**PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS**

**PUC Minas Virtual**

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

Projeto Integrado

Relatório Técnico

Virtual Match – Rede Social de Relacionamento

Carlos Alberto Saraiva Junior

Leandro Alves Santos

Belo Horizonte

Março, 2022.

# Projeto Integrado – Arquitetura de Sistemas Distribuídos

**Sumário**

[1. Projeto Integrado – Arquitetura de Sistemas Distribuídos 3](#_Toc116682117)

[1. Introdução 4](#_Toc116682118)

[2. Cronograma do Trabalho 6](#_Toc116682119)

[3. Especificação Arquitetural da solução 7](#_Toc116682120)

[3.1 Restrições Arquiteturais 7](#_Toc116682121)

[3.2 Requisitos Funcionais 7](#_Toc116682122)

[3.3 Requisitos Não-funcionais 8](#_Toc116682123)

[3.4 Mecanismos Arquiteturais 9](#_Toc116682124)

[4. Modelagem Arquitetural 9](#_Toc116682125)

[4.1 Diagrama de Contexto 10](#_Toc116682126)

[4.2 Diagrama de Container 11](#_Toc116682127)

[4.3 Diagrama de Componente 12](#_Toc116682128)

[5. Prova de Conceito (PoC) 12](#_Toc116682129)

[5.1. Integrações entre Componentes 13](#_Toc116682130)

[5.2. Código da Aplicação 13](#_Toc116682131)

[6. Avaliação da Arquitetura (ATAM) 14](#_Toc116682132)

[a. 6.1. Análise das abordagens arquiteturais 14](#_Toc116682133)

[b. 6.2. Cenários 14](#_Toc116682134)

[c. 6.3. Evidências da Avaliação 15](#_Toc116682135)

[d. 6.4.  Resultados Obtidos 22](#_Toc116682136)

[7. Avaliação Crítica dos Resultados 23](#_Toc116682137)

[8. Conclusão 23](#_Toc116682138)

[Referências 24](#_Toc116682139)

## Introdução

É fato notório que a criação e popularização da internet foi um dos maiores marcos na evolução humana, tanto nas variadas formas de se fazer negócios, como nas variadas formas de se relacionar.

De acordo com Volpato, as 10 redes sociais mais utilizadas no Brasil em 2022 são:

1. **WhatsApp (165 mi)**
2. **YouTube (138 mi)**
3. **Instagram (122 mi)**
4. **Facebook (116 mi)**
5. **TikTok (73,5 mi)**
6. **Messenger (65,5 mi)**
7. **LinkedIn (56 mi)**
8. **Pinterest (30 mi)**
9. **Twitter (19 mi)**
10. **Snapchat (7,6 mi)**

Ainda segundo Volpato, “Redes sociais são estruturas formadas dentro ou fora da internet, por pessoas e organizações que se conectam a partir de interesses ou valores comuns. Muitos confundem com mídias sociais, porém as mídias são apenas mais uma forma de criar redes sociais, inclusive na internet.”

As redes sociais on-line têm grande impacto nas relações pessoais, de acordo com Silva: “Atualmente as pessoas passam mais tempo na internet do que se comunicando pessoalmente com outras pessoas. Refletir sobre os impactos das redes sociais na vida pessoal é importante para não se deixar levar pelo mundo virtual apenas e se isolar do mundo real”.

É claro que as redes sociais têm impacto positivo nas relações humanas, mas também impacto negativo, conforme Coelho, alguns impactos negativos são o Cyber Bullying, assédio e o impacto na produtividade, além destes ainda podemos colocar golpes financeiros, riscos à segurança pessoal e privacidade.

Segundo Holanda: “Uma coisa podemos dizer que é homônimo para todas as épocas, seja na primitiva ou na contemporânea, é a necessidade de se relacionar e garantir contatos com os grupos.

A diferença para os dias atuais é a velocidade e a forma de como isso tem acontecido, deixando de ser demorado e limitado para ser mais rápida e fácil. Tudo isso graças ao surgimento da internet e das mídias de relacionamento, fazendo com que a distância deixasse de ser um fator impeditivo.”

Pensando em relacionamentos humanos surgiu a ideia de desenvolvermos o Virtual Match. Virtual Match tem o objetivo de ser um aplicativo de relacionamento, onde pessoas com perfis validados, com interesse mútuo podem se conhecer, conversar, compartilhar interesses para, eventualmente, começarem um namoro na vida real.

## Cronograma do Trabalho

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

|  |  |  |  |
| --- | --- | --- | --- |
| **Datas** | | **Atividade / Tarefa** | **Produto / Resultado** |
| **De** | **Até** |
| 14/04/2022 | 14/04/2022 | 1. Cronograma do Trabalho | Construção desta tabela |
| 15/04/2022 | 15/04/2022 | 1. Contextualização do trabalho | Construção da contextualização deste projeto |
| 25/04/2022 | 25/04/2022 | 1. Definição dos requisitos Arquiteturais | Lista dos requisitos Arquiteturais identificados |
| 26/04/2022 | 27/04/2022 | 1. Definição dos requisitos Funcionais | Lista dos requisitos funcionais identificados |
| 28/04/2022 | 28/04/2022 | 1. Definição dos requisitos Não-funcionais | Lista dos requisitos Não-funcionais identificados |
| 04/05/2022 | 04/05/2022 | 1. Definição dos Mecanismos Arquiteturais | Lista dos Mecanismos Arquiteturais identificados |
| 05/05/2022 | 06/05/2022 | 1. Construção dos Diagramas de Contextos – Modelo C4 | Diagrama de contexto criado no Draw.io e documentado |
| 10/05/2022 | 10/05/2022 | 1. Revisão da Etapa 1 | Documento Etapa 1 revisado |
| 20/05/2022 | 21/05/2022 | 1. Construção do vídeo de apresentação da Etapa 1 | Vídeo criado da Etapa 1 |
| 20/05/2022 | 21/05/2022 | 1. Apresentação em PPT da Etapa 1 | PPT |
| 14/06/2022 | 14/06/2022 | 1. Publicação no repositório Github Etapa 1 | Arquivos produzidos no Github disponíveis abertamente |
| 10/07/2022 | 12/07/2022 | 1. Construção dos Diagramas de Contêineres | Diagramas de contêineres |
| 12/07/2022 | 15/07/2022 | 1. Construção dos Diagramas de Componentes | Diagramas de componentes |
| 15/07/2022 | 20/07/2022 | 1. Desenho dos Wireframes da POC | Protótipos de telas de baixa fidelidade |
| 20/06/2022 | 07/08/2022 | 1. Código da aplicação | Aplicação com 3 requisitos implementados |
| 10/08/2022 | 10/08/2022 | 16. Publicação no repositório Github Etapa 2 | Arquivos produzidos no Github disponíveis abertamente |
| 08/09/2022 | 15/09/2022 | 17. Análise das abordagens arquiteturais | Seção do documento produzido |
| 16/09/2022 | 17/09/2022 | 18. Cenários | Seção do documento produzido |
| 18/09/2022 | 22/09/2022 | 19. Evidências da avaliação | Seção do documento produzido |
| 25/09/2022 | 01/10/2022 | 20. Resultados obtidos | Seção do documento produzido |
| 02/10/2022 | 04/10/2022 | 21. Avaliação crítica dos resultados | Seção do documento produzido |
| 05/10/2022 | 07/10/2022 | 22. Conclusão | Seção do documento produzido |
| 08/10/2022 | 12/10/2022 | 23. Construção do vídeo de apresentação da Etapa 3 | Vídeo da etapa 3 disponível |
| 14/10/2022 | 14/10/2022 | 24. Publicação no repositório Github Etapa 3 | Arquivos produzidos no Github disponíveis abertamente |

## 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 permitem visualizar a macro arquitetura da solução.

## Restrições Arquiteturais

Os Requisitos Arquiteturais são todos os requisitos, sejam eles Funcionais ou Não-Funcionais que têm **impacto direto** sobre a Arquitetura do Sistema. Dessa forma, o Arquiteto precisa analisar os requisitos do sistema identificando algumas propriedades e então “filtrando” os Requisitos Arquiteturais. A lista a seguir apresenta os requisitos arquiteturais que foram identificados para implementação inicial da plataforma.

R1: O software deve ser desenvolvido utilizando C# e .Net Core;

R2: As APIs devem seguir o padrão RESTful;

R3: A base de dados a ser utilizada deve ser o Postgres;

R4: Github deve ser utilizado para versionamento do código fonte;

R5: A autenticação e autorização será feita via uma API interna que gera um token JWT;

## Requisitos Funcionais

Os Requisitos Funcionais são todos aqueles que estão associados às funcionalidades que ditam **o que o sistema** deve fazer. A lista a seguir apresenta os requisitos funcionais identificados para o desenvolvimento inicial da plataforma.

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Descrição Resumida** | **Dificuldade (B/M/A)\*** | **Prioridade**  **(B/M/A)\*** |
| RF01 | O sistema deve permitir aos usuários criar credenciais de acesso na plataforma. | A | A |
| RF02 | O sistema deve permitir aos usuários atualizar suas credenciais de acesso criadas previamente na plataforma | B | B |
| RF03 | O sistema deve permitir que os usuários acessem o sistema de acordo com suas credenciais | A | A |
| RF04 | O sistema deve controlar os acessos através das credenciais para não permitir que um usuário veja informações de outro. | M | A |
| RF05 | O sistema deve permitir que um usuário encontre outro usuário conforme suas preferências. | M | A |
| RF06 | O sistema deve permitir que o usuário favorite outro usuário | M | A |
| RF07 | O sistema deve permitir o cadastro de perfil pessoal para cada usuário. | B | A |
| RF08 | O sistema deve permitir aos usuários conversarem entre si através de um chat. | M | B |
| RF09 | O sistema deve permitir que um usuário possa enviar uma mensagem privada para outro. | M | A |

\*B=Baixa, M=Média, A=Alta.

## Requisitos Não-funcionais

Os Requisitos Não-Funcionais estão associados às restrições de funcionalidades que **ditam como** o sistema deve fazer.A lista a seguir apresenta os requisitos funcionais identificados para o desenvolvimento inicial da plataforma.

|  |  |  |
| --- | --- | --- |
| **ID** | **Descrição** | **Prioridade**  **B/M/A** |
| RNF01 | A plataforma deve ter autenticação e autorização próprias por meio de API. | A |
| RNF02 | A plataforma deve funcionar o mais próximo possível de 24 x 7. | A |
| RNF03 | O frontend da plataforma será desenvolvido somente para web, inicialmente. | M |
| RNF04 | A plataforma deve funcionar nos browsers, Chrome e Edge (PCs). | B |
| RNF05 | A plataforma deve ser hospedada na nuvem. | A |
| RNF06 | O código fonte será armazenado no Github | A |
| RNF07 | O sistema deve armazenar as imagens no Cloudinary. | A |

## Mecanismos Arquiteturais

Os mecanismos arquiteturais representam conceitos técnicos fundamentais que serão padronizados por toda a solução. Eles são refinados durante o projeto em três estados, representados pelas três categorias de Mecanismos Arquiteturais:

* Mecanismo de Análise, que dá ao mecanismo um nome, uma descrição resumida e alguns atributos básicos derivados dos requisitos do projeto.
* Mecanismos de Design, que são mais concretos e assumem alguns detalhes do ambiente de implementação.
* Mecanismo de Implementação, que especifica a exata implementação de cada mecanismo.

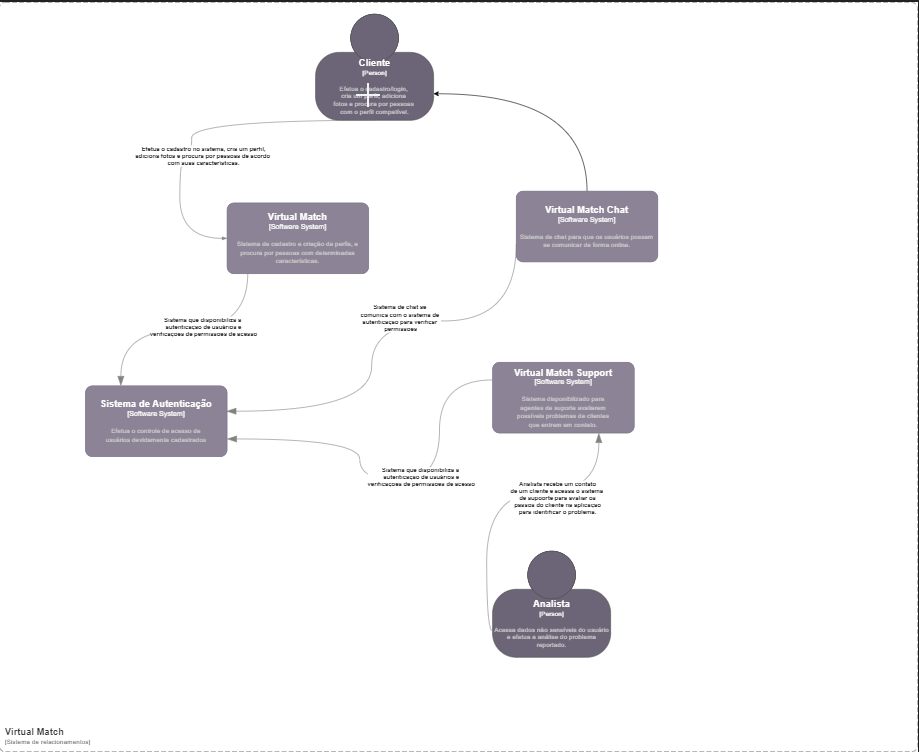
|  |  |  |
| --- | --- | --- |
| **Análise** | ***Design*** | **Implementação** |
| Persistência | ORM | EntityFramework |
| Persistência | ORM | PostgreSQL |
| Back end | REST API | Json |
| Back end | REST API | C# |
| Front end | MVVM | Angular |
| Front end | Navegador Web | Chrome like |
| Autenticação | JWT + Cripto | Desenvolvimento Interno |
| Autorização | JWT + Cripto | Desenvolvimento Interno |
| Deploy | Manual | Manual |

## 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) do aplicativo Virtual Match na seção 5.

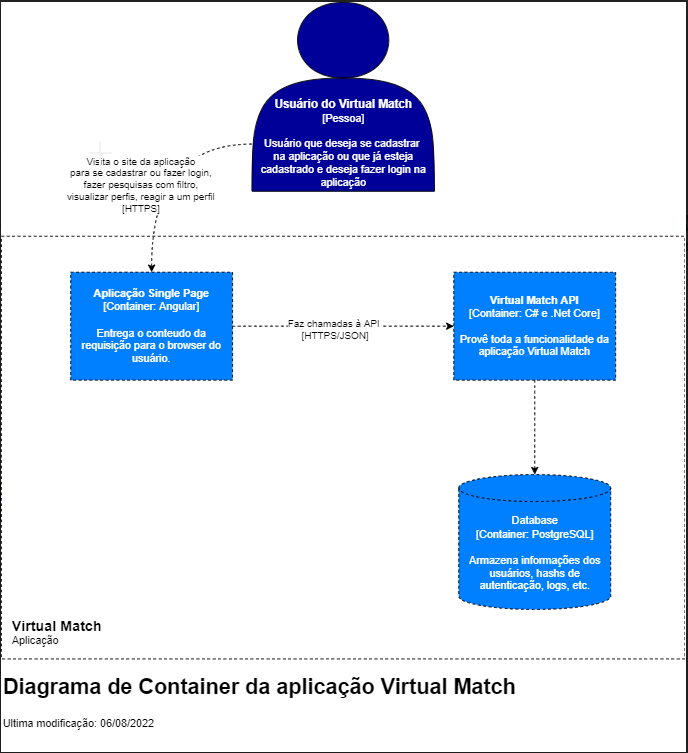
Para esta modelagem arquitetural optou-se por utilizar o modelo C4 para a documentação de arquitetura de software. Mais informações a respeito podem ser encontradas aqui: <https://c4model.com/> e aqui: <https://www.infoq.com/br/articles/C4-architecture-model/>. Dos quatro níveis que compõem o modelo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

## Diagrama de Contexto

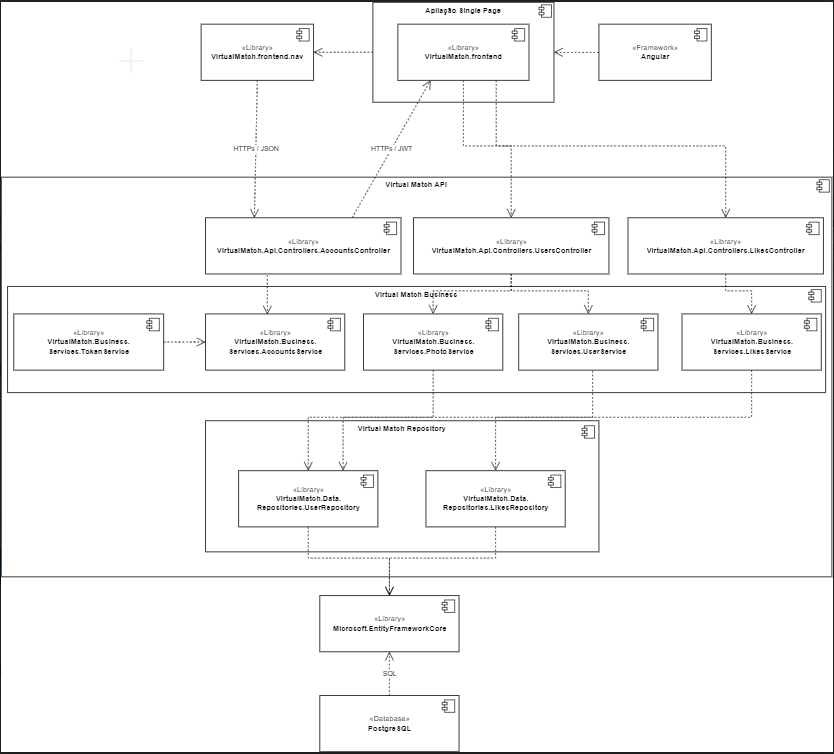
 ***Figura 1 - Visão Geral do Virtual Match.***

A figura 1 mostra a especificação do diagrama geral da solução proposta de um aplicativo que facilita o relacionamento entre usuários da plataforma. O usuário autentica no sistema e cadastrar seu perfil e suas preferências e pode encontrar outros usuários que compartilhem as mesmas preferências.

## Diagrama de Container

**

## Diagrama de Componente

**

## Prova de Conceito (PoC)

Os componentes arquiteturais validados nesta PoC foram:

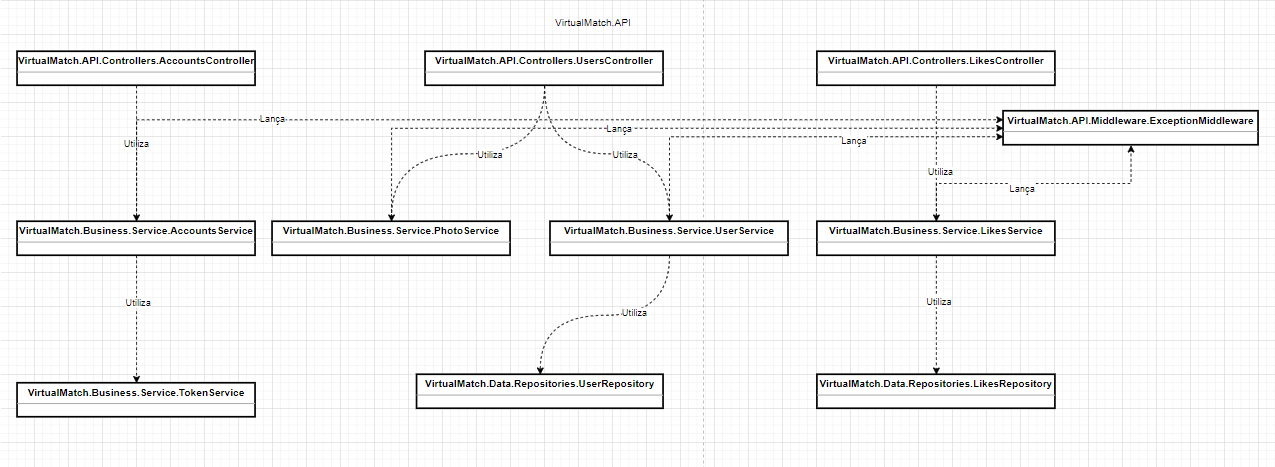
1. Login (Autenticação) e controle de usuário (Autorização) no sistema através de API desenvolvida utilizando JWT.
2. Implementação da solução em nuvem para disponibilidade próxima de 24x7.
3. Navegabilidade e usabilidade da aplicação nos browsers Chrome e Edge (PCs).
4. Integração entre frontend e backend utilizando Web API Rest.
5. Armazenamento e consulta de dados utilizando Entity Framework Core.
6. Armazenamento e consulta de imagens utilizando Cloudinary.

## Integrações entre Componentes

O protótipo da aplicação foi desenvolvido utilizando a ferramenta online Uizard e pode ser encontrado no seguinte caminho:

<https://app.uizard.io/p/2490d40e>

## 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: Fazem 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.

- Repositories: Responsáveis por fazer a persistência de dados.

- ExceptionMiddleware: Middleware implementado para controlar as exceções lançadas pela API e adequar o retorno utilizando http status.

Link Aplicação: <http://virtualmatch.herokuapp.com/>

Repositório código fonte Frontend:

<https://github.com/carlosaraiva/VirtualMatch/tree/main/src/frontend>

Repositório do código fonte Backend:

<https://github.com/carlosaraiva/VirtualMatch/tree/main/src/VirtualMatch.Api>

## 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 Qualidade** | **Cenários** | **Importância** | **Complexidade** |
| Autenticação | Cenário 1: A plataforma deve ter autenticação e autorização próprias por meio de API. | A | A |
| Disponibilidade | Cenário 2: A plataforma deve funcionar o mais próximo possível de 24 x 7. | A | B |
| Compatibilidade 1 | Cenário 3: O frontend da plataforma será desenvolvido somente para web, inicialmente. | M | M |
| Compatibilidade 2 | Cenário 4: A plataforma deve funcionar nos browsers, Chrome e Edge (PCs). | B | M |
| Hospedagem | Cenário 5: A plataforma deve ser hospedada na nuvem. | A | B |
| Manutenibilidade | Cenário 6: O código fonte será armazenado no Github. | A | B |
| Armazenamento | Cenário 7: O sistema deve armazenar as imagens no Cloudinary. | A | 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.

*Cenário 1 – Autenticação: Criação de novas contas e autenticação são realizadas através de um endpoint específico utilizando criptografía HMACSHA512.*

*Cenário 2 – Disponibilidade: Aplicação hospedada na nuvem (heroku) para garantir o máximo de disponibilidade*

*.*

*Cenário 3 – Compatibilidade 1: Primeira versão da aplicação disponível apenas para navegadores web. Próxima versão será disponibilizada para browsers em smartphones*

*Cenário 4 – Compatibilidade 2: Aplicação funcional e com o mesmo formato de exibição de conteúdo tanto para Chrome e Edge.*

*Cenário 5 – Hospedagem: A aplicação (backend e Frontend) foram hospedadas no Heroku de modo.*

*Cenário 6 – O código fonte foi disponibilizado no github. É possível fazer a publicação diretamente da branch main para o heroku.*

*Cenário 7 – Armazenamento: O armazenamento de fotos dos usuários será feito no Cloudinary. A empresa Cloudinary oferece o serviço de armazenamento e consulta de imagens através de API.*

## 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: | Autenticação |
| Requisito de Qualidade: | A plataforma deve ter autenticação e autorização próprias por meio de API |
| Preocupação: | |
| O sistema deve efetuar a autenticação por meio de API própria, sem confiar essa parte a API de terceiros. | |
| Cenário(s): | |
| Cenário 1 | |
| Ambiente: | |
| Sistema em operação normal | |
| Estímulo: | |
| Antes de visualizar quaisquer informações retornadas do banco de dados, os usuários devem estar autenticados. | |
| Mecanismo: | |
| Criar um serviço REST para atender às requisições de autenticação | |
| Medida de resposta: | |
| Retornar os dados requisitados no formato JSON | |
| Considerações sobre a arquitetura: | |
| Riscos: | Instabilidade de conexão e/ou instabilidade nos servidores heroku |
| Pontos de Sensibilidade: | Não há |
| Tradeoff: | Não há |

|  |  |
| --- | --- |
| Atributo de Qualidade: | Disponibilidade |
| Requisito de Qualidade: | A plataforma deve funcionar o mais próximo possível de 24 x 7 |
| Preocupação: | |
| O sistema deve ser implantado na nuvem para garantir que estará em funcionamento o mais próximo possível de 24 x 7 | |
| Cenário(s): | |
| Cenário 2 | |
| Ambiente: | |
| Sistema em operação normal | |
| Estímulo: | |
| Usuários utilizando o sistema | |
| Mecanismo: | |
| Acesso em diferentes horários | |
| Medida de resposta: | |
| Implantar sistema em infraestrutura resiliente a falhas. Para essa prova e conceito, optamos por utilizar o Heroku. | |
| Considerações sobre a arquitetura: | |
| Riscos: | Instabilidade de conexão e/ou instabilidade nos servidores heroku |
| Pontos de Sensibilidade: | Não há |
| Tradeoff: | Não há |

|  |  |
| --- | --- |
| Atributo de Qualidade: | Compatibilidade 1 |
| Requisito de Qualidade: | O frontend da plataforma será desenvolvido somente para web, inicialmente. |
| Preocupação: | |
| O frontend da aplicação foi inicialmente disponibilizado para os principais web browsers da atualidade. | |
| Cenário(s): | |
| Cenário 3 | |
| Ambiente: | |
| Sistema em operação normal | |
| Estímulo: | |
| O usuário, independente do browser, consegue visualizar a informação bem formatada na tela. | |
| Mecanismo: | |
| Adequar código do front-end para diferentes navegadores. | |
| Medida de resposta: | |
| Informação é apresentada corretamente | |
| 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 |

|  |  |
| --- | --- |
| Atributo de Qualidade: | Compatibilidade 2 |
| Requisito de Qualidade: | A plataforma deve funcionar nos browsers, Chrome e Edge (PCs). |
| Preocupação: | |
| O frontend da aplicação foi inicialmente disponibilizado para os principais web browsers da atualidade. | |
| Cenário(s): | |
| Cenário 4 | |
| Ambiente: | |
| Sistema em operação normal | |
| Estímulo: | |
| O usuário, independente do browser, consegue visualizar a informação bem formatada na tela. | |
| Mecanismo: | |
| Adequar código do front-end para diferentes navegadores. | |
| Medida de resposta: | |
| Informação é apresentada igualmente para os browsers suportados  Chrome    Edge | |
| Considerações sobre a arquitetura: | |
| Riscos: | Atualização de algum dos browsers suportados |
| Pontos de Sensibilidade: | Não há |
| Tradeoff: | Não há |

|  |  |
| --- | --- |
| Atributo de Qualidade: | Hospedagem |
| Requisito de Qualidade: | A plataforma deve ser hospedada na nuvem. |
| Preocupação: | |
| O sistema deve ser implantado na nuvem para garantir que estará em funcionamento o mais próximo possível de 24 x 7 | |
| Cenário(s): | |
| Cenário 5 | |
| Ambiente: | |
| Sistema em operação normal | |
| Estímulo: | |
| Garantir que a aplicação ficará disponível o mais próximo possível de 24 x 7 | |
| Mecanismo: | |
| Acesso em diferentes horários. | |
| Medida de resposta: | |
| Implantar sistema em infraestrutura resiliente a falhas. Para essa prova e conceito, optamos por utilizar o Heroku. | |
| 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 |

|  |  |
| --- | --- |
| Atributo de Qualidade: | Manutenibilidade |
| Requisito de Qualidade: | O código fonte será armazenado no Github. |
| Preocupação: | |
| O código fonte deve estar disponível para que todos os desenvolvedores possam fazer manutenções e novos desenvolvimentos | |
| Cenário(s): | |
| Cenário 6 | |
| Ambiente: | |
| Sistema em operação normal | |
| Estímulo: | |
| Manutenção que possa ser feita por qualquer desenvolvedor autorizado | |
| Mecanismo: | |
| Código fonte disponibilizado no Github | |
| Medida de resposta: | |
| Código fonte no github | |
| Considerações sobre a arquitetura: | |
| Riscos: | Não há |
| Pontos de Sensibilidade: | Não há |
| Tradeoff: | Não há |

|  |  |
| --- | --- |
| Atributo de Qualidade: | Armazenamento |
| Requisito de Qualidade: | O sistema deve armazenar as imagens no Cloudinary. |
| Preocupação: | |
| O cloudinary possui uma API robusta para trabalhar com imagens, isso facilita a implementação de novas features focadas em tratamento de imagem e possibilidade de inclusão de video. | |
| Cenário(s): | |
| Cenário 7 | |
| Ambiente: | |
| Sistema em operação normal | |
| Estímulo: | |
| Inclusão e consulta de imagens através da API do Virtual Match efetua chamadas para a API do Cloudinary | |
| Mecanismo: | |
| Conta no Cloudinary | |
| Medida de resposta: | |
|  | |
| Considerações sobre a arquitetura: | |
| Riscos: | Instabilidade nos servidores do Cloudinary |
| Pontos de Sensibilidade: | Não há |
| Tradeoff: | Não há |

## 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: A plataforma deve ter autenticação e autorização próprias por meio de API. | OK | OK |
| RNF02: A plataforma deve funcionar o mais próximo possível de 24 x 7. | OK | N.A. |
| RNF03: O frontend da plataforma será desenvolvido somente para web, inicialmente. | OK | OK |
| RNF04: A plataforma deve funcionar nos browsers, Chrome e Edge (PCs). | OK | OK |
| RNF05: A plataforma deve ser hospedada na nuvem. | OK | N.A. |
| RNF06: O código fonte será armazenado no Github | OK | N.A. |
| RNF07: O sistema deve armazenar as imagens no Cloudinary. | OK | N.A. |

*Obs: N.A.: não se aplica.*

## Avaliação Crítica dos Resultados

|  |  |
| --- | --- |
| **Ponto avaliado** | **Descrição** |
| Autenticação | O sistema de autenticação desenvolvido permite cadastrar e entrar na aplicação, como é suposto, mas é inflexível, poderíamos utilizar OAuth2 e permitir integrar com o login de outras plataformas, como Microsoft, Google, Facebook, etc.... |
| Autorização | O sistema de autorização cumpre o seu papel permitindo o acesso somente a quem deve acessar. |
| Disponibilidade | O sistema foi bem desenvolvido para a nuvem e, no futuro, poderemos colocá-lo em contêineres para aumentar ou diminuir as instancias conforme a necessidade. |
| API | A API foi bem desenhada e formatada. Poderia estar atrás de um API Gateway para melhor controle dos acessos. |
| Segurança | Por ora, contamos apenas com a infraestrutura de segurança em nuvem (Firewall). Este é um aspecto que pode ser melhorado futuramente. |
| Armazenamento de dados | Utilizamos o Entity Framework Core 6 e PostgreSql para armazenar os dados e a resposta entre a aplicação e a base de dados é satisfatória. Poderíamos implementar uma factory de base de dados para permitir trocar de base de dados facilmente. |
| Modelo de dados | O modelo de dados está bem definido o que permite evolução futura. |
| Frontend | A navegação e a interface estão boas e intuitivas, porém, pode ser melhorado utilizando técnicas de UI/UX. Como utilizamos Angular para desenvolver o frontend, tudo está em componentes o que permite fácil manutenção e evolução da interface visual. |

## Conclusão

Nosso objetivo com este trabalho foi criar e apresentar uma aplicação onde seja possível pessoas se relacionarem. Para atingir este objetivo priorizamos a facilidade de utilização, manutenção do código e deixar o código o mais preparado possível para futuras evoluções e intervenções. Além disso queríamos a aplicação disponível 24x7 em qualquer lugar do mundo, por isto optamos por implantar a aplicação em nuvem. As camadas desenvolvidas atendem a padrões de arquitetura modernos e nos permite facilmente alterar ou melhorar o código. Para nós o desempenho, pelo menos neste momento inicial, foi mais importante que a segurança.

Os maiores aprendizados que tivemos foi o desenho da arquitetura utilizando C4 Model, medir os pontos chave da aplicação utilizando ATAM e ter feito o protótipo antes de qualquer desenvolvimento nos ajudou a perceber o que queríamos da aplicação antes mesmo da primeira linha de código ser escrita.

Conforme citado acima, alguns pontos que poderão ser melhorados no futuro são:

1. Melhorar o design e a navegação da aplicação
2. Permitir autenticação de terceiros como Microsoft, Google, Facebook, etc.
3. Implementar a aplicação em container para permitir escalabilidade
4. Permitir a utilização da aplicação em outras línguas
5. Criar factory de base de dados para alternar facilmente para outra plataforma
6. Implementar outros componentes de segurança

## Referências

Coelho, André M. **Qual o impacto das redes sociais na sociedade?** Sem data. Disponível em <https://www.pontorh.com.br/qual-impacto-redes-sociais-sociedade/>. Acesso em 24 de maio de 2022.

Holanda, Isabel. **A influência das redes sociais na comunicação humana.** 03 de setembro de 2021. Disponível em <https://blog.fortestecnologia.com.br/tecnologia-e-inovacao/a-influencia-das-redes-sociais/>. Acesso em 05 de junho de 2022.

Martins da Silva, Maria do Rosário. **As redes sociais e seus impactos nas relações pessoais**. 19 de dezembro de 2015. Disponível em <https://administradores.com.br/artigos/as-redes-sociais-e-seus-impactos-nas-relacoes-pessoais>. Acesso em 24 de maio de 2022.

Volpato, Bruno. **Ranking: as redes sociais mais usadas no Brasil e no mundo em 2022, com insights e materiais gratuitos**. 23 de maio de 2022. Disponível em <https://resultadosdigitais.com.br/marketing/redes-sociais-mais-usadas-no-brasil/>. Acesso em 24 de maio de 2022.