PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS

PUC Minas Virtual

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

Projeto Integrado

Relatório Técnico

Arquive-se - Armazenamento de arquivos

Felipe da Silva Spinelli Cruz

Belo Horizonte,
Maio/2022.

Projeto Integrado – Arquitetura de Software Distribuído

Sumário

1. Introdução

A empresa **Arquive-se** há mais 5 anos oferece armazenamento de documentos e arquivos, tendo como principais clientes escritórios de contabilidade e advocacia. Estes clientes necessitam que seus documentos sejam armazenados por, no mínimo, 5 anos e que sejam enviados quando solicitados. Hoje o envio dos documentos é feito através do protocolo FTP, para seu servidor próprio, necessitando uma configuração no servidor de seus clientes.

Com o forte processo de transformação digital e diminuição de custos com infraestrutura em nuvem, a **Arquive-se** está com dificuldades em conquistar novos clientes, além de ter perdido alguns para concorrentes com soluções mais modernas e/ou mais baratas. Neste cenário, a empresa contratou a consultoria Tech-Tok, especializada em projetos de transformação digital e desenvolvimento de software.

Após estudo de mercado sobre a área de negócio da aplicação, os objetivos da solução proposta são:

- Plataforma baseada em planos;
- Camada gratuita para upload de arquivos, limitados por quantidade e/ou tamanho, para aumentar o alcance da marca;
- API pública, com endpoints protegidos através de mecanismo de autorização de usuários e aplicações clientes;
- O envio via FTP deve ser substituído por um serviço próprio, de fácil configuração que se comunique com a API pública;

2. *Cronograma do Trabalho*

A seguir é apresentado o cronograma proposto para as etapas deste trabalho.

Datas		Atividade / Tarefa	Produto / Resultado
De	Até		
13 / 08 / 22	13 / 08 / 22	1. Cronograma do Trabalho	Construção desta tabela
13 / 08 / 22	13 / 08 / 22	2. Contextualização do trabalho	Construção da contextualização deste projeto
13 / 08 / 22	13 / 08 / 22	3. Definição dos requisitos arquiteturais	Lista dos requisitos arquiteturais identificados
13 / 08 / 22	13 / 08 / 22	4. Definição dos requisitos funcionais	Lista dos requisitos funcionais identificados
13 / 08 / 22	13 / 08 / 22	5. Definição dos requisitos não-funcionais	Lista dos requisitos não-funcionais identificados
13 / 08 / 22	13 / 08 / 22	6. Definição dos mecanismos arquiteturais	Listas dos mecanismos arquiteturais identificados
13 / 08 / 22	13 / 08 / 22	7. Construção dos diagramas de contextos - Modelo C4	Diagrama de contexto criado no Draw.io e documentado
14 / 08 / 22	14 / 08 / 22	8. Revisão da Etapa 1	Documento etapa 1 revisado
14 / 08 / 22	14 / 08 / 22	9. Construção do vídeo de apresentação da etapa 1	Vídeo criado da etapa 1
14 / 08 / 22	14 / 08 / 22	11. Publicação no repositório Github etapa 1	Arquivos produzidos num repositório público no Github
15 / 08 / 22	21 / 08 / 22	12. Construção dos diagramas de contêineres	Diagrama de contêineres
22 / 08 / 22	28 / 08 / 22	13. Construção dos diagramas de componentes	Diagramas de componentes
29 / 08 / 22	04 / 09 / 22	14. Desenho dos Wireframes da POC	Protótipos de telas de baixa fidelidade
05 / 08 / 22	14 / 10 / 22	15. Código da aplicação	Aplicação com 3 requisitos implementados
14 / 10 / 22	14 / 10 / 22	16. Publicação no repositório Github etapa 2	Arquivos produzidos num repositório público no Github
15 / 10 / 22	14 / 12 / 22	17. Análise das abordagens arquiteturais	Seção do documento produzido
15 / 10 / 22	14 / 12 / 22	18. Cenários	Seção do documento produzido
15 / 10 / 22	14 / 12 / 22	19. Evidências da avaliação	Seção do documento produzido
15 / 10 / 22	14 / 12 / 22	20. Resultados obtidos	Seção do documento produzido
15 / 10 / 22	14 / 12 / 22	21. Avaliação crítica dos resultados	Seção do documento produzido
15 / 10 / 22	14 / 12 / 22	22. Conclusão	Seção do documento produzido

15 / 10 / 22	14 / 12 / 22	23. Construção do vídeo de apresentação da Etapa 3	Vídeo da etapa 3 disponível
15 / 10 / 22	14 / 12 / 22	24. Publicação no repositório Github Etapa 3	Arquivos produzidos num repositório público no Github

3. *Especificação Arquitetural da solução*

Esta seção apresenta a especificação básica da arquitetura da solução a ser desenvolvida, incluindo diagramas, restrições e requisitos definidos pelo autor, tal que permite visualizar a macro arquitetura da solução.

3.1 Restrições Arquiteturais

R1: O software deve ser desenvolvido em C#, utilizando o dotnet 6;
R2: As APIs devem seguir o padrão RESTful.
R3: As APIs devem ser serverless.
R3: A base de dados para fins de leitura deve ser não relacional.
R4: O portal deve ser desenvolvido utilizando Blazor WebApp
R5: O mecanismo de autorização deve ser utilizado de terceiros.

3.2 Requisitos Funcionais

ID	Descrição Resumida	Dificuldade (B/M/A)*	Prioridade (B/M/A)*
RF01	O sistema deve permitir upload de arquivos individuais	M	A
RF02	O portal deve permitir compartilhamento através de link com data de expiração e utilizável uma única vez	B	A
RF03	O portal deve permitir download de arquivo individual	B	A
RF04	O serviço deve permitir listagem de arquivos	B	M
RF05	O serviço deve permitir consulta de arquivos	M	M
RF06	O sistema deve permitir o auto cadastramento do usuário	A	M
RF07	O portal deve permitir compartilhamento, através de concessão de acesso	A	M
RF08	O sistema deve permitir upload de múltiplos arquivos, ou pasta	M	B
RF09	O sistema deve permitir download de múltiplos arquivos, compactados em um só	M	B
RF10	O sistema deve permitir a exclusão de arquivos	B	B
RF11	O portal deve permitir adesão/alteração de planos	M	M

RF12	O portal deve permitir acompanhamento do uso, através de um dashboard	B	M
RF13	O portal deve permitir a criação de credencial para o serviço de envio	M	B
RF14	O portal deve permitir download da fatura do plano	B	B
RF15	O serviço deve permitir criação de estruturas de pastas	M	B
RF16	O portal deve permitir importação de arquivos no Google Drive	A	B
RF17	O portal deve permitir importação de arquivos no Dropbox	A	B
RF18	O portal deve permitir criação de grupos de usuários	M	B
RF19	O serviço de envio deve permitir configurar as pastas para envio	M	M
RF20	O serviço de envio deve permitir configurar filtros por tipo de arquivos em cada pasta configurada	M	B
RF21	O serviço deve permitir definição de data de expiração do arquivo	B	B

*B=Baixa, M=Média, A=Alta.

3.3 Requisitos Não-funcionais

ID	Descrição	Prioridade B/M/A
RNF01	O sistema deve ser apresentar disponibilidade 24 X 7 X 365	A
RNF02	O sistema deve ter o design responsivo	M
RNF03	O sistema deve ter tolerância a falhas	A
RNF04	O sistema deve ser capaz de gerar alertas caso ocorra indisponibilidades.	A
RNF5	O sistema deve permitir monitorar a saúde dos microsserviços	M
RNF6	O sistema deve permitir escalonar os micro serviços horizontalmente	M

3.4 Mecanismos Arquiteturais

Análise	*Design*	Implementação
Persistência	SDK	CosmoDb SDK
Front end	WebServer	Blazor WebServer
Back end	Serverless Function	Azure Function
Autenticação/Autorização	IAM	Auth0
Armazenamento	Armazenamento Nuvem	Azure Blob Storage
Log do sistema	Horário, Nível do Log, nome do logger e a mensagem	Serilog
Teste de Software	Unit Test	xUnit
Deploy	CI/CD	Azure Devops
Redirecionamento de Rotas	API Gateway	Azure API Management
Monitoramento	Serverless	Azure Monitor
Documentação de API	Open API	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 (PoC) da plataforma Arquive-se na seção 5.

Para esta modelagem arquitetural optou-se por utilizar o modelo C4 para a documentação de arquitetura de software. Mais informações a respeito podem ser encontradas aqui: https://c4model.com/ e aqui: 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

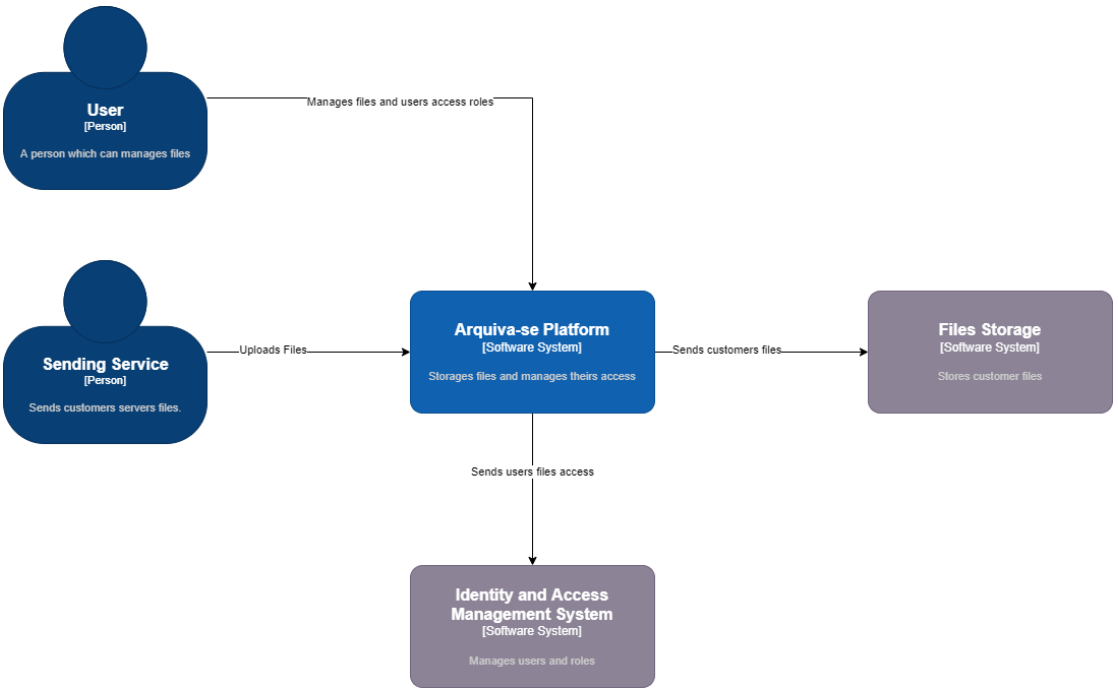


Figura 1 - Visão Geral da Solução.

A figura 1 mostra a especificação do diagrama geral da solução proposta, com todos seus principais módulos e suas interfaces.

Link do vídeo explicativo:

https://github.com/FelipeSpinelli/puc-minas-pos-tcc/blob/main/2022-08-15%2023-45-41.mkv

4.2 Diagrama de Container

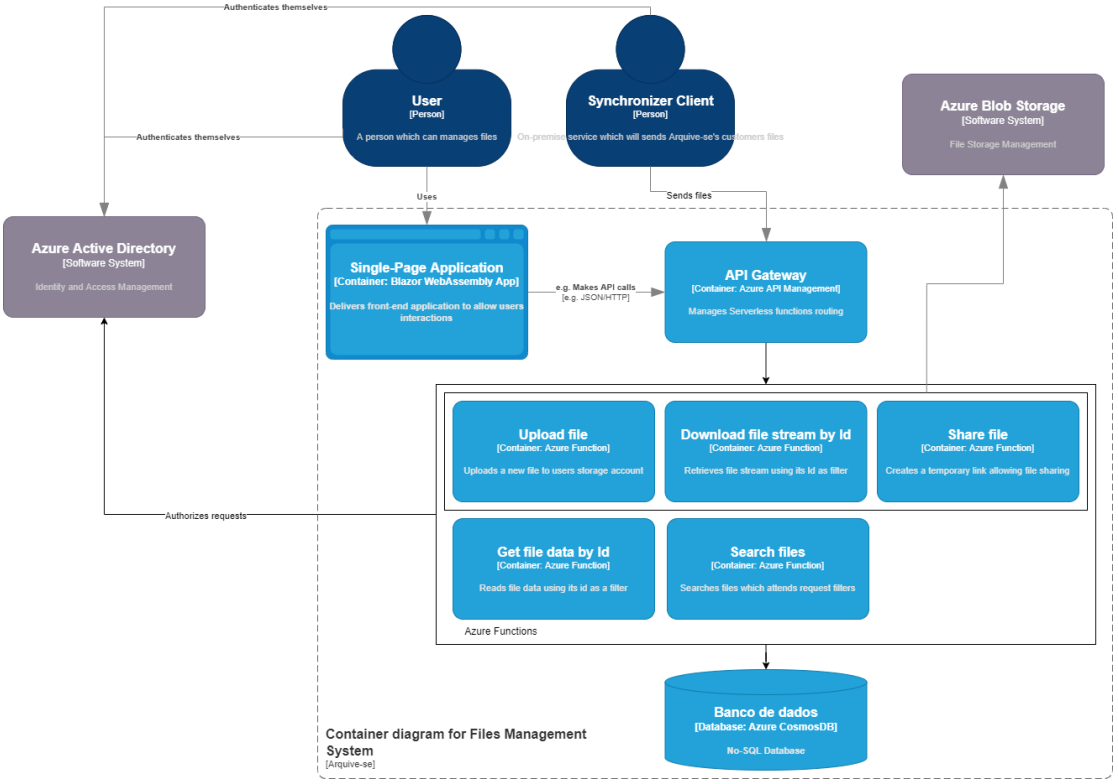


Figura 2 – Diagrama de container

A figura 2 apresenta os containers da aplicação (aplicativos, armazenamentos de dados, serviços Web, etc.) que compõem a plataforma e como estão distribuídos e organizados.

A aplicação terá dois tipos de usuários: Uma aplicação no servidor dos clientes, que irá automatizar o envio de arquivos para a plataforma Arquive-se, além de usuários usuários pontuais, que desejam compartilhar arquivos de forma rápida e gratuita.

A aplicação cliente se comunicará diretamente com a API, através de requisições HTTP autenticadas. Usuário comuns também deverão autenticar-se, mas utilizarão uma aplicação web para isso.

A API utilizada, tanto pela aplicação cliente, quanto pela webapp, será a mesma, sendo na realidade uma API Gateway, que fará o redirecionamento das requisições para funções *serverless*. Essas funções estarão responsáveis por realizar o acesso ao banco de dados (No-SQL Azure CosmosDB), gerenciar os arquivos através de um serviço de armazenamento em nuvem (Azure Blob Storage) e autorização as requisições com um serviço de gestão de acessos e identidade (Azure Active Directory).

4.3 Diagrama de Componentes

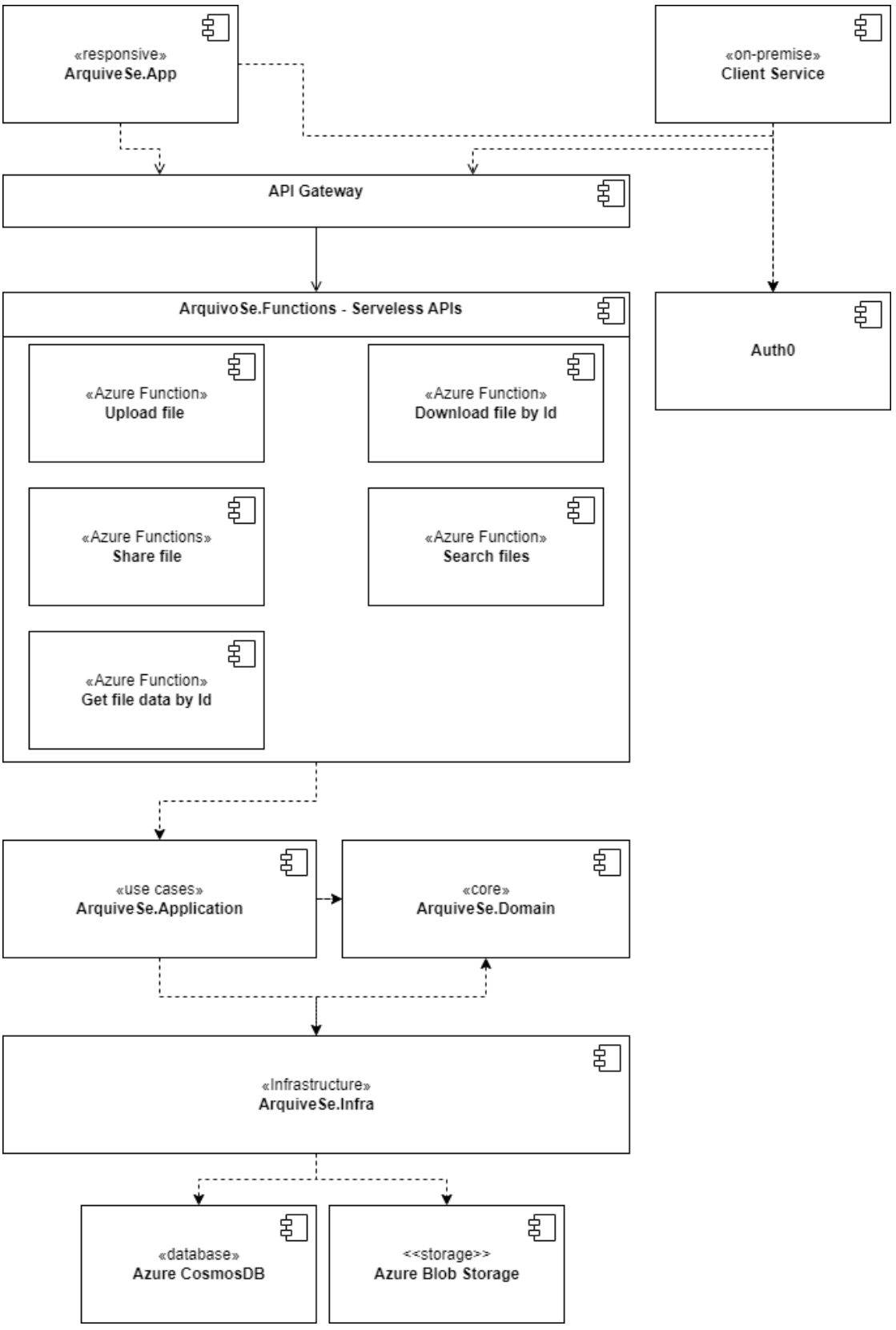


Figura 3 – Diagrama de Componentes

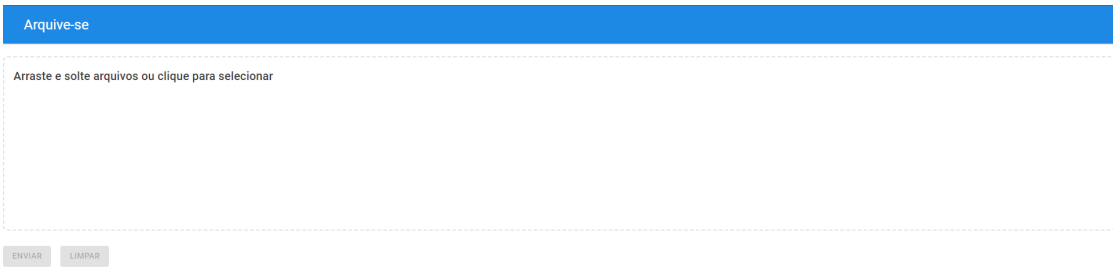
A figura 3 apresenta os componentes da aplicação e estes podem ser detalhados como:
- ArquiveSe.App - Aplicação web que disponibiliza a UI para utilização dos usuários;
- API Gateway - Serviço do Azure para proxy das requisições
- Client Service - Aplicação que será implantanda nos servidores dos clientes, para envio dos arquivos locais para a plataforma web;
- Auth0 - Serviço de terceiro para gestão de identidade, autentição e autorização;
- ArquiveSe.Functions - Projeto que contém as funções serverless;
- ArquiveSe.Application - Lib que contém a lógica do negócio, organizada em casos de uso;
- ArquiveSe.Domain - Lib que contém as regras do domínio da aplicação;
- ArquiveSe.Infra - Lib que contém as implementações das conexões com os serviços relacionados à infraestrutura (BD e armazenamento de arquivos);
- Azure CosmosDb - Serviço do Azure para armazenamento de dados No-SQL;
- Azure Blob Storage - Serviço do Azure para armazenamento de arquivos e afins;

5. Prova de Conceito (PoC)

Para validar as implementações propostas foram utilizados padrões de projeto e boas práticas como SOLID, Singleton e Repository;

5.1 Wireframe

5.1.1 - Tela de Upload



5.1.2 - Tela de Download

5.2 Código da Aplicação

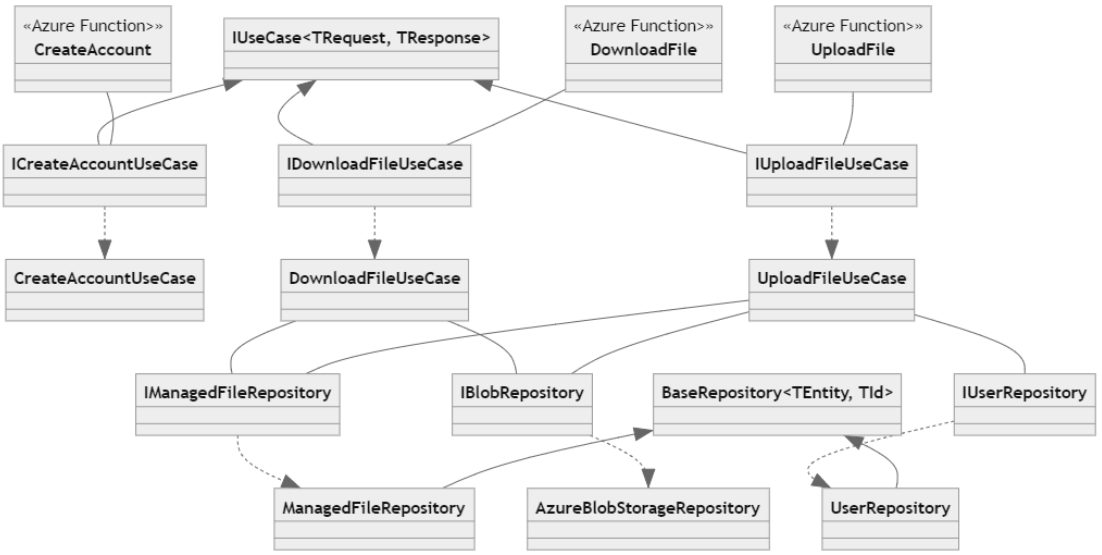


Figura 4 – Código da aplicação

O código da aplicação encontra-se no repositório abaixo:
https://github.com/FelipeSpinelli/puc-minas-pos-tcc

6. Avaliação da Arquitetura (ATAM)

A avaliação da arquitetura desenvolvida neste trabalho é abordada nesta seção visando avaliar se ela atende ao que foi solicitado pelo cliente, segundo o método ATAM.

6.1. Análise das abordagens arquiteturais

Apresente aqui um breve resumo das principais características da proposta arquitetural. Para isto, utilize o método Architecture Tradeoff Analysis Method (ATAM), no qual são utilizados cenários para fazer essa análise.

Exemplo:

Atributos de Quailidade	Cenários	Importância	Complexidade
Interoperabilidade	Cenário 1: O sistema deve se comunicar com sistemas de outras tecnologias.	A	M
Usabilidade	Cenário 2: O sistema deve prover boa usabilidade.	M	B
Manutenibilidade	Cenário 3: O sistema deve ter a manutenção facilitada.	M	M

6.2. Cenários

Mostre os cenários utilizados na realização dos testes da sua aplicação. Escolha cenários de testes que demonstrem os requisitos não funcionais (atributos de qualidade) sendo satisfeitos. Priorize os cenários para a avaliação segundo critérios quantitativos ou qualitativos.

Exemplos de cenários:

Cenário 1 - Interoperabilidade: Ao acessar a URL do serviço de informações gerenciais via HTTP GET, o mesmo deve retornar as informações no formato JSON.

Cenário 2 - Usabilidade: Ao navegar na tela, o sistema deve apresentar boa usabilidade. A navegação deve apresentar facilidade e o acesso as funcionalidades deve ser bem objetivo para a função que precisar ser realizada, o usuário deve ser capaz de efetuar uma compra em no máximo 5 minutos, assim garantindo a agilidade e a usabilidade para ficar de acordo com um dos requisitos não funcionais.

Cenário 3 - Manutenibilidade: Havendo a necessidade de alterar o gateway de pagamento somente será necessário fazer alteração no broker da funiconalidade de pagamento, facilitando a manutenção e os testes.

6.3. Evidências da Avaliação

Apresente as medidas registradas na coleta de dados. Para o que não for possível quantificar apresente uma justificativa baseada em evidências qualitativas que suportem o atendimento ao requisito não-funcional.

Atributo de Qualidade:	Interoperabilidade
Requisito de Qualidade:	O sistema deve se comunicar com outras tecnologias.
Preocupação:	
O sistema deve ter como resposta a uma requisição uma saída de fácil leitura por outro componente.	
Cenário(s):	
Cenário 1	
Ambiente:	
Sistema em operação normal	
Estímulo:	
O sistema de monitoramento envia uma requisição para o serviço REST do módulo de informações gerenciais.	
Mecanismo:	
Criar um serviço REST para atender às requisições do sistema de monitoramento	
Medida de resposta:	
Retornar os dados requisitados no formato JSON	
Considerações sobre a arquitetura:	
Riscos:	Alguma instabilidade na rede pode deixar a conexão lenta ou mesmo a perda de pacotes.
Pontos de Sensibilidade:	Não há
Tradeoff:	Não há

Acrescente imagens e descreva os testes realizados, de tal forma que se comprove a realização da avaliação.

Faça isto para todos os cenários apresentados no tópico 6.1.

6.4. Resultados Obtidos

Apresente os resultados da arquitetura produzida, indicando seus pontos fortes e suas limitações. A título de sugestão construa uma tabela apresentando esses resultados, como no exemplo que segue:

Requisitos Não Funcionais	Teste	Homologação
RNF01: O sistema deve ...	OK	OK
RNF02: O sistema deve ...	OK	N.A.
RNF03: ...	OK	N.A.

Obs: N.A.: não se aplica.

7. Avaliação Crítica dos Resultados

Apresente aqui, de forma resumida, os principais pontos positivos e negativos da arquitetura proposta. Adote uma postura crítica que permita entender as limitações arquiturais, incluindo os prós e contras das tecnologias. Você pode utilizar o formato textual ou produzir um quadro resumo.

Ex. de quadro resumo:

Ponto avaliado	**Descrição**
xxxxxxxxxxxxxxx	xxxxxxxxxxxxxxxxxxxx

8. Conclusão

Descreva, de forma sucinta, quais foram as lições aprendidas na execução do seu projeto arquitetural. Procure apresentá-las de tal forma que fiquem configurados os *trade-offs* da arquitetura produzida, como por exemplo, Segurança X Desempenho, Granularidade X Manutenibilidade, etc.

Aqui deve ser apresentado também tudo que se aprendeu com esse projeto, de modo a servir como ajuda para outros profissionais.

Também se faz necessário evidenciar as possibilidades de melhoria do projeto, caso se deseje dar continuidade a ele. Nesse sentido, indique possíveis ajustes ou melhorias arquiteturais, que possam vir a ser realizados futuramente.

Lições aprendidas (ex.):

1. xxxxxxxxxxxxxxxxx
2. xxxxxxxxxxxxxxxxx
3. xxxxxxxxxxxxxxxxx

Referências

Esse trabalho não requer revisão bibliográfica e, por isso, a inclusão das referências não é obrigatória, embora seja recomendada. Caso você deseje incluir referências empregadas em seu trabalho, relacione-as de acordo com as normas ABNT, disponíveis em www.pucminas.br, no *link*: http://portal.pucminas.br/imagedb/documento/DOC_DSC_NOME_ARQUI20160217102425-n.pdf.

Exemplo:

SOBRENOME DO AUTOR, Nome do autor. **Título do livro ou artigo.** Cidade: Editora, ano.