PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS

PUC Minas Virtual

Pós-graduação *Lato Sensu* em Arquitetura de *Software* Distribuído

Projeto Integrado

Relatório Técnico

My-Customers - Sistema para Gestão de Clientes em Bancos

Cleidson Ferreira de Jesus

Belo Horizonte
Abril 2022.

Projeto Integrado – Arquitetura de Software Distribuído

Sumário

1. Introdução

O sistema financeiro nacional tem status constitucional e possui vital importância para o desenvolvimento econômico do país, os bancos promovem a logística complexa e arriscada das intermediações financeiras, atuando em extremos, desde transações milionárias em financiamento de grandes empresas até o cidadão comum, proporcionando acesso a serviços financeiros gratuitos através dos bancos digitais.

No ano de 2020 a pandemia da covid-19 impulsionou o celular a tornar-se o canal favorito dos brasileiros para pagar contas, fazer transferências, contratar crédito dentre outras operações bancárias, no mesmo ano, pela primeira vez as transações realizadas nos apps bancários representaram mais da metade (51%) do total das operações feitas no país, as transações feitas pelo celular chegaram a 52.9 bilhões, ante 37 bilhões no ano anterior, se considerados todos os canais bancários (celular, internet, maquininhas, agências, caixas eletrônicos, correspondentes bancários e *contact centers*) o total das operações chega a 103,5 bilhões, um crescimento de 20%.

Bancos tradicionais possuem um enorme legado tecnológico que não é facilmente escalável e adaptável, tecnologias antigas, acoplamento de sistemas e inexistência de padrões arquiteturais dificultam e podem até inviabilizar o suporte ao crescimento apresentado, além disso, é preciso conviver com *fintechs* que já são constituídas digitalmente e não enfrentam esse tipo de desafio.

A modernização do legado tecnológico é essencial para que os bancos tradicionais se mantenham competitivos no mercado, através da modernização é esperado diminuir o tempo gasto com o processo de *onboarding* de clientes, melhorar a estrutura dos dados facilitando a prestação de contas perante a órgãos reguladores, melhor integração entre sistemas diminuindo o acoplamento e facilitando futuras evoluções e por fim, possibilitar a construção de aplicativos intuitivos que evitem a ida de clientes a agências, gerando assim economia de recursos.

Para que os ganhos apresentados sejam alcançados um dos sistemas que deve ser modernizado é de gerenciamento de pessoas, o objetivo deste trabalho é apresentar o projeto arquitetural para construção dessa aplicação visando alcançar três objetivos:

- Construir um sistema que centralize e gerencie dados de clientes, atendendo a regulamentação imposta por órgãos governamentais.

- Projetar uma arquitetura resiliente, escalável e segura para suportar o crescimento da operação no decorrer dos anos.
- Permitir que as áreas de negócio foquem no que realmente importa fornecendo um sistema simples e eficiente.

2. Cronograma do Trabalho

Quadro 1: Cronograma de atividades.

Datas		Atividade / Tarefa	Produto / Resultado
De	**Até**		
07/03/2022	07/03/2022	1. Desenvolver cronograma do trabalho	Tabela com cronograma do trabalho
08/03/2022	09/03/2022	2. Redigir introdução do trabalho	Tópico de introdução do trabalho
10/03/2022	15/03/2022	3. Definir restrições arquiteturais	Lista com restrições arquiteturais identificadas
16/03/2022	21/03/2022	4. Definir requisitos funcionais	Lista com requisitos funcionais identificados
22/03/2022	25/03/2022	5. Definir requisitos não-funcionais	Lista com requisitos não-funcionais identificados
28/03/2022	30/03/2022	6. Definir mecanismos arquiteturais	Lista com mecanismos arquiteturais identificados
31/03/2022	04/04/2022	7. Elaborar diagrama de contexto	Diagrama de contexto da solução
05/04/2022	07/04/2022	8. Gravar vídeo de apresentação etapa 1	Vídeo de apresentação da etapa 1
08/04/2022	11/04/2022	9. Construir apresentação em PPT da etapa 1	Apresentação em Power Point
12/04/2022	12/04/2022	10. Revisão da etapa 1	Etapa 1 revisada
13/04/2022	13/04/2022	11. Publicar no GitHub arquivos produzidos na etapa 1	Arquivos produzidos disponíveis para acesso público
18/04/2022	20/04/2022	12. Elaborar diagrama de containers	Diagrama de containers da solução
21/04/2022	22/04/2022	13. Elaborar diagrama de componentes	Diagrama de componentes da solução
25/04/2022	26/04/2022	14. Elaborar wireframe da POC	Protótipo do visual do projeto
27/04/2022	10/06/2022	15. Implementar três requisitos funcionais do total listado	Três requisitos funcionais implementados
13/06/2022	13/06/2022	16. Revisão da etapa 2	Etapa 2 revisada
14/06/2022	14/06/2022	17. Publicar no GitHub arquivos produzidos na etapa 2	Arquivos produzidos disponíveis para acesso público
20/06/2022	23/06/2022	18. Redigir tópico Análise das abordagens arquiteturais	Tópico Análise das abordagens arquiteturais
27/06/2022	30/06/2022	19. Redigir tópico Cenários	Tópico Cenários

04/06/2022	11/07/2022	20. Redigir tópico Evidências da avaliação	Tópico Evidências da avaliação
12/07/2022	18/07/2022	21. Redigir tópico Resultados obtidos	Tópico Resultados Obtidos
19/07/2022	19/07/2022	22. Redigir tópico Avaliação crítica dos resultados	Tópico Avaliação crítica dos resultados
20/06/2022	26/07/2022	23. Redigir conclusão	Tópico Conclusão
27/07/2022	02/08/2022	24. Gravar vídeo de apresentação da etapa 3	Vídeo de apresentação da etapa 3
03/08/2022	03/08/2022	25. Revisão da etapa 3	Etapa 3 revisada
15/08/2022	15/08/2022	26. Publicar no GitHub arquivos produzidos na etapa 3	Arquivos produzidos disponíveis para acesso público

3. Especificação Arquitetural da solução

Esta seção apresenta a especificação básica da arquitetura da solução a ser desenvolvida, incluindo diagramas, restrições e requisitos definidos, tal que permitem visualizar a macroarquitetura da solução.

3.1 Restrições Arquiteturais

ID	Descrição
R01	O back-end deve ser desenvolvido com .Net Core C#
R02	O front-end deve ser desenvolvido com Angular
R03	Pacotes Nuget ou NPM a serem utilizados no projeto devem ser obtidos exclusivamente do repositório interno, a inclusão de novos pacotes no repositório deve ser avaliada pela equipe de segurança
R04	Implementar RabbitMQ como event broker onde necessário a comunicação orientada a eventos
R05	Toda a comunicação de origem externa ao ambiente deve passar por um API Gateway
R06	O registro de log's deve ser feito utilizando a slack ELK

3.2 Requisitos Funcionais

ID	Descrição Resumida	Dificuldade (B/M/A)*	Prioridade (B/M/A)*
RF01	As implementações efetuadas para este sistema devem atender as orientações estabelecidas pela LGPD.	A	A
RF02	Deve ser possível cadastrar os dados genéricos de clientes	A	A
RF03	Deve ser possível cadastrar os dados específicos de clientes	A	A
RF04	Deve ser possível cadastrar os representantes para Pessoa Física e para Pessoa Jurídica	M	A
RF05	Para clientes menores de idade a inclusão de ao menos um representante é obrigatória	B	M

RF06	Para clientes Pessoa Jurídica é obrigatória a inclusão de ao menos um representante	B	M
RF07	Para clientes analfabetos é obrigatório a inclusão das informações do rogado	B	M
RF08	Deve ser possível cadastrar a composição societária dos clientes Pessoa Jurídica	M	A
RF09	Deve ser possível cadastrar o faturamento dos clientes Pessoa Jurídica	M	A
RF10	O porte dos clientes Pessoa Jurídica deve ser calculado com base no faturamento cadastrado	B	M
RF11	O sistema deve obter e persistir informações de clientes PEP (Politicamente Exposto)	M	A
RF12	O sistema deve obter e persistir a exposição em mídia de clientes	M	M
RF13	O sistema deve obter e persistir o status da situação cadastral de Pessoas Físicas e Jurídicas perante a receita federal	M	M
RF14	O sistema deve iniciar o processo de encerramento de contas para clientes com o status irregular na receita federal	A	A
RF15	O sistema deve reclassificar automaticamente o risco de lavagem de dinheiro dos clientes	A	B
RF16	O sistema deve calcular a data da renovação cadastral dos clientes	A	M
RF17	O sistema deve obter e persistir informações de clientes falecidos	M	M
RF18	O sistema deve obter e persistir a lista de clientes optantes pelo cadastro positivo	M	M
RF19	O sistema deve atualizar automaticamente o porte dos clientes Pessoa Física de acordo com o valor do salário-mínimo vigente	M	B
RF20	O cadastro de clientes originados no onboarding simplificado deve ser completado automaticamente através de consultas em bureaus	A	M
RF21	O sistema deverá gerar o arquivo de dados cadastrais para envio ao E-Financeira	M	A

*B=Baixa, M=Média, A=Alta.

3.3 Requisitos Não-funcionais

ID	Descrição	Prioridade B/M/A
RNF01	Deve ser registrado trilha de auditoria para todas as alterações que ocorrerem no cadastro de clientes	A
RNF02	As API's devem possuir um throughput de no mínimo 340 tps	A
RNF03	O sistema deve possuir disponibilidade 24x7x365	A
RNF04	O front-end deve ser compatível com os browsers: Chrome, Firefox e Edge	M
RNF05	O front-end deve apresentar características de design que facilitem o uso por PCD's	A
RNF06	O deploy em produção das aplicações e a infraestrutura devem ser automatizadas usando pipelines CI/CD	M
RNF07	O processamento de grandes quantidades de dados deve ser feito preferencialmente após as 20h	M

3.4 Mecanismos Arquiteturais

Análise	*Design*	Implementação
Persistência	ORM	Dapper
Persistência	Banco de dados relacional	Mysql
Front-end	Single Page Application	Angular
Back-end	Arquitetura em camadas	.Net Core
Gateway	API Gateway	AWS API Gateway
Log	Solução para gestão de logs	ELK
Teste de Software	Teste unitários	xUnit
	Teste de carga	JMeter
Versionamento	Versionamento do código das aplicações	GIT
Distribuição	Integração e Entrega Continua (CI/CD)	AWS Code Pipelines
Barramento de mensagens	Integração entre sistemas através de mensageria	RabbitMQ
Documentação de API	Solução para documentação das APIs da solução	Swagger

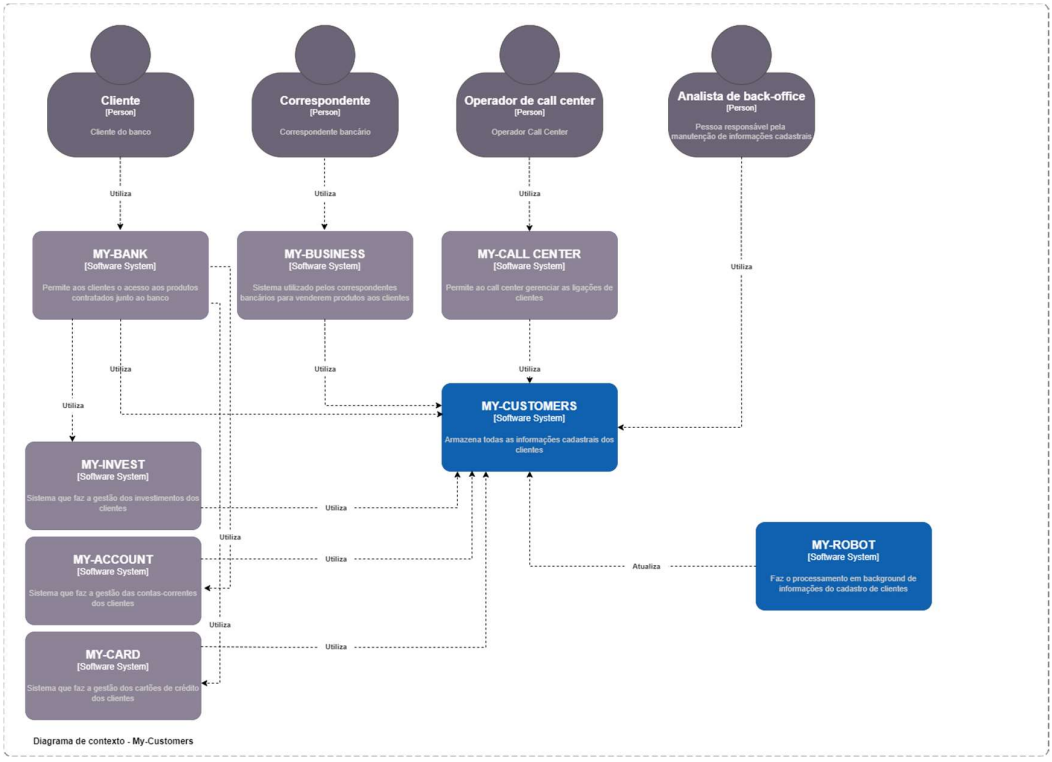
4. *Modelagem Arquitetural*

Esta seção apresenta a modelagem arquitetural da solução proposta, de forma a permitir seu completo entendimento visando à implementação da prova de conceito do sistema My-Customers na seção 5.

Para esta modelagem arquitetural optou-se por utilizar o modelo C4 para documentação de arquitetura de software. Mais informações a respeito podem ser encontradas nos links: https://c4model.com/ e https://www.infoq.com/br/articles/C4-architecture-model/. Dos quatro níveis que compõem o modelo C4 três serão apresentados aqui e somente o código será apresentado na próxima seção (5).

4.1 *Diagrama de Contexto*

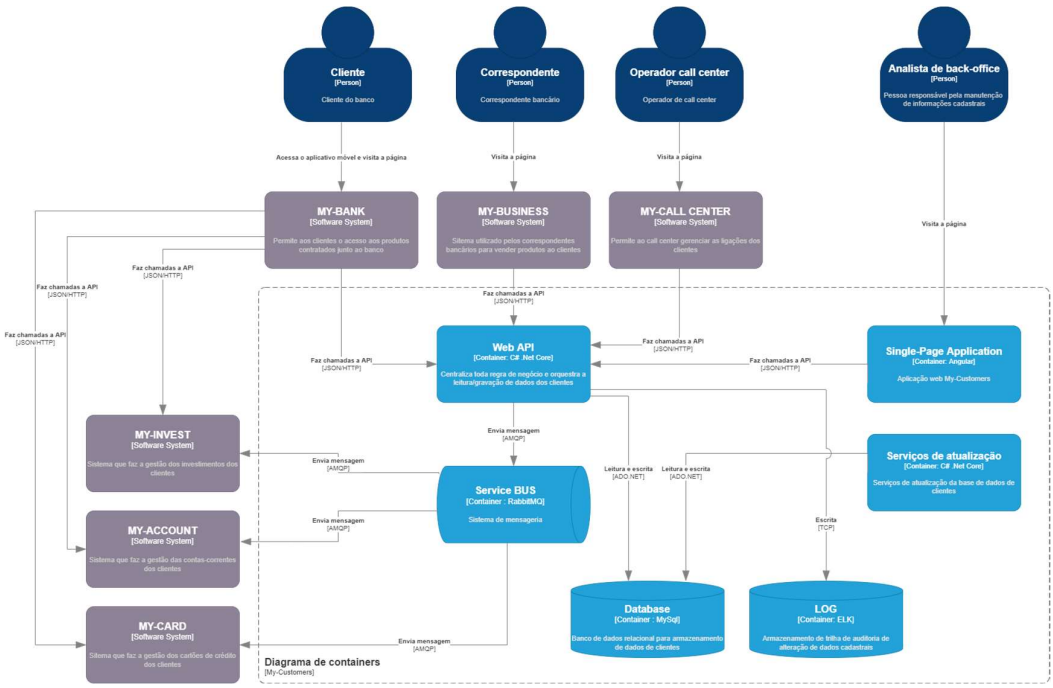
Figura 1 - Visão Geral da Solução



A Figura 1 mostra o contexto em que a aplicação My-Customers será inserida, demostrando seus principais módulos e integrações.

4.2 Diagrama de Container

Figura 2 – Diagrama de container



A Figura 2 apresenta os containers da aplicação e a forma como estão organizados, a arquitetura visa manter somente uma fonte de verdade tratando-se dos dados de clientes, o database apresentado é a fonte de dados oficial do banco perante o Banco Central.

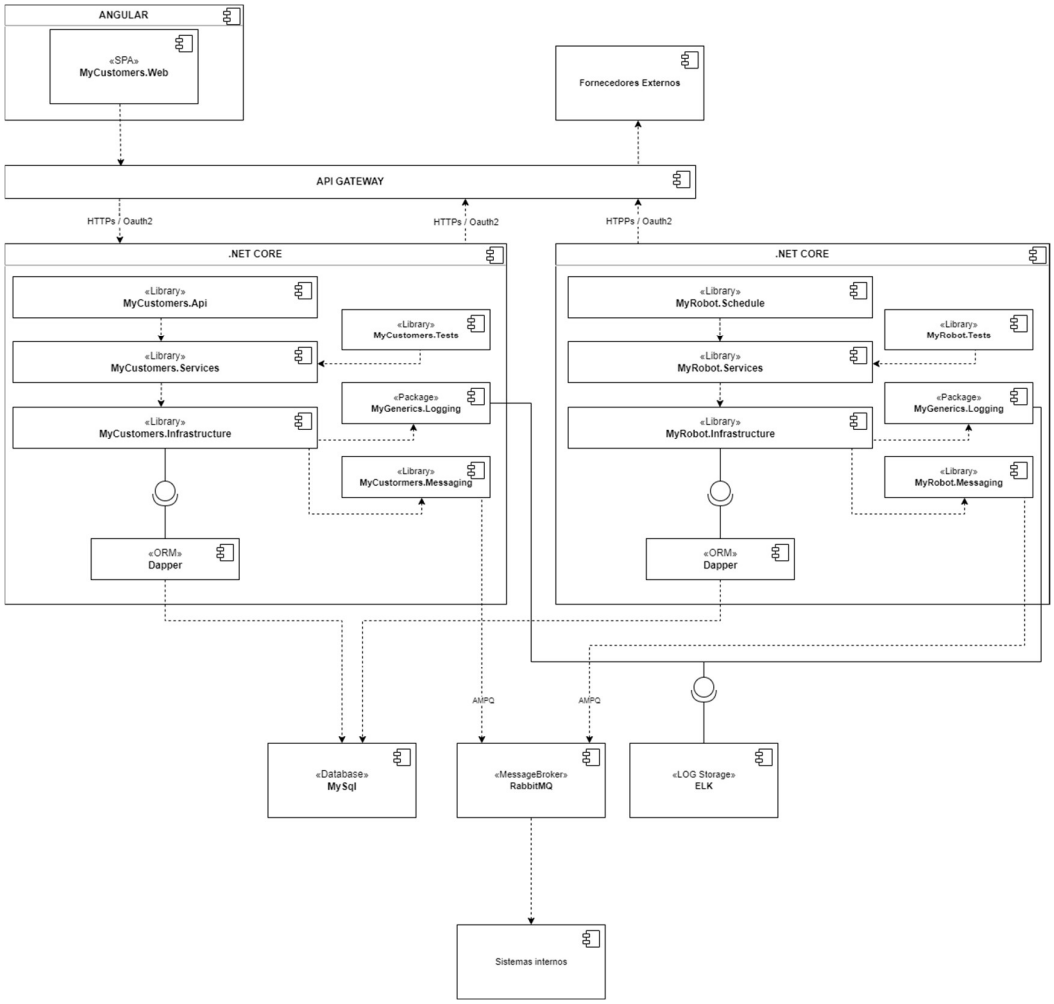
Os demais sistemas recebem novos clientes e atualizações através de mensageria, caso necessário eles armazenam informações pontuais em sua estrutura funcionado assim de forma desacoplada. Além disso, a implantação de novos sistemas fica mais simples, para receber informações de clientes basta que o novo sistema seja plugado na mensageria.

Toda manipulação de dados cadastrais é feita pela Web-Api do My-Customers ou pelos serviços de atualização, isso garante a consistência de dados já que a regra de negócio sempre será respeitada.

A rastreabilidade de alterações cadastrais é garantida pelo LOG, a solução será implementada em uma plataforma permite a pesquisa em segundos da trilha de auditoria, possibilitando rápida resposta em caso de questionamento de órgãos reguladores ou até mesmo em identificação de fraudes.

4.3 Diagrama de Componentes

Figura 3 – Diagrama de componentes



A Figura 3 apresenta os componentes da aplicação, estes componentes podem ser detalhados como:

- Angular – Plataforma/Framework de desenvolvimento de sistemas WEB.
- MyCustomers.Web – SPA que disponibiliza UI para utilização de usuários internos do banco.
- Fornecedores Externos – Empresas parceiras que disponibilizam informações a serem utilizados pelo banco, mas que não são de seu domínio.
- API Gateway – Ferramenta responsável por orquestrar chamadas de API's.

- .NET Core – Framework utilizado no desenvolvimento do back-end da solução My-Customers e do robô My-Robot.
- MyCustomers.Api – Projeto responsável pela organização e exposição das rotas da API.
- MyCustomers.Core – Projeto que contém toda a regra de negócio da API.
- MuCustomers.InfraStructure – Projeto responsável por centralizar a comunicação com o banco de dados e a mensageria.
- MyCustomers.Tests – Projeto com os testes automatizados da API.
- MyGenerics.Logging – Pacote responsável pela comunicação com o logstash.
- MyCustomers.Messaging – Projeto responsável por orquestrar a comunicação com o serviço de mensageria.
- Dapper – ORM responsável pela comunicação com o banco de dados.
- MyRobot.Schedule – Projeto responsável por organizar todos os serviços que vão executar no MyRobot.
- MyRobot.Services – Projeto com a regra de negócio dos serviços que serão executados no MyRobot.
- MuRobot.InfraStructure – Projeto responsável por centralizar a comunicação com o banco de dados e a mensageria.
- MyRobot.Tests – Projeto com os testes automatizados do robô.
- MyRobot.Messaging – Projeto responsável por orquestrar a comunicação com o serviço de mensageria.
- MySql – Banco de dados relacional que vai persistir os dados de clientes.
- RabbitMQ – Ferramenta responsável pela mensageria.
- ELK – Junção das ferramentas ElasticSearch, Logstash e Kibana responsáveis pelo armazenamento e consulta de logs.
- Sistemas Internos – Demais sistemas que compõem o ambiente interno do banco.

5. Prova de Conceito (PoC)

Para validar algumas implementações propostas para a aplicação foram utilizadas provas de conceitos (POC), três aspectos de alta importância para a arquitetura foram contemplados:

1. **Persistência de dados:** O objetivo dessa poc foi avaliar o micro ORM Dapper a fim de entender sua implementação e funcionamento, buscando comprovar sua performance que é considerada uma vantagem em relação a outras bibliotecas semelhantes. Espera-se que com a utilização da biblioteca seja possível obter maior controle na performance de querys;

2. **Design pattern para API:** O objetivo dessa poc foi comprovar que a arquitetura em camadas alinhada a conceitos de DDD (Domain-Driven Design) pode organizar bem o código diante da grande quantidade de regras de negócio que compõem a aplicação. Espera-se que seja possível incluir, retirar ou modificar regras de negócio de forma simples e rápida;

3. **Angular:** O objetivo dessa POC foi estudar a utilização do framework Angular para desenvolvimento do front-end que será utilizado pelo backoffice da área de cadastro. Espera-se que com a utilização desse framework seja possível desenvolver um sistema responsivo, performático e que não dependa de muitos componentes de terceiros para seu funcionamento.

5.1 Integrações entre Componentes

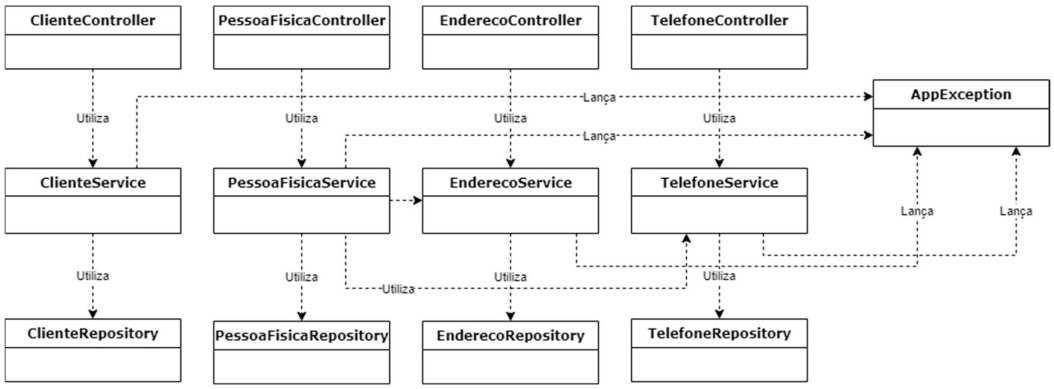
Para apresentação da POC foi desenvolvido um protótipo navegável utilizando a ferramenta Figma, o wireframe e o protótipo estão disponíveis através dos links:

Wireframe:
https://www.figma.com/file/61sc72Z37exQtdckIfP4qK/MyCustomers?node-id=0%3A1

Protótipo:
https://www.figma.com/proto/61sc72Z37exQtdckIfP4qK/MyCustomers?node-id=4%3A25&scaling=min-zoom&page-id=0%3A1&starting-point-node-id=2%3A2

5.2 Código da Aplicação

Figura 4 – Código da aplicação



A figura 4 demonstra como a estrutura da aplicação foi organizada e como os componentes implementados se relacionam.

Os componentes implementados são divididos em:

- Controllers: Responsáveis por fazerem o roteamento das requisições recebidas pela API e implementar as definições que permitem a geração dinâmica da documentação da API.
- Services: Responsáveis por comportar toda a regra de negócio da API, trata-se do core da API onde a maior parte do trabalho é executado.
- Repositorys: Responsáveis por fazer a persistência de dados.
- AppException: Middleware implementado para controlar as exceções lançadas pela API e adequar o retorno utilizando http status.

Link API: http://mycustomersapi.us-east-1.elasticbeanstalk.com/swagger/index.html
Link Front: http://mycustomersfront.s3-website-us-east-1.amazonaws.com/

Usuário: usuario
Senha: senh@123

Existem clientes incluídos no banco de dados com o nome “Cleidson”, após o login pesquisar por esse nome utilizando o input “Procure por um usuário”, a pesquisa é ativada quando a tecla enter é pressionada.

Após esse procedimento será exibida uma lista de clientes que possuem o nome pesquisado, clique em cima de algum dos clientes exibidos para abrir o cadastro, os campos exibidos podem ser editados e persistidos no banco de dados através do botão salvar.

Etapa 3 – Conteúdo a ser produzido

Referências

SOUZA, Ludmila**. Pandemia mudou a relação dos brasileiros com tecnologias bancárias.** São Paulo: Agência Brasil**,** 2021. Disponível em: https://agenciabrasil.ebc.com.br/economia/noticia/2021-06/pandemia-mudou-relacao-dos-brasileiros-com-tecnologias-bancarias **.** Acesso em: 03 de março de 2022.