."

Albert Einstein

RESUMO

Este trabalho apresenta a especificação, projeto e implementação de um produto de software, capaz de fazer análise cadastral de pessoas físicas para obtenção de crédito, tendo como exemplo proposto um banco digital. São contemplados a metodologia usada no desenvolvimento, os *frameworks* adotados, estilo arquitetural e *design patterns* empregados. O produto de software resultante encontra-se disponível para download na *Play Store*.

Palavras-chave: *Domain Driven Design*. Estilo Arquitetural. Microsserviços.

ABSTRACT

This work presents, in a simple and objective way, the implementation of a software project, capable of making registration analysis of individuals to obtain credit, using a digital bank as an example. It includes the methodology used in the development, the adopted frameworks, architectural style, and design patterns used. The resulting software product is available for downloading at the Play Store.

Key Words: Domain Driven Design. Arquitetural Style. Microservices.

LISTA DE ILUSTRAÇÕES

Figura 1 - Cadastrar Cliente 12

Figura 2 - Autenticar Cliente 13

Figura 3 - Autenticação do Cliente 14

Figura 4 – Acesso para Alterar Senha de Cliente 15

Figura 5 - Informar CPF - Redefinição de Senha 16

Figura 6 - Mensagem de Redefinição de Senha 16

Figura 7 - Cadastrar Nova Senha 17

Figura 8 - Lista de Solicitações de Empréstimos 18

Figura 9 - Solicitar Novo Empréstimo 19

Figura 10 - Solicitação Reprovada 20

Figura 11 - Solicitação Aprovada 20

Figura 12 - Complementação de Dados Cadastrais 21

Figura 13 - Captura de Selfie 22

Figura 14: Bloco de código da classse DomainServices 27

Figura 15: Bloco de código da classse ClienteSolicitacaoService 27

Figura 16: Bloco de código da classse DomainServices 29

LISTA DE SIGLAS

CPF - Cadastro de Pessoa Física

CNH - Carteira Nacional de Habilitação

RG - Registro Geral

SERPRO - Serviço Federal de Processamento de Dados

DETRAN - Departamento Estadual de Trânsito

DDD - *Domain Driven Design*

VPS - *Virtual Private Server*

SSO - *Single Sign-On*

API - *Application Programming Interface*

REST - *Representational State Transfer*

SUMÁRIO

RESUMO 6
ABSTRACT 7
SUMÁRIO 10
1 INTRODUÇÃO 11
1.1 - O Smart Bank 11
1.2 Recursos 11
1.2.1 Cadastrar Usuário Cliente (Requisito 1) 11
1.2.2 Autenticar Cliente (Requisito 2) 13
1.2.3 Redefinir Senha de Cliente (Requisito 3) 15
1.2.4 Solicitar Serviço Financeiro (Requisito 4) 17
1.2.5 Validar Documentos (Requisito 5) 20
1.2.6 Liberação de Crédito (Requisito 6) 23
2 PROCESSO DE DESENVOLVIMENTO E DECISÕES ARQUITETURAIS 24
2.1 PROCESSO DE DESENVOLVIMENTO 24
2.2 ESPECIFICAÇÃO DE REQUISITOS 25
2.3 PROJETO DE SOFTWARE E DECISÕES ARQUITETURAIS 25
2.3.1 PADRÃO ARQUITETURAL 26
2.3.2 DESIGN PATTERNS 26
2.3.3 MODELAGEM DO DOMÍNIO 29
2.3.4 LINGUAGENS DE PROGRAMAÇÃO E FRAMEWORKS 29
2.3.5 IMPLANTAÇÃO 30
3 CONCLUSÃO 31
REFERÊNCIAS 32
APÊNDICE A – ESPECIFICAÇÃO DE REQUISITOS DO SOFTWARE 34
APÊNDICE B – DISTRIBUIÇÃO DOS SERVIÇOS 45
APÊNDICE C – CÓDIGO FONTE DO PROJETO NO GITHUB 47
APÊNDICE D – APLICATIVO NA PLAYSTORE 48

1 INTRODUÇÃO

A sociedade tem passado por constantes mudanças nas últimas décadas, a evolução das tecnologias, a introdução em massa dos microcomputadores, a invasão dos smartphones aliada a facilidade de acesso à internet e aos diversos meios de comunicação instantânea, faz parecer que o mundo se tornou menor.

Ao perceberem tal evolução, as empresas começaram a adaptar seus processos às novas realidades. O segmento de serviço financeiro se destaca como um dos que sofreram mudanças drásticas nos seus processos internos, as filas de agências e correspondentes bancários, foram substituídas pelo conforto de se usar os aplicativos bancários a qualquer hora e em qualquer lugar.

Os serviços bancários têm sido completamente reformulados transformando-se no todo ou em parte em bancos digitais. Ao conjunto de tecnologias neste contexto denomina-se ***smartbanking***. É nesse contexto que esse trabalho se insere.

Trata-se de uma proposta de um serviço bancário específico em um banco digital, um pouco diferente dos modelos atuais. Os serviços contemplados por essa proposta visam dar suporte à obtenção de um empréstimo a partir uma empresa de serviço financeiro, passando por todo processo de validação do cliente até a liberação do crédito, de maneira integral pelo sistema, sem a intervenção humana.

O tópico a seguir apresenta um *tour* pelo aplicativo.

1.1 - O Smart Bank

Smart Bank é o nome dado ao produto resultante da realização desse projeto. Tem como objetivo atender às especificações de requisitos contidas no Apêndice A. Vale destacar, o uso do nome Smart Bank é uma decisão pessoal do autor e não possui nenhuma relação com marca ou tecnologia existente.

Às facilidades oferecidas aos usuários denominamos “Recursos”.

1.2 Recursos

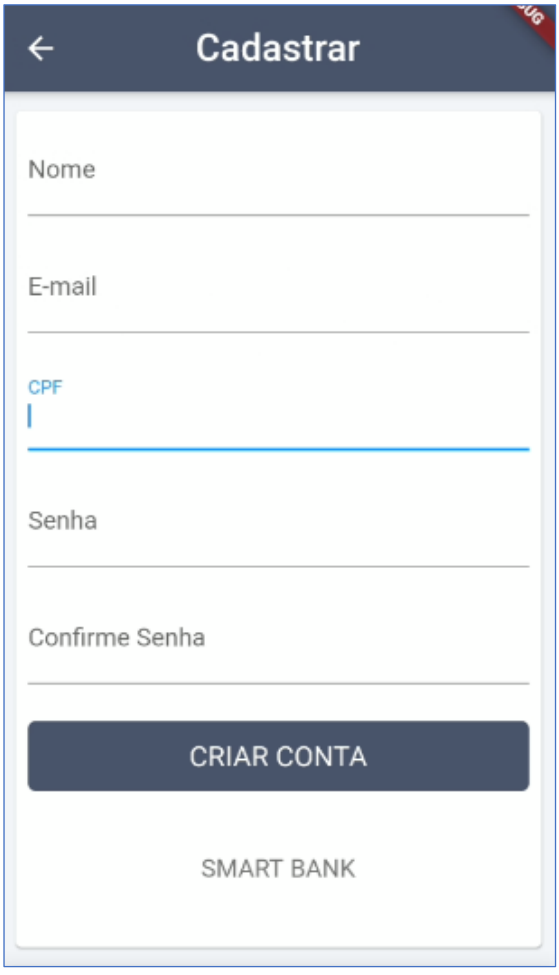
1.2.1 Cadastrar Usuário Cliente (Requisito 1)

Como o nome sugere, este recurso possibilita o cadastramento de usuário do tipo Cliente.

Para acessar esse formulário, na tela de login deve-se clicar na *label* “Criar Conta”. Atendendo à especificação dos requisitos são solicitadas as informações: nome, e-mail, telefone, cpf, rg ou cnh, endereço, data de nascimento, nome da mãe, e opcionalmente o nome do pai. No entanto, existe uma etapa obrigatória de atualização de dados, que é realizada após o cadastro do cliente, para facilitar o uso do aplicativo. Assim, os dados pessoais passam a serem exigidos apenas nessa etapa de atualização, exceto o CPF.

A Figura 1 apresenta a interface para Cadastrar Cliente.

Figura 1 - Cadastrar Cliente



Fonte: O autor

1.2.2 **Autenticar Cliente** (Requisito 2)

Essa é a primeira tela ao iniciar o aplicativo. O cliente deve fornecer seu identificador de usuário (CPF) e a senha informada ao se cadastrar no aplicativo. Se autenticação for realizada com sucesso, o usuário é redirecionado para tela inicial.

A Figura 2 apresenta a interface para Autenticar Cliente.

Figura 2 - Autenticar Cliente

Fonte: O autor

Para fins de facilidade de apresentação, alguns dados são exibidos:

- *Score*: dados fictícios, gerado de maneira aleatória pelo sistema;

- Biometria Facial: valor real, essa validação é feita junto a API Datavalid, disponibilizada pela SERPRO, esse campo indica o percentual/probabilidade de veracidade;
- CPF Disponível: apresenta valor real, essa validação é feita junto a API Datavalid, disponibilizada pela SERPRO, esse campo indica se o CPF foi encontrado na base da receita federal;
- CPF Regular: informação real, indica se possui alguma pendência no CPF do cliente;
- Nome Validado: Indica se o nome preenchido no cadastro está de acordo com registro na receita federal.

A Figura 3 apresenta a interface com o resultado da autenticação do Cliente.

Figura 3 - Autenticação do Cliente

Fonte: O autor

1.2.3 **Redefinir Senha de Cliente** (Requisito 3)

O aplicativo oferece também a opção de redefinição de senha. Para acesso à essa funcionalidade, na tela de autenticação, clicar em “Esqueceu Senha?”.

A Figura 4 apresenta a interface para acesso à redefinição da Senha do Cliente.

Figura 4 – Acesso para Alterar Senha de Cliente

Fonte: O autor

Será então redirecionado para um formulário, o qual requer que seja preenchido o CPF do usuário.

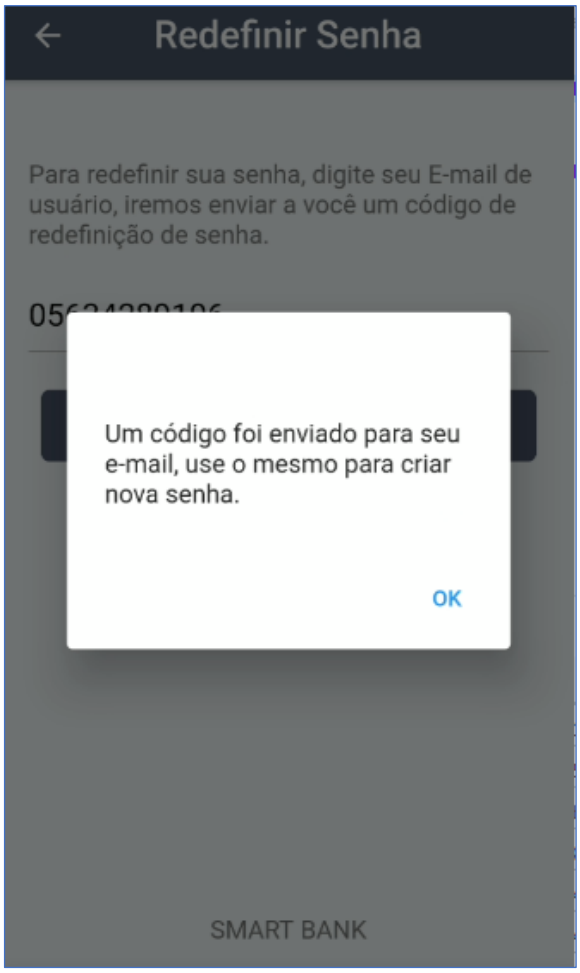
A Figura 5 apresenta a interface para informar o CPF.

A Figura 6 apresenta um *pop-up*, indicando que houve sucesso na solicitação do código de redefinição de senha.

Figura 5 - Informar CPF - Redefinição de Senha

Fonte: O autor

Figura 6 - Mensagem de Redefinição de Senha



Fonte: O autor

Um outro formulário será exibido para que o usuário possa informar o código que recebeu por e-mail e a nova senha.

A Figura 7 apresenta a interface para se cadastrar a nova senha.

Figura 7 - Cadastrar Nova Senha

Fonte: O autor

1.2.4 **Solicitar Serviço Financeiro** (Requisito 4)

Para solicitar serviço financeiro, deve-se acessar a aba “Empréstimo” disponível no rodapé da interface do aplicativo. Ao ser selecionado, é apresentada uma lista de todas as solicitações realizadas pelo cliente, bem como a opção de iniciar uma nova solicitação.

A Figura 8 apresenta a interface com a lista de solicitações de empréstimos.

Figura 8 - Lista de Solicitações de Empréstimos

Fonte: O autor

Ao acionar o botão nova solicitação [+], será apresentado um formulário, que deve ser preenchido com o valor desejado, a quantidade de parcelas a pagar e data de pagamento da primeira parcela.

A Figura 9 apresenta a interface para uma nova solicitação de empréstimo.

Figura 9 - Solicitar Novo Empréstimo

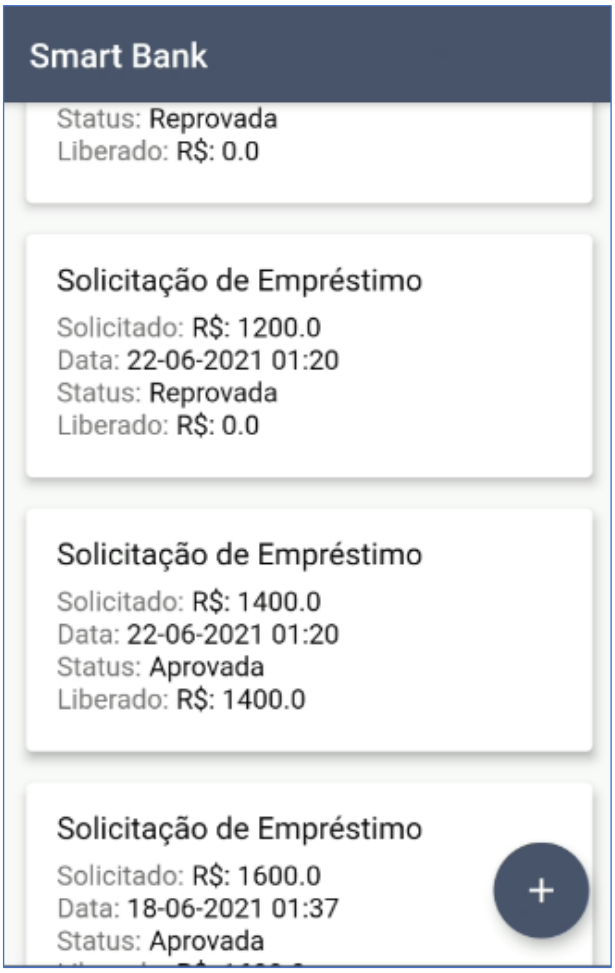
Fonte: O autor

Imediatamente após uma solicitação ser enviada pelo cliente, ela fica com status de iniciada, após realizar validação cadastral, score e renda, o novo status poderá ser:

- Reprovada: caso exista alguma pendência, *score* abaixo de 500 pontos, ou rende incompatível.
- Aprovada: nessa situação o cliente poderá aceitar ou cancelar a solicitação.

As Figuras 10 e 11 apresentam as situações de solicitação reprovada e solicitação aprovada respectivamente.

Figura 10 - Solicitação Reprovada



Fonte: O autor

Figura 11 - Solicitação Aprovada

Fonte: O autor

1.2.5 **Validar Documentos** (Requisito 5)

A validação de documentos tem por objetivo garantir que os dados informados no cadastro existem na base de dados da Receita Federal, com o propósito de se evitar que solicitações sejam realizadas por pessoas fictícias. Esse processo de validação é realizado por uma integração com o serviço Datavalid, disponibilizado pelo SERPRO. “O Datavalid consiste em uma solução inteligente para qualificação de pessoas físicas ou jurídicas”, (Datavalid API Docs, disponível em: https://apidocs.datavalidp.estaleiro.serpro.gov.br, acessado em 24 de junho de 2021). Para o projeto Smart Bank, os recursos do Datavalid foram implementados em duas etapas.

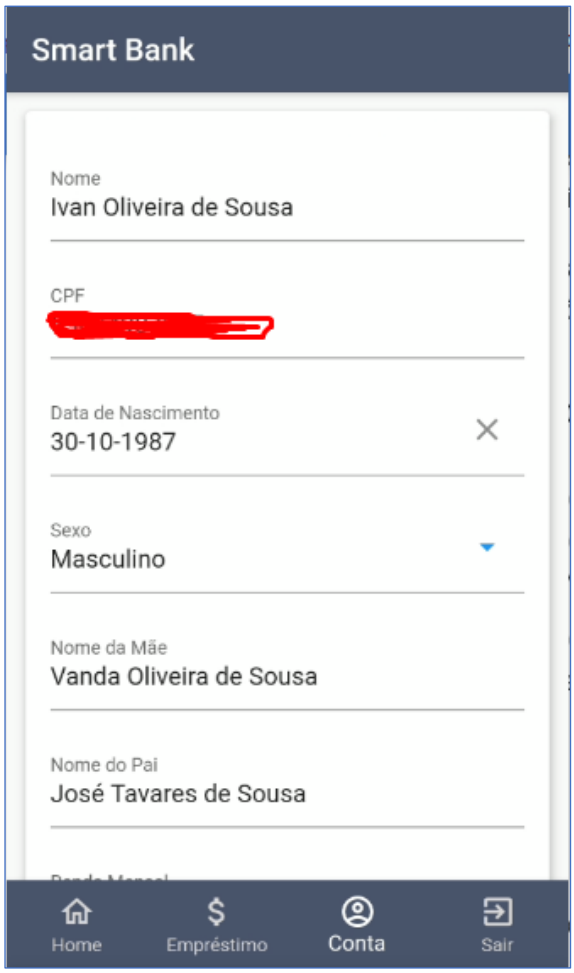
Etapa 1: Validação Cadastral

Ao finalizar seu cadastro no aplicativo, para que seja possível aprovação de solicitação de empréstimo o cliente precisa informar os seus dados pessoais requeridos na aba “Conta”.

Ao finalizar e enviar os dados, estes passam pela validação cadastral. Caso os dados não sejam validadados nessa etapa, as solicitações de empréstimo não serão aprovadas.

A Figura 12 apresenta a interface para completar as informações dos dados cadastrais requeridos.

Figura 12 - Complementação de Dados Cadastrais



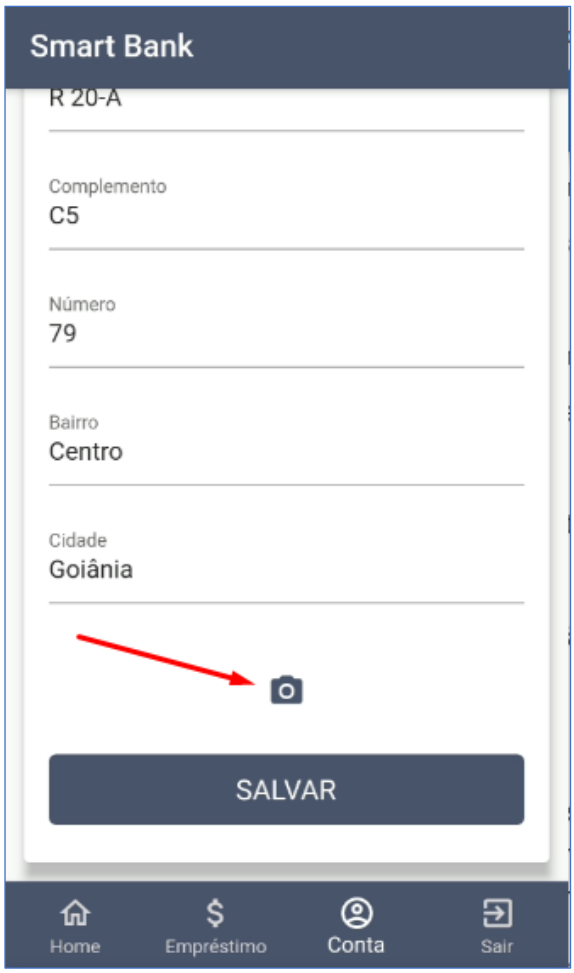
Fonte: O autor

Etapa 2: Validação Biométrica Facial

A segunda etapa de validação consiste em enviar uma *selfie* para validação facial, recurso incluído no formulário de complementação dos dados cadastrais.

A Figura 13 destaca o acesso à câmera digital do dispositivo para captura da *selfie*.

Figura 13 - Captura de Selfie



Fonte: O autor

Após enviar a selfie e a validação finalizada, o sistema pode retornar os seguintes status:

- Altíssima Probabilidade: validação aprovada;
- Alta Probabilidade: validação reprovada;
- Baixa Probabilidade: validação reprovada;

- Não Realizada: usuário ainda não enviou a *selfie* ou imagem não reconhecida.

Para que a validação seja concluída é necessário que o cliente tenha foto digital registrada em algum órgão de identificação, como o DETRAN, por exemplo.

1.2.6 Liberação de Crédito (Requisito 6)

Após os dados cadastrais e a biometria facial serem aprovados, será realizada a avaliação de crédito. Para esse recurso, seria ideal uma integração com algum dos serviços de proteção ao crédito, que já fornecem *score* para todos cidadão brasileiros. No entanto, essa integração não foi viável para ambiente de teste devido a exigências contratuais impostas pelos prestadores de serviços que disponibilizam tais informações. A fim de finalizar o processo de solicitação de crédito foi implementado um *score* gerado de maneira aleatória, com valores variando de 300 a 1000 pontos.

2 PROCESSO DE DESENVOLVIMENTO E DECISÕES ARQUITETURAIS

2.1 PROCESSO DE DESENVOLVIMENTO

O processo de desenvolvimento é mais uma decisão complexa, a ser tomada antes de iniciar um projeto de sistema, no entanto o movimento das metodologias ágeis tem se mostrado cada vez mais eficaz, e entregado muitos valores para as mais diversas organizações. O Manifesto Ágil, inicialmente lançado em fevereiro de 2001, serviu como base para o surgimento de inúmeras *frameworks*, facilitando de maneira significativa a implantação nas organizações.

Além dos princípios ágeis, existem diversos conceitos, *frameworks*, metodologias e ferramentas que se usados de maneira adequada, trazem muita produtividade durante todo o ciclo de vida do projeto. Entre os mais populares, temos:

- Scrum, um *framework* muito bem definido e amplamente usado nas organizações. Em sua documentação oficial, Scrum guides, é assim definido: “Scrum is a lightweight framework that helps people, teams and organizations generate value through adaptive solutions for complex problems” (SCRUM GUIDES, disponível em: https://scrumguides.org/scrum-guide.html, acessado em 24 de junho de 2021). Muito eficiente para trabalhos em equipes, tendo como filosofia um pequeno time de pessoas formado por *Product Owner*, *Scrum Master* e *Developers*. Segue o conceito de incremental, dividido em 5 eventos:
    - *Sprint Planning*, o planejamento do que deve ser adicionado na sprint, juntamente com a definição de como deve ser implementado.
    - *The Sprint*, a implementação do que foi planejado, ao final desse evento, deve ter um entregável ao projeto.
    - *Daily Scrum*, evento diário, com menor tempo possível. Seu principal propósito é avaliar o progresso da sprint, e discutir adaptações, caso seja necessário.
    - *Sprint Review*, apresentação e avaliação do que foi realizado na *sprint*.
    - *Sprint Retrospective*, tem por finalidade fazer uma análise da *sprint* finalizada e discutir pontos que precisam melhorar.

- Kanban, metodologia simples para gerenciar tarefas dentro de um projeto ou processo, em sua documentação oficial é definido: “Kanban is a strategy for optimizing the flow of value through a process that uses a visual, pull-based system” (KANBAN GUIDE, disponível em: https://kanbanguides.org/html-kanban-guide/#DefinitionOfKanban, acessado em 24 de junho de 2021). Diferente do Scrum que é um *framework*, o Kanban é apenas uma metodologia, podendo ser usado tanto por equipe ou apenas uma pessoa.

  Tendo em análise um excelente *framework* e uma simples metodologia de trabalho, considerando que a decisão deve ser tomada de acordo com a realidade de quantidade de pessoas disponíveis para o projeto, para o presente trabalho, por ser desenvolvido por apenas uma pessoa, foi definido o Kanban como uma metodologia mais adequada para gerenciamento de implementação do projeto.

2.2 ESPECIFICAÇÃO DE REQUISITOS

A etapa de levantamento de requisitos, é fundamental para o sucesso do projeto de software. A partir dela é possível tomar todas a decisões seguintes, a exemplo a arquitetura a ser definida para o sistema. Nesse projeto, foi usado o modelo de especificação de requisitos misto envolvendo o Standard IEEE 830 e o modelo Volere. Toda especificação de requisitos, incluindo os Casos de Uso se encontra disponível no Apêndice A.

2.3 PROJETO DE SOFTWARE E DECISÕES ARQUITETURAIS

A construção de um projeto de software, demanda muito esforço e disciplina, desde o início ao fim do projeto, no entanto, existe uma etapa, logo após a definição de escopo e levantado os requisitos principais, a qual pode determinar o sucesso ou fracasso do projeto, economizar ou aumentar o tempo e os custos de desenvolvimento e ainda pode dificultar ou facilitar o ciclo de manutenção. Essa etapa é comumente chamada de definição da arquitetura.

2.3.1 PADRÃO ARQUITETURAL

O estilo arquitetural é uma etapa muito complexa, e deve ser decidido com muita cautela, um erro pode comprometer tanto o processo de desenvolvimento quanto o produto final. Os estilos arquiteturais ou padrões arquiteturais, segue 2 principais conceitos:

- Monolítico, a qual todo produto é empacotado em um único projeto.
- Distribuído, a qual o produto final é dividido em diversos serviços, podendo até ser independentes entre si, é o caso do microsserviços.

Para esse projeto, foi adotado o estilo arquitetural distribuído em microsserviços. Alguns dos fatores considerados:

- Separação de responsabilidade, cada serviço corresponde a uma solução/recurso específico do projeto final;
- Maior facilidade em escalabilidade;
- Baixo acoplamento;
- Maior resistência a falhas.

O modelo de distribuição dos microsserviços e seu respectivo "*design*" arquitetural está disponível no Apêndice B desse documento.

2.3.2 DESIGN PATTERNS

Existem diversos dos chamados "padrões de projeto" na atualidade, sendo a maioria definida no livro *Design Patterns: Elements of Reusable Object-Oriented Software* (1994). Alguns são evoluções a partir de um padrão já existente. Dentre eles, ressalta-se aqui os implementado no projeto.

2.3.2.1 TEMPLATE METHOD

O *Template Method* é um dos padrões mais popular em um projeto de software, esse também faz parte da coleção apresentado no famoso livro de Erich Gamma, a qual definiu:

> Definir o esqueleto de um algoritmo em uma operação, postergando alguns passos para as subclasses. Template Method permite que subclasses redefinam certos passos de um algoritmo sem mudar a estrutura do mesmo. (GAMMA, 2000, p. 301)

Em outras palavras o *Template Method* permite a uma subclasse alterar ou expandir um determinado método da superclasse. Na linguagem de programação C#, para que seja possível a implementação desse *pattern*, a declaração do método na classe base deve ser precedido da palavra reservada *virtual*, enquanto na subclasse o método deve ser precedido da palavra *override*. como nas 2 imagens abaixo.

```
public virtual async Task<TEntity> Adicionar(TEntity entity)
{
    //validações
    if (entity.Invalid)...

    return await this._repository.Adicionar(entity);
}
```

Figura 14: Bloco de código da classse DomainServices, uma classe base e genérica, disponível no link: https://github.com/ronilsonsilva/smartbank/blob/main/src/services/SmartBank/SmartBank.Domain/Services/DomainServices.cs linha 21.

```
8 references | R S Silva, 5 days ago | 1 author, 4 changes
public async override Task<ClienteSolicitacao> Adicionar(ClienteSolicitacao solicitacao)
{
    await base.Adicionar(solicitacao);

    [Implementações Adicionais]

    return solicitacao;
}
```

Figura 15: Bloco de código da classse ClienteSolicitacaoService, método adicionar sobreescreve o método Adicionar da classe pai, na classe ClienteSolicitacaoService, disponível em: https://github.com/ronilsonsilva/smartbank/blob/main/src/services/SmartBank/SmartBank.Domain/Services/ClienteSolicitacaoService.cs linha 41.

2.3.2.2 MEDIATOR

O *Mediator* é um padrão que tem como principal objetivo, reduzir o acoplamento entre classes ou serviços da aplicação.

> Definir um objeto que encapsula a forma como um conjunto de objetos interage. O Mediator promove o acoplamento fraco ao evitar que os objetos se refiram uns aos outros explicitamente e permite variar suas interações independemente. (GAMMA, 2000, p. 257)

Nesse projeto foi definido uma classe SmartBank.Shared.Mediator.MediatorHandler, que implmenta a interface ImediatorHandler, é encontrada no url:

https://github.com/ronilsonsilva/smartbank/tree/main/src/services/SmartBank.Shared/Mediator.

2.3.2.3 REPOSITORY PATTERN

O *Repository Pattern* como definido na documentação oficial da Microsoft:

"O padrão Repositório é uma maneira bem documentada de trabalhar com uma fonte de dados", (disponível em https://docs.microsoft.com/en-us/dotnet/architecture/microservices/microservice-ddd-cqrs-patterns/infrastructure-persistence-layer-design, acesso em 24 de junho de 2021). A implementação da camada de repositório do projeto, está disponível em: https://github.com/ronilsonsilva/smartbank/tree/main/src/services/SmartBank/SmartBankUsers.Infra.Data.Repository/Repositories.

2.3.2.4 DOMAIN NOTIFICATIONS

Para Martin Fowler *notifications* pode assim ser definida: "A Notificação é um objeto que o domínio usa para coletar informações sobre erros durante a validação. Enquanto um erro aparece a notificação é enviada para camada de apresentação" (FOWLER, disponível em: https://martinfowler.com/eaaDev/Notification.html, acesso em 24 de junho de 2020).

Neste projeto foi especificamente implementada a classe SmartBank.Domain.Notifications.NotificationContext, responsável por lançar as notificações do domínio da aplicação. Esse arquivo está disponível no repositório do projeto, no link:

https://github.com/ronilsonsilva/smartbank/blob/main/src/services/SmartBank/SmartBank.Domain/Notifications/Notification.cs, na imagem abaixo a captura do trecho de código, de uma classe de serviço de domínio, a qual é realizada a validação.

```
public virtual async Task<TEntity> Adicionar(TEntity entity)
{
    //validações
    if (entity.Invalid)
    {
        this._notificationContext.AddNotifications(entity.ValidationResult);
        return entity;
    }

    return await this._repository.Adicionar(entity);
}
```

Figura 16: método Adicionar, que lança notificações de domínio se encontrar erros, disponível em: https://github.com/ronilsonsilva/smartbank/blob/main/src/services/SmartBank/SmartBank.Domain/Services/DomainServices.cs, linha 21.

2.3.3 MODELAGEM DO DOMÍNIO

Alguns dos serviços, foram modelados seguindo o padrão do DDD (*Domain Driven Design*), originalmente publicado por Eric Evans em 2003. A exemplo do serviço de envio de e-mails, não houve necessidade de uma modelagem de domínio complexa.

2.3.4 LINGUAGENS DE PROGRAMAÇÃO E FRAMEWORKS

Além dos desafios de decisões arquiteturais e modelagem de domínio, desenvolver um projeto de software exige outra grande decisão, que afeta diretamente todo o ciclo de vida do projeto.

O projeto como um todo é dividido em 2, o *back-end* e *front-end*, sendo desenvolvido com a seguintes linguagens e *frameworks*:

- *Flutter* para desenvolvimento do app mobile, framework para desenvolvimento *cross platforms*, desenvolvido pelo Google. Usa a linguagem dart.
- C# e ASP .NET Core 5.0, mantida pela Microsoft®, oferece excelentes recursos para desenvolvimento de web API.

2.3.5 IMPLANTAÇÃO

A Distribuição dos serviços encontra-se descrita no Apêndice C.

Ao todo, o projeto como possui dois produtos finais:

- *Backend* da aplicação, composto por diversos serviços independentes, os quais comunicam entre sim por meio de mensagerias. Essa parte será implantada em um VPS (Virtual Private Server).
- *Frontend*, aplicação mobile, disponível apenas para Sistema Operacional Android.

O aplicativo está disponível para download na Play Store. Para executá-lo basta baixá-lo e realizar a instalação. O link se encontra disponível no Apêndice D.

3 CONCLUSÃO

O desenvolvimento de um produto de software é complexo, cheio de desafios e decisões a serem tomadas. Como apresentado nesse trabalho, desde a etapa de identificação dos processos e regras de negócios, especificação dos requisitos, definição de processo de desenvolvimento, definições arquiteturais e implantação, em todas as etapas importantes decisões devem ser tomadas. Uma escolha errada pode impactar negativamente no projeto.

Buscou-se nesse trabalho utilizar a abordagem mais adequada ao contexto, baseada na metodologia Kanban, a especificação segundo um padrão estruturado e a adoção de estilo arquitetural incorporando padrões de projeto adequados às necessidades do aplicativo.

O resultado obtido no produto de software construído contempla o atendimento aos requisitos especificados e foi implementado seguindo as boas práticas do desenvolvimento de software.

Por fim, espera-se que esse trabalho possa contribuir para futuros Trabalhos de Conclusão de Curso que tenham como objetivo a construção de produtos de software.

REFERÊNCIAS

DATAVALID API Docs. 2021. Disponível em: https://apidocs.datavalidp.estaleiro.serpro.gov.br/. Acesso em: 24 jun. 2021.

FLUTTER (comp.). **Flutter Documentation**: versão 2.2. Versão 2.2. 2021. Documentação oficial flutter. Disponível em: https://flutter.dev/docs. Acesso em: 02 jun. 2021.

JAMES. **Modelo para Especificações de Requisitos**: edição 14.: agosto 2009. Edição 14—Agosto 2009. 2009. Por James & Suzanne Robertson — diretores da Associação Atlântica de Sistemas. Disponível em: https://www.volere.org/wp-content/uploads/2018/12/template14_ptbra.pdf. Acesso em: 02 jun. 2021.

KANBAN GUIDE. 2021. Disponível em: https://kanbanguides.org/html-kanban-guide/#DefinitionOfKanban. Acesso em: 24 jun. 2021.

MICROSOFT (org.). **Web API apps**: create web apis with asp.net core. Create web APIs with ASP.NET Core. 2021. Documentação oficial da microsoft. Disponível em: https://docs.microsoft.com/en-us/aspnet/core/web-api/?view=aspnetcore-5.0. Acesso em: 02 jun. 2021.

SCRUM GUIDES: The 2020 Scrum GuideTM. The 2020 Scrum GuideTM. 2021. Disponível em: https://scrumguides.org/scrum-guide.html. Acesso em: 24 jun. 2021.

VMWARE (org.). **RabbitMQ**: documentation. Documentation. 2021. Documentação oficial do RabbitMQ. Disponível em: https://www.rabbitmq.com/documentation.html. Acesso em: 02 jun. 2021.

Especificação de Requisitos de Software

Smart Bank: Uma proposta para automação de análise de crédito em banco digitais

Histórico de Revisões

Data	Versão	Descrição	Autor
22/09/2020	0.0.0.1	Requisitos Iniciais	R. S. Silva
01/10/2020	0.0.0.2	Casos de Uso	R. S. Silva
27/10/2020	0.0.0.3	Definição Casos de uso	R. S. Silva
24/11/2020	0.0.0.4	Formatação do documento	R. S. Silva

Sumário

1. Introdução 4
2. Descrição Geral 5
3. Requisitos Específicos 6
    3.1. Requisitos Funcionais 6
    3.2 Requisitos Não Funcionais 10
4. Caso de Usos 11
    1. Caso de Uso Geral 11
    2. Definição dos Caso de Uso 11
        1. Cadastrar Cliente 11
        2. Efetuar Login 12
        3. Redefinir Senha 12
        4. Solicitar Empréstimo 13

1 - Introdução

1.1 - Propósito

Esse documento apresenta de forma clara e objetiva o escopo e os requisitos elicitados do projeto Smart Bank. Tem o propósito de servir como um guia para o projeto, implementação, manutenção e extensão futura do produto de software.

Esse documento deve ser usado pelos Analistas, Arquitetos, Desenvolvedores e todas as partes interessadas no projeto.

1.2 Objetivos Específicos (Escopo)

O escopo desse projeto diz respeito a uma proposta de um banco digital, diferenciado dos modelos atuais, que visa a garantir a obtenção de um empréstimo a partir uma empresa de serviço financeiro, passando por todo processo de validação do cliente até a liberação do crédito, totalmente de maneira digital e integral pelo sistema, sem a intervenção humana.

1.2.1 Nome do produto e de seus componentes principais

SmartBank é apenas um nome fictício, não tem nenhum valor ou intenção comercial, apenas será usado para o projeto de Trabalho de Conclusão de Curso.

O sistema será divido em 4 produtos de software:

- Aplicativo mobile: usado para os clientes acessarem as suas contas.
- Serviço de gerenciamento dos clientes: uma Web API, responsável por prover a comunicação com a aplicação mobile, serviço de autenticação e serviço de auditoria.
- Serviço de autenticação: responsável por prover o acesso seguro a todos sistemas.
- Serviço de auditoria: é o *core* do projeto, esse serviço é o responsável por validar as documentações do cliente.

1.2.2 Missão

O principal objetivo do projeto é propor uma possibilidade de se melhorar um processo de solicitação de serviço financeiro em um banco digital.

1.2.3 Limites

Todo código fonte, bem como a documentação será disponibilizada gratuitamente no Github. Trata-se apenas uma aplicação acadêmica e não será usada para fins comerciais.

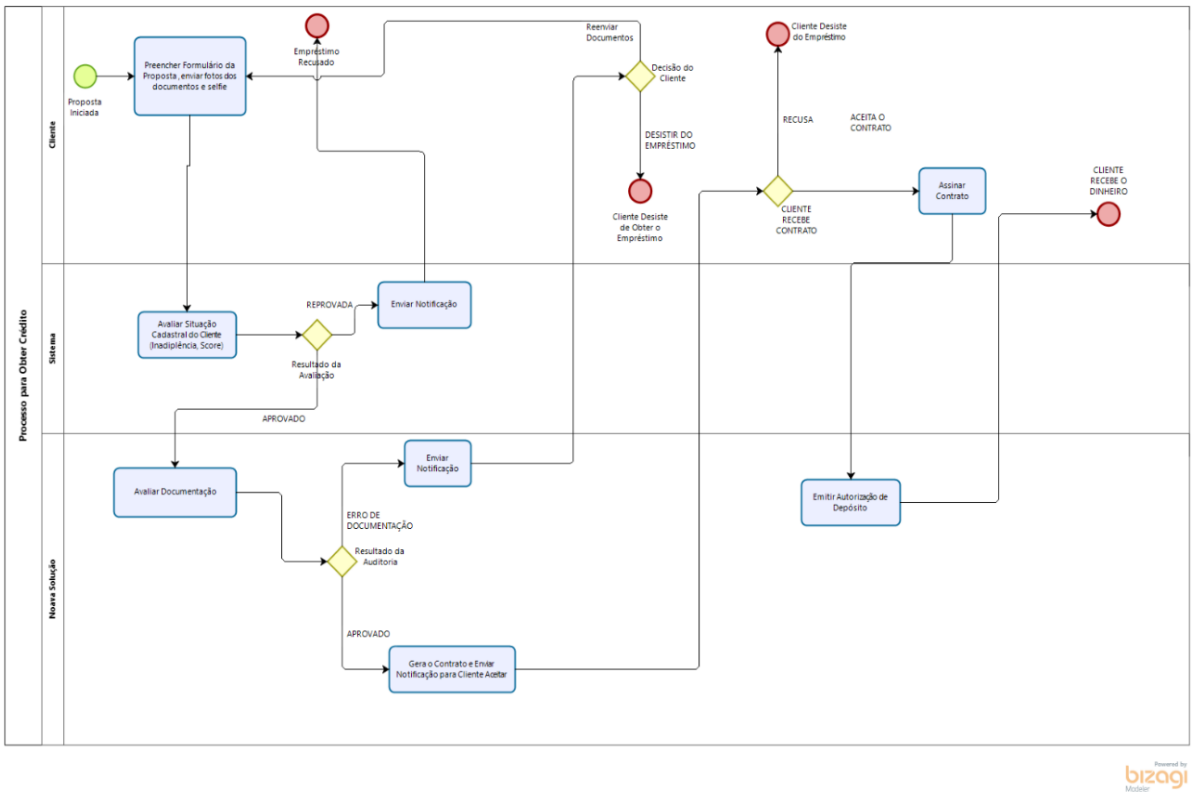
1.3 Definições, siglas e abreviações

CSU – Caso de Uso

BPMN- *Business Process Modelling Notation*

2 - Descrição Geral

O processo de solicitação de serviço financeiro que essa aplicação deve atender está modelado no diagrama de processo de negócio, seguindo a notação BPMN a seguir:



Pode-se observar 3 atores básicos: o cliente, a aplicação *mobile* para o cliente e o serviço de auditoria de documentação. A sequência do fluxo do processo pode ser descrita:

1. Cliente inicia a proposta de obtenção do serviço financeiro, preenchendo um formulário e comprovando sua identidade. Essa comprovação de identidade pode ser feita de 2 formas possível: por selfie ao lado de um documento de identificação ou pela digital.

2. Um sistema intermediário fica responsável por validar os dados cadastrais do cliente. Essa validação é apenas de existência da pessoa, ainda não comprova se o solicitante é o verdadeiro proprietário da documentação. Caso seja reprovado, uma notificação é retornada para o cliente, em caso de aprovação seguiremos para a próxima etapa.

3. Uma outra etapa “Avaliar Documentação” nesse ponto é que determinamos a verdadeira identidade do solicitante.

3 - Requisitos Específicos

3.1 - Requisitos Funcionais

3.1.1 – Cadastrar Cliente

Identificação do Requisito: 1

Nome: Cadastrar Cliente

Descrição: O usuário cliente, deve ser possível realizar o cadastro preenchendo os dados básicos de acesso:

nome,
e-mail,
telefone,
cpf,
rg ou cnh,
endereço,
data de nascimento,
nome da mãe,
nome do pai (opcional).

Justificativa: Dados são essenciais para uso do sistema.

Critério de Aceitação:

Ser possível criar uma conta no sistema.

Dependências: Nenhuma

Conflitos: O cliente pode já ter realizado o cadastro, nessa situação o sistema deve notificá-lo, solicitando para realizar autenticação na conta existente.

3.1.2 – Autenticar Usuário Cliente no Sistema

Identificação do Requisito: 2

Nome: Autenticar Cliente

Descrição: O usuário cliente, deve ser possível realizar sua autenticação informand o apenas o cpf e senha de acesso.

Justificativa: Dados essencias para realizar autenticação no sistema.

Critério de Aceitação:

Uma vez realizado um cadastro de usuário cliente, deve ser possível autenticar o mesmo no sistema informando seus dados de acesso informado no cadastro.

Dependências: Requisito 1.

Conflitos: Nenhum.

3.1.3 – Redefinir Senha do Usuário Cliente

Identificação do Requisito: 3

Nome: Redefinir senha do cliente

Descrição: O usuário cliente, tendo seu cadastro realizado através do aplicativo, de ve ser possível redefinir a senha, informando:

cpf,
data de nascimento,
nome da mãe,
email.

Justificativa: Para os usuários que esqueceram suas credenciais de acesso.

Critério de Aceitação: Tendo realizado um cadastro de usuário cliente, deve ser po ssível redefinir seus dados de acesso, cadastrando uma nova senha.

Dependências: Requisito 1.

Conflitos: Nenhum.

3.1.4 – Solicitar Serviço Financeiro

Identificação do Requisito: 4

Nome: Solicitar Serviço Financeiro

Descrição: O usuário cliente, tendo seu cadastro realizado através do aplicativo, de ve ser possível solicitar um serviço financeiro, como empréstimo.

Os dados necessários:

foto do documento de identificação (RG, CNH),
Selfie, dados profissionais (salário, empresa que trabalha, comprovante de renda), comprovante de endereço.

Justificativa: Necessário permitir solicitação de serviço via aplicativo.

Critério de Aceitação: Tendo realizado um cadastro de usuário cliente, deve ser

possível solicitar um empréstimo.

Dependências: Requisito 1.

Conflitos: Nenhum.

3.1.5 - Validar Documentos

Identificação do Requisito: 5

Nome: Validar Documentos

Descrição: O sistema deve buscar nos mais diversos sistemas públicos a comprova ção de existência da pessoa informada no formulário.

Validar a Existência do CPF, documento de identificação, comparar foto da *selfie* com documento de identificação, validar a digital do cliente com a fornecida no RG.

Justificativa: Evitar fraudes e processo manual.

Critério de Aceitação: Tendo o cliente realizado a solicitação de empréstimo,
o sistema deve usar métodos para validar a veracidade das informações fornecida.

Dependências: Requisito 1.

Conflitos: Nenhum.

3.1.6 - Avaliar Possibilidade de Crédito

Identificação do Requisito: 6

Nome: Avaliar Possibilidade de Liberação de Crédito

Descrição: O sistema deve avaliar o histórico financeiro do cliente e simular uma possível liberação de crédito.

Com essa liberação um contrato deve ser retornado, para conscientização por parte do cliente.

Justificativa: Complemento do processo.

Critério de Aceitação: Tendo o cliente realizado a solicitação de empréstimo e o sistema aprovado a documentação enviada na auditoria, deve ser retornado um contrato para conscientização do cliente, sobre os critérios de empréstimo.

Dependências: Requisito 5.

Conflitos: Nenhum.

3.2 Requisitos Não Funcionais

3.2.1 - Plataformas que devem ser atendidas

O aplicativo mobile deve ser compatível com a plataforma Android.

Identificação do Requisito: 7

Nome: Plataformas atendida pela aplicação mobile

Descrição: O sistema é uma aplicação de prova de conceito, por isso apenas a plataforma Android deverá ser atendida inicialmente.

Justificativa: Tempo para implementação.

Critério de Aceitação: O aplicativo deve ser disponibilizado na *Play Store*.

3.2.2 - Ambientes de Implantação

O projeto deve ser desenvolvido seguindo diversos padrões arquiteturais, sendo o mais importante para o ambiente de implantação, que as diversas APIs esteja distribuída como microsserviços.

Identificação do Requisito: 8

Nome: Ambientes de Implantação do Servidor

Descrição: O *back-end* do sistema deve ser implantado na Plataforma Azure.

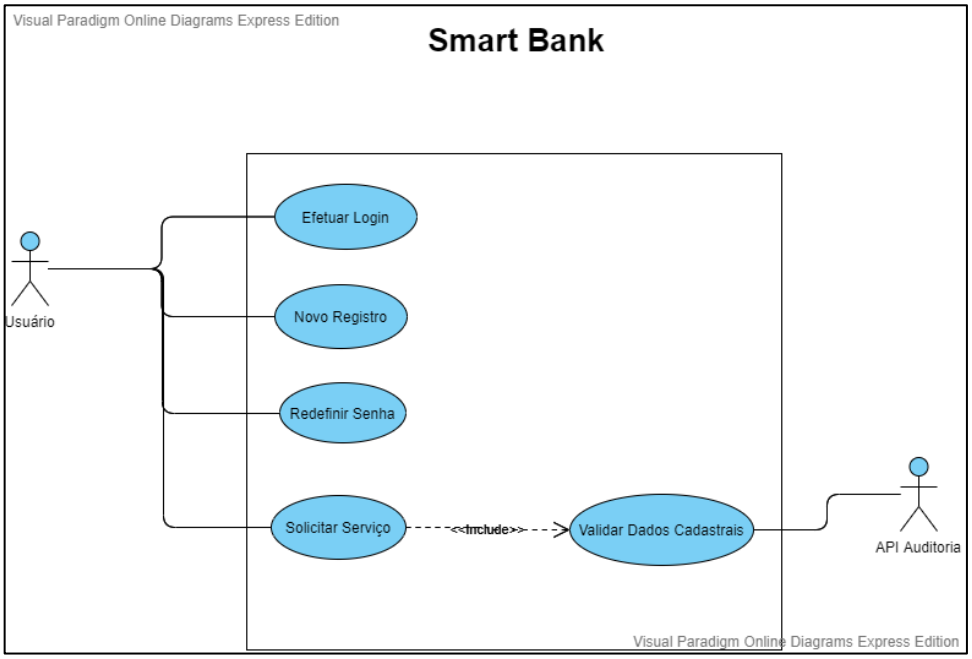
Justificativa: Tempo para implementação.

Critério de Aceitação: O projeto final deve ser semelhante ao diagrama do anexo 2.

4 Casos de Uso

4.1 Caso de Uso Geral

Diagrama de Caso de Uso Geral



4.2 Definição dos Caso de Uso

4.2.1 - CSU01 Cadastrar Cliente

Introdução: Cliente cria sua conta no aplicativo.
Ator Primário: Cliente
Atores Secundários: Não existe
Precondições: Cliente deve ter instalado o aplicativo em seu smartphone
Fluxo Principal

1. O cliente inicia o aplicativo, a partir do seu smartphone.
2. O sistema apresenta a tela de splash e na sequência a tela de login.
3. O cliente clica na opção novo usuário.
4. O sistema exibe a tela de pré-cadastro de novo cliente.
5. O cliente preenche seus dados básicos: nome completo, endereço, filiação, e-mail, telefone, senha e aciona opção salvar.
6. O sistema processa os dados do cliente, e envia um código de verificação por e-mail e exibe a tela para preenchimento do código de verificação.
7. O cliente preenche o código de verificação recebido no e-mail e aciona opção enviar.
8. O sistema aceita o código de verificação, finaliza o registro e redireciona o cliente para a tela inicial do sistema.

Fluxo de Exceção:

5.1 Cliente Já Cadastrado, o sistema informa sobre o cadastro existente, e retorna ao passo 2.
6.1 CPF inválido, o sistema informa ao usuário e retorna para o passo 4.
6.2 Cadastro existente não finalizado, o sistema informa de um cadastro em andamento com esses dados, e redireciona para o passo 7.

7.1 Código de verificação incorreto, o sistema deve retornar a mensagem de “Código de verificação não reconhecido”.

4.2.2 - CSU02 Efetuar Login

Introdução: Cliente deve realizar autenticação, antes de usar o aplicativo.
Ator Primário: Cliente
Atores Secundários: Não existe
Precondições: Cliente deve ter instalado o aplicativo em seu smartphone e ter realizado o CSU01
Fluxo Principal

1. O cliente inicia o aplicativo, a partir do seu smartphone.
2. O sistema apresenta a tela de splash e na sequência a tela de login.
3. O cliente preenche seu e-mail e senha de acesso e aciona opção entrar.
4. O sistema aceita os dados do usuário, e retorna para a tela inicial.

Fluxo Alternativo

a. O cliente não possui cadastro, cliente aciona opção “novo cadastro”, sistema vai para o passo CSU01.
b. O cliente esqueceu a senha, aciona a opção “esqueci senha”, vai para o CSU03.

Fluxo de Exceção:

4.1 Dados de acessos inválido. O sistema informa ao cliente que seus dados de acesso estão incorretos e retorna ao passo 3.
4.2 - Usuário não cadastrado, o sistema informa ao cliente que não existe cadastro com e-mail fornecido, e retorna ao passo 3.

4.2.3 - CSU03 Redefinir Senha

Introdução: O cliente esqueceu sua senha ou e-mail de acesso.
Ator Primário: Cliente
Atores Secundários: Não existe
Precondições: Cliente deve ter instalado o aplicativo em seu smartphone e ter realizado o CSU01
Fluxo Principal

1. O cliente inicia o aplicativo, a partir do seu smartphone.
2. O sistema apresenta a tela de splash e na sequência a tela de login.
3. O cliente clica na opção "esqueci senha".
4. O sistema exibe a tela de redefinição de senha.
5. O cliente informa o seu CPF ou e-mail e acionar opção enviar.
6. O sistema envia um código para o e-mail do cliente.
7. Cliente informa o código recebido no e-mail e aciona opção enviar.
8. O sistema aceita o código e exibe a tela para preenchimento da nova senha.
9. O cliente preenche a nova senha e aciona a opção enviar.
10. O sistema redireciona usuário para tela de login.

Fluxo de Exceção:

1. CPF/e-mail inválido ou não encontrado: sistema emite uma mensagem “(CPF ou e-mail) não encontrado” e retorna ao passo 5.

4.2.4 - CSU04 Solicitar Empréstimo

Introdução: O cliente deseja contratar um empréstimo.
Ator Primário: Cliente
Atores Secundários: API de Validação de Documentos

Precondições: Cliente deve ter usuário cadastrado.

Fluxo Principal

1. Na tela inicial, o cliente aciona a opção “Solicitar Empréstimo”.
2. O sistema exibe o formulário para simulação dos valores.
3. O cliente preenche os dados e aciona a opção “envia”.
4. O sistema valida os dados e simula o valor desejado pelo cliente, e envia os dados para API de validação.
5. API de validação determina a disponibilidade de crédito para o cliente e retorna para o sistema principal.
6. O sistema retorna os valores junto com uma proposta do serviço.
7. O cliente aciona a opção aceitar.
8. Sistema finaliza a transação.

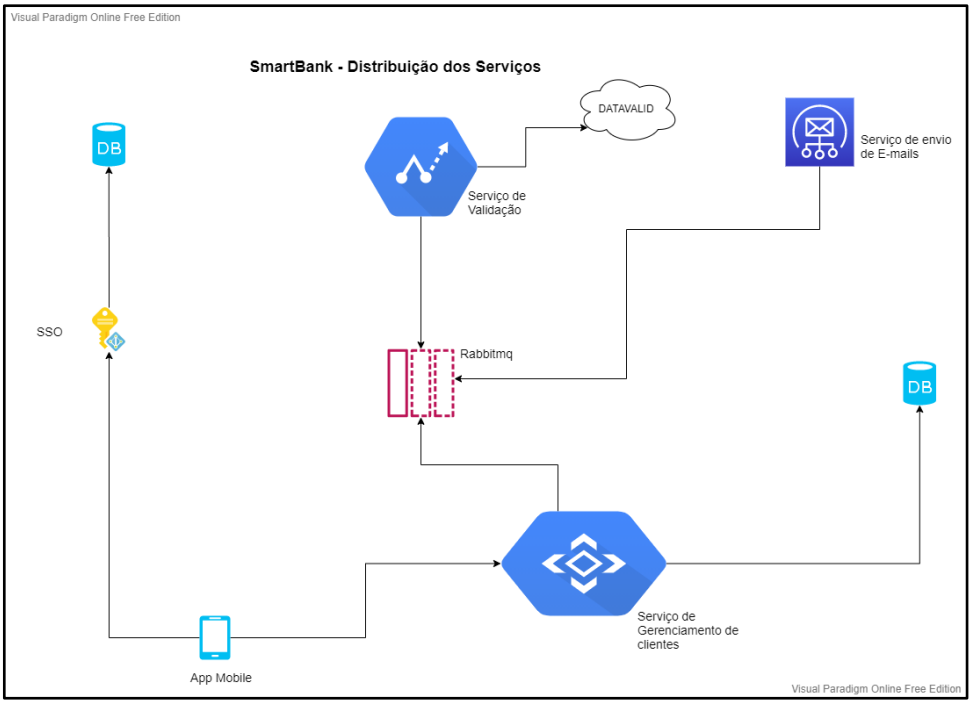
Fluxo de Exceção:

5.1 Cliente não atende aos requisitos para obter o valor de empréstimos: nesse caso, o sistema retorna ao passo 3.

5.2 A solicitação não foi aprovada, o sistema informa ao cliente sobre o a reprovação e retorna para a tela inicial.

APÊNDICE B – DISTRIBUIÇÃO DOS SERVIÇOS

Esquematicamente, os componentes da solução podem ser representada conforme a figura a seguir.



O sistema como um todo é composto por o aplicativo *mobile*, serviço de autenticação, gerenciamento de clientes, *e-mail*, validação, e ainda existe um serviço de mensagerias, o *Rabbitmq*. A responsabilidade de cada serviço e suas interações são descritas a seguir:

- SSO: o serviço de auntenticação, independente e com seu próprio banco de dados. É o responsável por gerenciar os usuários no sistema e gerar tokens de acesso. Esse serviço não integra com nenhum outro, mas todos interagem com ele. Pode ser acessado no link: https://sso.ronilson.dev.
- Serviço de Gerenciamento de Clientes: API REST, responsável por prover o cadastro dos clientes, solicitações e validações. Esse é o principal serviço de todo projeto, interagem com todos demais, sendo também o ponto mais crítico

do sistema. A definição dos *endpoints* pode ser encontrada no link: https://sbcliente.azurewebsites.net/swagger.

- Serviço de envio de E-mails: API REST, com um único *endpoint*, que recebe no corpo da requisição dados da mensagem que deseja enviar. Não possui banco de dados, a definição da API pode ser encontrada no link: https://sbemail.ronilson.dev/swagger.
- Serviço de Validação: API REST, responsável por integrar com o serviço Datavalid da SERPRO. Não realiza persistência em banco de dados, apenas processa as requisições que recebe e retorna o resultado para o solicitante, nesse contexto o serviço de gerenciamento de clientes que comunica com o serviço de validação. A definição da API pode ser encontrada no link: https://sbdatavalid.ronilson.dev/swagger.
- O serviço de mensageria *Rabbitmq* e o Datavalid são providos por terceiros.

APÊNDICE C – CÓDIGO FONTE DO PROJETO NO GITHUB

O código fonte do projeto está disponível em:

https://github.com/ronilsonsilva/smartbank

APÊNDICE D – APLICATIVO NA PLAYSTORE

O aplicativo final do projeto, está disponível na loja virtual do Google, no seguinte link:

https://play.google.com/store/apps/details?id=com.smar_bank_app