

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS

PUC Minas Virtual

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

Projeto Integrado

Relatório Técnico

Investidor Tools

Rafael dos Santos Pereira

São Paulo
Outubro 2022

Projeto Integrado – Arquitetura de Software Distribuído

Sumário

Projeto Integrado – Arquitetura de Software Distribuído	2
1. Introdução	3
2. Cronograma do Trabalho	5
3. Especificação Arquitetural da solução	7
3.1 Restrições Arquiteturais	7
3.2 Requisitos Funcionais	8
3.3 Requisitos Não-funcionais	9
3.4 Mecanismos Arquiteturais	10
4. Modelagem Arquitetural	11
4.1 Diagrama de Contexto	11
4.2 Diagrama de Container	12
4.3 Diagrama de Componentes	13
5. Prova de Conceito (PoC)	15
5.1 Integrações entre Componentes	15
5.2 Código da Aplicação	18
6. Avaliação da Arquitetura (ATAM)	22
6.1. Análise das abordagens arquiteturais	22
6.2. Cenários	23
6.3. Evidências da Avaliação	23
6.4. Resultados Obtidos	28
7. Avaliação Crítica dos Resultados	29
8. Conclusão	30
Referências	31

1. Introdução

O mercado de ações está cada vez mais se popularizando no Brasil, e mês após mês, o número de investidores em ações cresce. De acordo com a B3 (Bolsa de Valores Brasileira), em janeiro de 2022, foi atingida a marca de 5 milhões de contas de pessoas físicas abertas em corretoras no Brasil. Crescimento de 56% em comparação com o número de dezembro de 2020. Esse número é dividido entre 1.2 milhão de contas femininas e 3.8 milhões de contas abertas por pessoas do sexo masculino. Outros dados mostram que os primeiros investimentos, estão sendo feitos com valores cada vez mais baixos. Mostrando que o investimento em ações está começando a ser realizado, inclusive por pessoas com renda mais baixa.

Atualmente existem cerca de 400 empresas negociadas na bolsa de valores brasileira, além das empresas estrangeiras negociadas através de BDRs (Brazilian Depositary Receipt). Isso faz com que muitas vezes o investidor iniciante não sabia como escolher uma empresa para investir, o que pode fazer com que ele acabe entrando em um investimento, sem conhecer a empresa e os riscos envolvidos no setor em que a empresa atua e os próprios riscos inerentes a esta empresa. Devido a não ter tido acesso a conteúdo de qualidade sobre como escolher as empresas nas quais investir, quais indicadores analisar e as diversas possibilidades de tipos de investimentos disponíveis no mercado Brasileiro e internacional.

Atualmente, tratando-se de conteúdo sobre análise de empresas para investimentos a longo prazo, verificamos que existem alguns portais em que o investidor pode analisar múltiplos das empresas relacionados a valor de mercado, crescimento, dividendos, entre outros indicadores, como também, outros com dados referentes a números financeiros. Em ambos os casos os números são disponibilizados e o próprio investidor deve analisar e tomar a decisão de qual empresa escolher. Porém, somente esses dados, não são suficientes para se ter uma segurança de que o investidor está fazendo uma boa escolha. Esses portais não entram no detalhe do negócio da empresa, setor, vantagens competitivas, etc.

Com esse foco, de entrar no detalhe das empresas, existem as casas de Análise, como por exemplo a Suno Research, Nord e Empiricus. Essas casas de análise, funcionam no modelo de negócio de planos de assinatura. Geralmente o investidor escolhe um tipo de investimento como por exemplo Ações que pagam bons dividendos ou SmallCaps e precisam pagar uma assinatura para cada tipo de carteira desejada.

Os relatórios das casas de análise costumam ser bastante completos, porém, caso o investidor queira ter acesso a diversos tipos de relatórios, precisa pagar mais de uma assinatura ou então uma assinatura mais cara que contemple os relatórios que ele deseja.

Com base nesse problema, surgiu a ideia da criação de um portal chamado Investidor Tools, que seria como um marketplace de relatórios de empresas, onde seriam disponibilizados relatórios das casas de análise, além de relatórios de especialistas certificados, não vinculados a casas de análises.

Com isso, ao invés do investidor ter que pagar por diversas assinaturas ou uma assinatura cara, ele poderia escolher os relatórios das empresas em que tem interesse em investir, para ajudá-lo na tomada de decisão. Comprando os relatórios avulsos, sem a necessidade de ficar preso a uma assinatura. Podendo inclusive, comprar relatórios a respeito da mesma empresa, feitos por diferentes analistas, para ter diferentes pontos de vista.

Por se tratar de uma ferramenta ainda não existente no mercado e que pode atender a dor da dificuldade de se analisar a fundo uma empresa, se justifica o objetivo deste trabalho que é a apresentação e definição arquitetural da plataforma, para que se torne um portal no dia a dia do investidor.

Foram definidos 3 objetivos principais para a primeira versão da plataforma, são eles:

- Ser uma plataforma de negociação de relatórios, e tomadas de decisão sobre investimentos no mercado financeiro;
- Ter uma interface amigável e moderna;
- Ser segura, tolerante a falhas e disponível mesmo com picos de acesso simultâneos.

Com estes objetivos definidos, serão apresentados em seguida os requisitos, as definições de arquitetura, interfaces e versão inicial da plataforma.

2. Cronograma do Trabalho

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

Datas		Atividade / Tarefa	Produto / Resultado
De	Até		
20 / 02 / 22	21 / 02 / 22	1.Elaboração do Cronograma	Construção do Cronograma de acordo com datas de entrega
26 / 02 / 22	27 / 02 / 22	2. Introdução e Contexto	Apresentação inicial do trabalho
05 / 03 / 22	06 / 03 / 22	3. Restrições Arquiteturais	Definição das restrições arquiteturais
12 / 03 / 22	13 / 03 / 22	4. Requisitos Funcionais	Definição dos requisitos funcionais
19 / 03 / 22	20 / 03 / 22	5. Requisitos não-Funcionais	Definição dos requisitos não-funcionais
23 / 03 / 22	23 / 03 / 22	6. Mecanismos Arquiteturais	Definição dos mecanismos arquiteturais
26 / 03 / 22	27 / 03 / 22	7. Diagrama de Contexto	Apresentação do diagrama de contexto
02 / 04 / 22	02 / 04 / 22	8. Criação de Apresentação para entrega da Etapa 1	PPT de apresentação finalizado
03 / 04 / 22	03 / 04 / 22	9. Gravação de vídeo de Apresentação para entrega da Etapa 1	Vídeo de apresentação gravado
14 / 04 / 22	14 / 04 / 22	10. Entrega da Etapa 1	Etapa 1 Entregue
23 / 04 / 22	23 / 04 / 22	11. Diagrama de Contêineres	Apresentação do diagrama de Contêineres
24 / 04 / 22	24 / 04 / 22	12. Diagrama de Componentes	Apresentação do diagrama de Componentes
30 / 04 / 22	01 / 05 / 22	13. Desenvolver Wireframe	Desenho do wireframe
07 / 05 / 22	08 / 05 / 22	14. Gravação de vídeo de Apresentação do Wireframe	Vídeo de apresentação gravado
14 / 05 / 22	12 / 06 / 22	15. Criação do código e publicação no repositório	Software desenvolvido
13 / 06 / 22	13 / 06 / 22	16. Entrega da Etapa 2	Etapa 2 Entregue
16 / 07 / 22	16 / 07 / 22	17. Análise das abordagens arquiteturais	Abordagens arquiteturais
23 / 07 / 22	31 / 07 / 22	18. Cenários	Apresentação dos cenários
06 / 08 / 22	07 / 08 / 22	19. Evidências da avaliação	Descrição das evidências obtidas
13 / 08 / 22	14 / 08 / 22	20. Resultados obtidos	Apresentação dos resultados
20 / 08 / 22	21 / 08 / 22	21. Avaliação críticas dos resultados	Avaliação dos resultados
27 / 08 / 22	28 / 08 / 22	22. Conclusão	Conclusões do projeto desenvolvido

Investidor Tools

03 / 09 / 22	04 / 09 / 22	23. Gravação de vídeo de apresentação do Trabalho	Apresentação final do projeto
--------------	--------------	---	-------------------------------

3. Especificação Arquitetural da solução

Nesta seção serão apresentadas as definições principais da arquitetura da plataforma a ser desenvolvida. Incluindo diagramas, restrições e requisitos definidos para a implementação do projeto, que permitirão a visualização da macro arquitetura da solução.

3.1 Restrições Arquiteturais

Existem características e restrições que limitam o desenvolvimento e manutenção de projetos. Elas não são consideradas requisitos, pois, não constituem funcionalidade ou necessidades a serem satisfeitas. Porém, são limitações que possuem forte impacto sobre a arquitetura do Sistema. Com essa visão, listaremos a seguir as restrições arquiteturais encontradas para o desenvolvimento e implementação da aplicação descrita nesse trabalho.

ID	Descrição Resumida
RA01	O Back-end do sistema deve ser desenvolvido utilizando a linguagem Java em versão igual ou superior a 1.8 e o framework Spring Boot.
RA02	O Front-end deve ser desenvolvido utilizando React.
RA03	As comunicações devem sempre priorizar o padrão REST.
RA04	Como banco de dados deve ser utilizado o serviço da Azure para postgres.
RA05	Deve ser utilizada a plataforma Azure Devops como repositório de código.
RA06	Deve ser utilizada toda a infraestrutura e ferramentas da Azure para CI, CD
RA07	Todo armazenamento de chaves deve ser usado utilizando o Azure KeyVaults
RA08	O Sistema deve possuir boa experiência de uso tanto no desktop quanto no Mobile.
RA09	O Pagamento das compras dos cliente deve ser feito através do gateway pagar.me que possui boa reputação e é um dos mais utilizados pelo mercado.

RA10	O disparo de e-mails deve ser realizado através da plataforma Twilio Sendgrid que possui um plano gratuito para até 100 emails/dia
------	---

3.2 *Requisitos Funcionais*

Para que se tenha um desenvolvimento completo, uma visão macro das funções e para que seja evitado o retrabalho ou que alguma função importante possa passar despercebida, é importantíssimo que sejam definidos de forma clara os requisitos que modelam o projeto de forma completa. Abaixo serão listados todos os requisitos funcionais que precisam ser satisfeitos, para que o software cubra todas as necessidades identificadas.

ID	Descrição Resumida	Dificuldade (B/M/A)*	Prioridade (B/M/A)*
RF01	O sistema deve ter uma página que permita que o usuário realize seu cadastro.	B	A
RF02	O sistema deve ter uma tela para que o usuário que já possua cadastro efetue o login.	B	A
RF03	O sistema deve permitir que o usuário realize o logoff quando desejar, em qualquer página que estejam.	B	M
RF04	O sistema deve ter uma página com as informações do usuário logado.	M	B
RF05	O sistema deve ter usuários definidos por perfis, o perfil investidor e o perfil de analista.	M	A
RF06	O sistema deve ter uma página informativa para que o usuário analista possa verificar os dados referentes as vendas de seus relatórios.	M	M
RF07	O sistema deve ter uma página para que o usuário analista faça upload de um relatório sobre uma empresa.	A	A
RF08	O sistema deve permitir que o usuário faça o upload de uma foto para seu perfil.	M	B
RF09	O Upload e Download dos relatórios deve ser feito através da ferramenta de Armazenamento de Arquivos do Azure.	M	A
RF10	O Sistema deve exibir na página inicial, após o login, uma tela com os relatórios disponíveis para compra.	A	A
RF11	Ainda na tela inicial, deve existir uma seção com as últimas notícias referentes ao mercado financeiro.	M	B
RF12	Também deve ser exibido na página inicial um pequeno box com as ações que mais subiram e desceram no dia.	A	B

RF13	Deve existir uma página com um mapa baseado na variação das ações do dia, separando por setor da empresa.	A	B
RF14	O mapa deve exibir um botão de compartilhamento para compartilhar a imagem do mapa nas redes sociais.	B	B
RF15	A tela com os relatórios disponíveis deve possuir filtros para que o usuário faça buscas.	M	M
RF16	Ao selecionar um dos relatórios da listagem, deve abrir uma tela com detalhes daquele relatório e do autor. Além de um botão para compra ou adicionar ao carrinho.	M	A
RF17	Quando for um relatório gratuito, na página de detalhes do relatório, deve existir a opção de baixar no lugar do botão de comprar.	B	B
RF18	Ao clicar em comprar, o sistema deve abrir uma página para que seja feita a compra pelo usuário.	A	A
RF19	O sistema deve ter integração com um gateway de pagamento.	A	A
RF20	Após o pagamento, o sistema deve informar o código do pedido e enviar um email notificando o usuário.	B	A

*B=Baixa, M=Média, A=Alta.

3.3 *Requisitos Não-funcionais*

Os Requisitos Não-Funcionais são importantes para alinhar as expectativas sobre o uso do sistema e de requisitos que a arquitetura do sistema deve atender para que essas expectativas de uso sejam satisfeitas. Abaixo listamos todos os requisitos não-funcionais observados e suas respectivas prioridades.

ID	Descrição	Prioridade B/M/A
RNF01	O sistema deve ser apto a ser utilizado tanto por desktop como mobile. Sendo ajustado de acordo com cada tela.	M
RNF02	O sistema deve ser compatível com os principais navegadores mais modernos.	M
RNF03	O sistema deve permitir o login dos usuários utilizando JWT e utilizar roles para os perfis de usuários.	A
RNF04	O sistema deve estar disponível 24horas por dia de segunda a domingo.	A
RNF05	O sistema deve ser capaz de receber 3.000 acessos simultâneos	A
RNF06	A aplicação deve ser resiliente em falhas em APIs externas, implementando fallbacks.	M

RNF07	O tempo de resposta de alguma ação do usuário nunca pode ser superior a 4 segundos.	A
RNF08	Visando uma performance mais otimizada da aplicação, deve ser utilizado cache sempre que possível.	M
RNF09	A plataforma deve ser distribuída em containers para que seja escalada em momentos de pico de acessos.	A
RNF10	Toda parte de infraestrutura deve ser automatizada utilizando a plataforma Azure DevOps.	A
RNF11	O armazenamento de arquivos deve ser realizado na ferramenta de armazenamento da Azure.	M
RNF12	O sistema deve disparar um email de confirmação da compra através do sistema terceiro Twilio Sendgrid	M
RNF13	As comunicação entre Front-End e APIs devem ser realizadas através do padrão REST	A

3.4 Mecanismos Arquiteturais

Esta seção visa apresentar as escolhas do arquiteto referente às escolhas das tecnologias que são mais aderentes a proposta de desenvolvimento e arquitetura do projeto. Tendo em vista as restrições arquiteturais e conhecimentos técnicos da equipe.

Estes mecanismos são divididos em três grupos: Análise, design e implementação.

No grupo de Análise, são listados os aspectos gerais básicos que compõem a arquitetura do software. Em Design são apresentados os padrões tecnológicos escolhidos. E finalmente no grupo de Implementação são identificados os frameworks/bibliotecas escolhidos que implementam cada mecanismo.

Análise	Design	Implementação
Persistência	ORM	Hibernate
Persistência	PostgreSQL	Azure database for PostgreSQL
Armazenamento de arquivos	Blob	Azure Blob Storage
Front end	MVC	ReactJs
Back end	APIS	Java/ Spring Boot
Integração	API Rest	OpenFeign
Log	Recursos de log	Log4j
Resiliência	Retry e Circuit Breaker	Spring Retry
Autorização e Autenticação	Json Web Token	Java JWT
Disparo de Email	SMTP	Twilio Sendgrid
Teste de Software	Testes Unitários	Junit

Deploy	CI/CD	Azure Devops
--------	-------	--------------

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 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 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

Abaixo, apresentamos o diagrama de contexto representando uma visão geral da arquitetura da plataforma

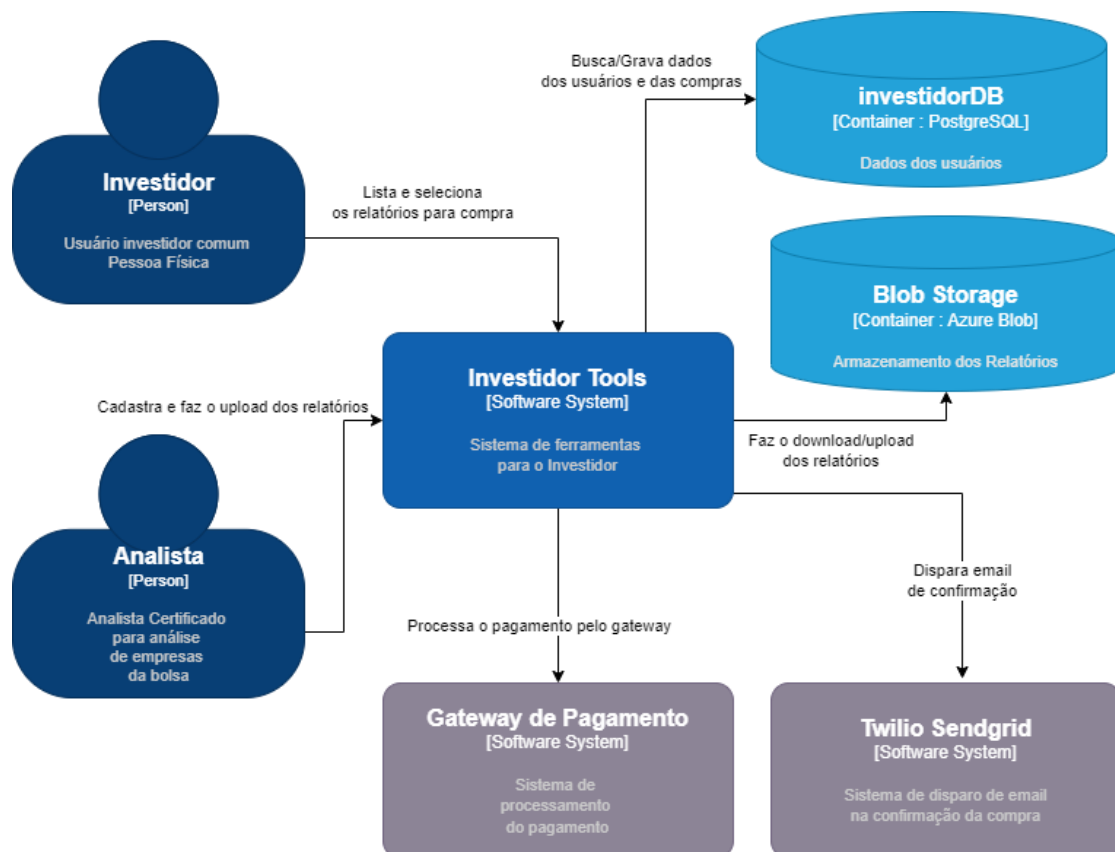


Figura 1 - Visão Geral da Solução

A figura 1 mostra a o diagrama de contexto geral da solução proposta, com todos seus principais sistemas internos e externos, e os usuários envolvidos no fluxo do negócio. Podemos verificar que o usuário analista utilizará principalmente, as APIs desenvolvidas internamente neste projeto, tendo a única dependência externa com o serviço de armazenamento dos relatórios. Já o usuário investidor, terá interação com as APIs que serão desenvolvidas internamente e também com todas as APIs de terceiros que serão utilizadas para download dos arquivos, processamento do pagamento e envio de e-mails.

4.2 Diagrama de Container

Nesta seção, apresentamos o diagrama de container desenvolvido com base nos levantamentos feitos anteriormente.

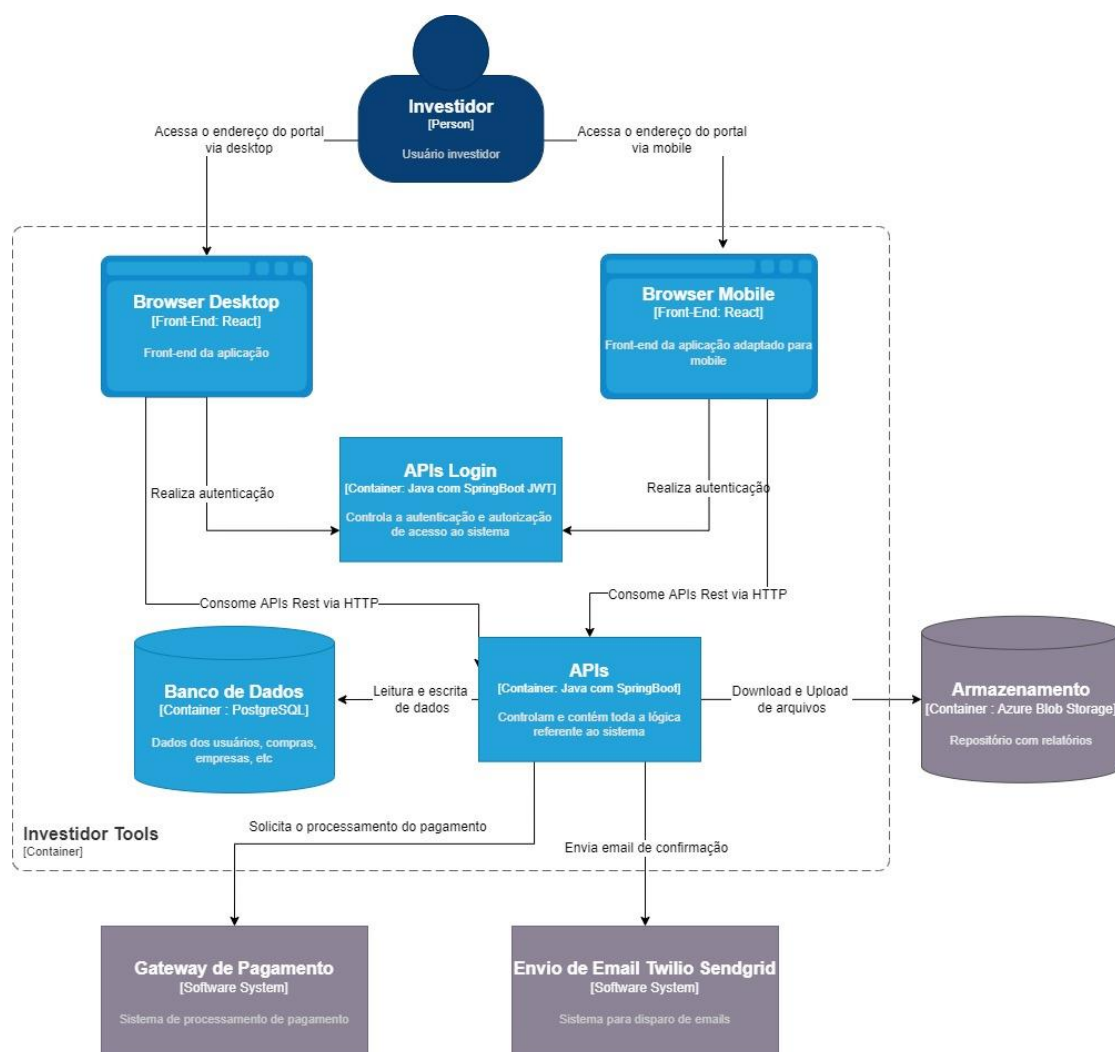


Figura 2 – Diagrama de container da plataforma Investidor Tools.

A figura 2 apresenta os containers da aplicação Investidor Tools, onde são representados os usuários do sistema, APIs, repositórios de arquivos, base de dados e sistemas externos utilizados no processo do sistema.

O usuário investidor irá utilizar o sistema para consultar e comprar relatórios de análise de empresas listadas na bolsa de valores do brasil.

Para utilizar o sistema, o usuário precisará realizar seu cadastro e depois fazer o login no sistema, onde serão validadas suas informações.

A plataforma Investidor Tools, será disponibilizada por navegadores online. O front-end irá adaptar automaticamente seu layout de acordo com as dimensões do dispositivo do usuário, tanto quando for via desktop como quando for via mobile.

O usuário irá interagir com o sistema através de APIs via comunicação HTTP. As APIs, por sua vez, irão buscar e gravar os dados do sistema em uma base de dados postgresql.

Quando o usuário decidir comprar um relatório, então as APIs, irão integrar com um gateway de pagamento externo que garantirá segurança durante a operação. Após o fim do processamento do pagamento, as APIs, permitirão que o usuário realize o download do relatório comprado por meio de uma integração com o azure Blob storage.

O usuário será também notificado por email, a respeito de sua compra e demais comunicações, através de uma integração com a API de disparo de e-mails da Twilio.

4.3 Diagrama de Componentes

Abaixo apresentamos o Diagrama de Componentes da aplicação, indicando os elementos da arquitetura e as interfaces entre eles.

O usuário irá interagir com a aplicação através de um browser, que poderá ser utilizada em qualquer browser dos mais recentes do mercado. Através do browser será feito o acesso ao front-end da aplicação que será desenvolvido utilizando ReactJS, utilizando o padrão MVC.

O front-end da aplicação irá se comunicar com as APIs que serão desenvolvidas, passando primeiramente por um API Gateway, se autenticando através de JWT. As APIs serão desenvolvidas utilizando a linguagem Java utilizando a versão 11, utilizando o frameworks Spring Boot para acelerar o desenvolvimento das APIs.

As APIs utilizarão o Redis para realizar o cache de chamadas que se repetem, ajudando assim, na performance da aplicação.

Além disso, as APIs se comunicarão com serviços da azure como o banco de dados postgresql e azure blob storage. Essa comunicação será realizada através do protocolo HTTPs.

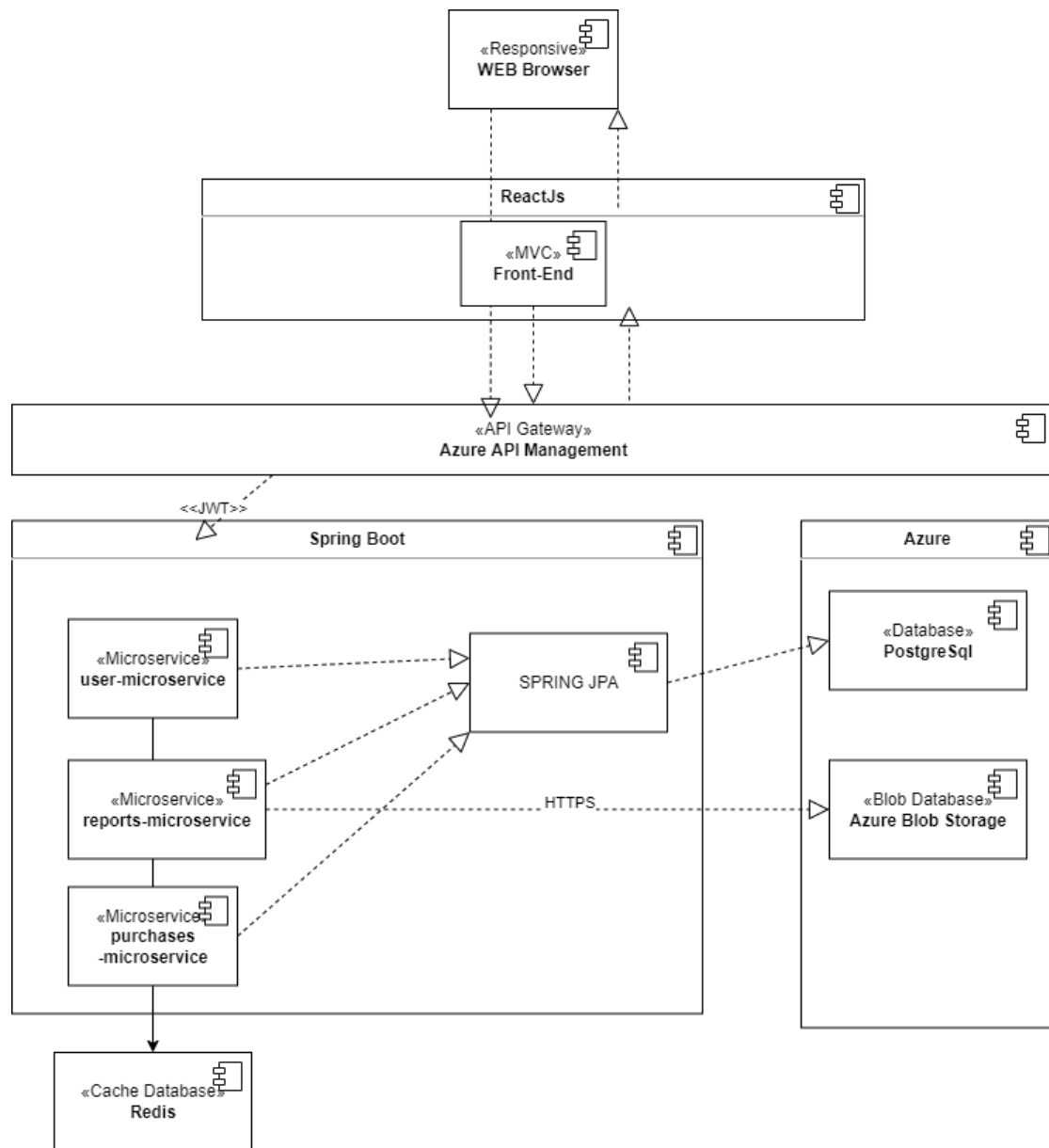


Figura 2 – Diagrama de Componentes.

5. Prova de Conceito (PoC)

Para que seja validada a ideia do projeto, foram escolhidos 3 Requisitos funcionais principais para serem implementados no desenvolvimento da prova de conceito.

1. **Funcionalidade de Login:** Para qualquer aplicação, é necessário um sistema de login para o usuário realizar suas operações, principalmente em um sistema que vai permitir compra de relatórios, é importante que o sistema tenha bem estruturada a funcionalidade de login e logout;
2. **Funcionalidade de Listagem de relatórios:** Essa é a principal funcionalidade do sistema. Permite que o usuário liste os relatórios cadastrados pelos analistas em nossa base de dados, permitindo que o usuário possa ver detalhes do relatório e realizar a compra / download dos mesmos.
3. **Funcionalidade de Download de relatórios:** Após a funcionalidade de listagem dos relatórios, a de download foi considerada também de muita importância. Para a POC iremos realizar o download dos relatórios que estão armazenados no azure blob storage.

5.1 Integrações entre Componentes

Foi desenvolvido um protótipo navegável para representar o que se espera relacionado a interface da aplicação quando todos os requisitos funcionais forem atendidos.

O protótipo visa um design diferenciado e de fácil navegação para os usuários.

1 – Tela de Registro:

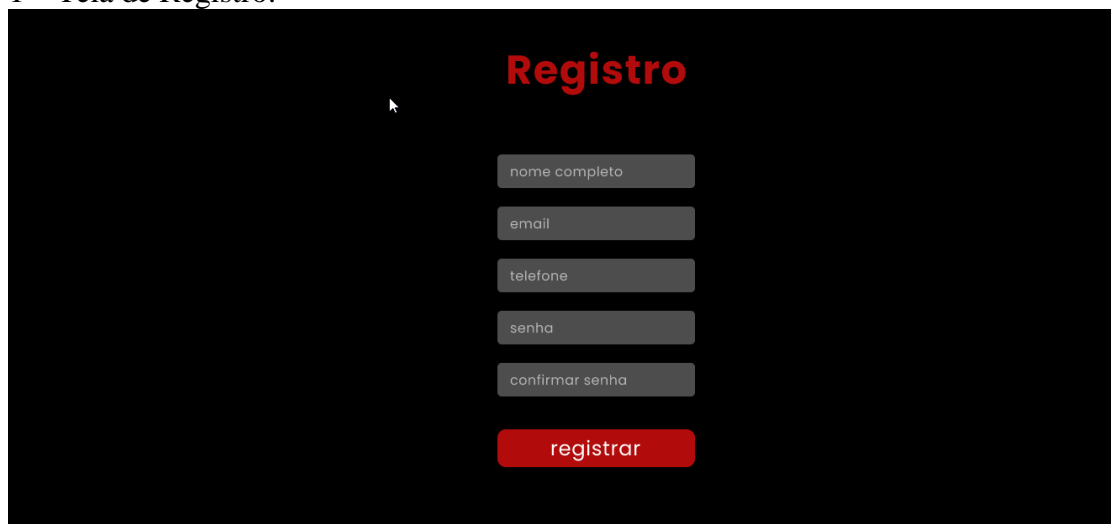
A imagem mostra um protótipo de uma tela de registro com o título "Registro" em vermelho. Abaixo do título, há cinco campos de entrada de texto cinza: "nome completo", "email", "telefone", "senha" e "confirmar senha". No final, há um botão vermelho com o texto "registrar".

Figura 4 – Protótipo

2 - Tela de Listagem de Relatórios



Figura 5 – Protótipo

3 – Tela de detalhes do relatório

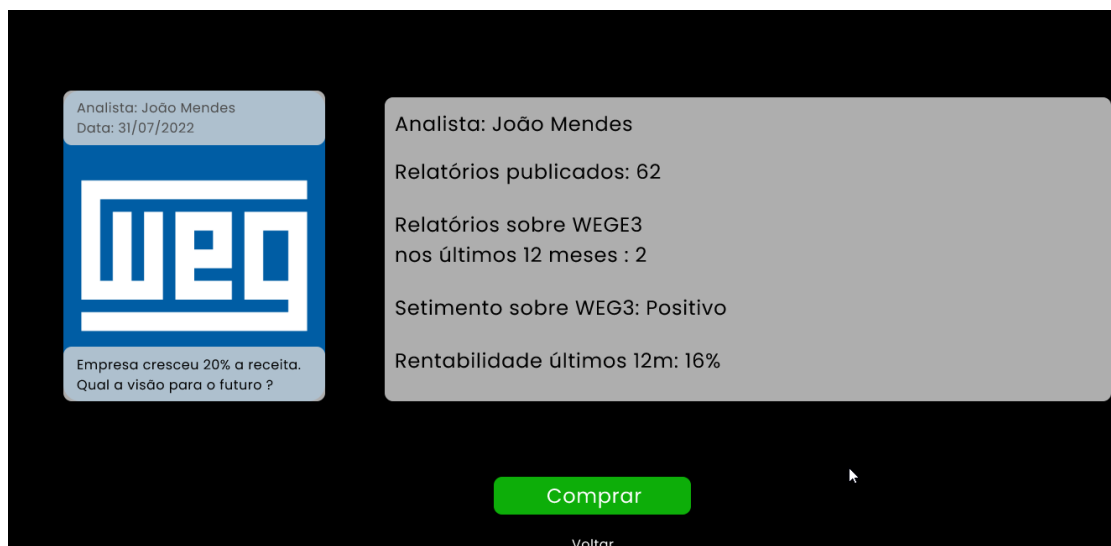


Figura 6 – Protótipo

4 – Tela de Compra



Figura 7 – Protótipo

5 – Tela de compra aprovada



Figura 8 – Protótipo

Link do protótipo navegável:

<https://www.figma.com/proto/wyAZEBJYtS0JpelycBkX6P/Investidor-Tools?node-id=31%3A42&scaling=min-zoom&page-id=31%3A30&starting-point-node-id=31%3A42>

5.2 Código da Aplicação

Abaixo está descrito o código da aplicação conforme o padrão arquitetural C4. O código fonte e endereço dos serviços estão descritos logo em seguida.

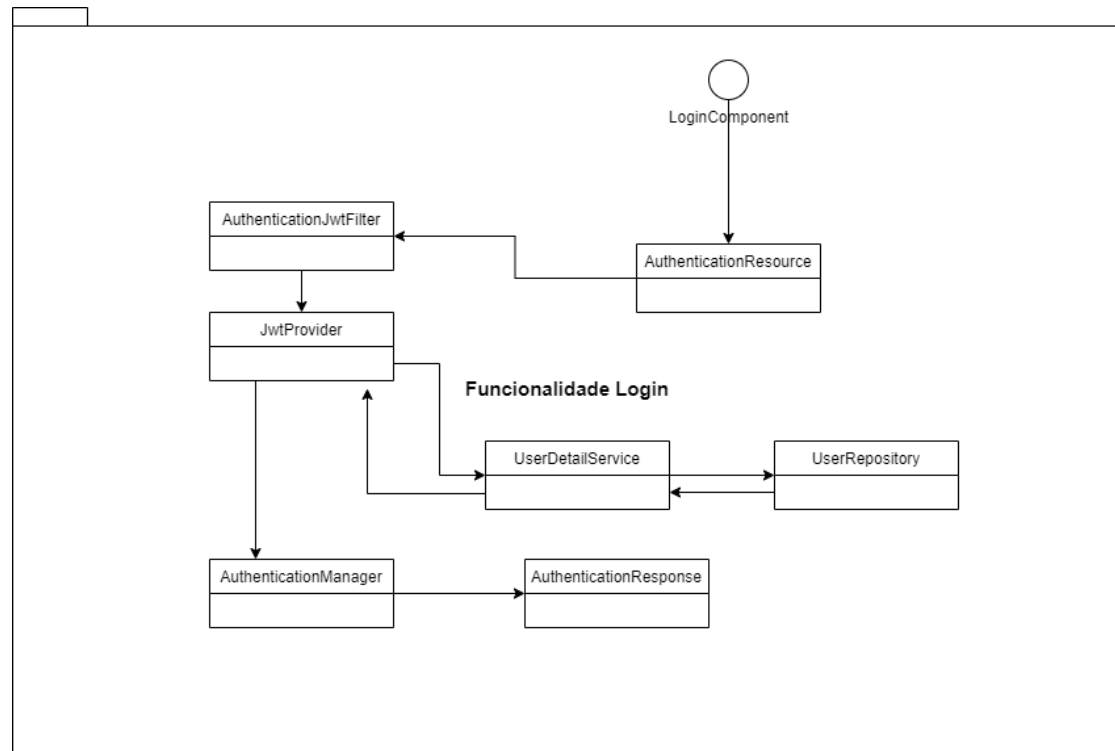


Figura 9 – Estrutura de código da funcionalidade Login

Na figura 9 é detalhado o funcionamento do código da funcionalidade de login do sistema. O fluxo começa ao receber uma requisição no AuthenticationResource, o filtro AuthenticationJwtFilter verifica os dados de login enviados pelo usuário, acionando o serviço UserDetailsService que chama o microserviço User, que vai ao userdetailRepository verificar se existe usuário com o login e senha recebidos. Se existe usuário e senha compatíveis, então o UserDetailsService retorna ao JwtProvider que gera um Jwt token e passa ao AuthenticationManager para liberar o acesso ao usuário, retornando o token no response.

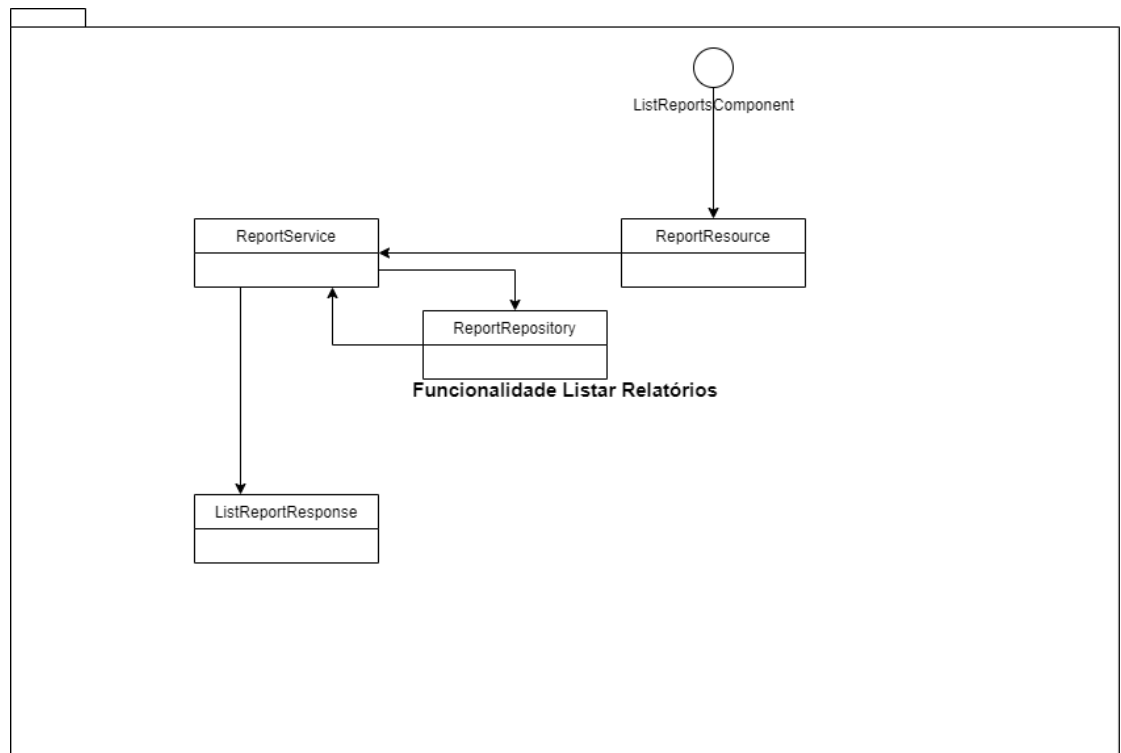


Figura 10 – Estrutura de código da funcionalidade Listar Relatórios

Na figura 10 é detalhado o funcionamento do código da funcionalidade de listagem de relatórios do sistema. O fluxo inicia ao receber uma requisição no **ReportResource**, que chama o **ReportService** que faz a chamada Http Rest ao Microserviço Report, que acessa o banco postgresdb hospedado na azure através do Repository, buscando os relatórios registrados no banco, ordenados pela data de cadastro a partir do mais recente. Esses dados são retornados em um objeto Json para o **ReportService** que monta e retorna o response ao front-end da aplicação.

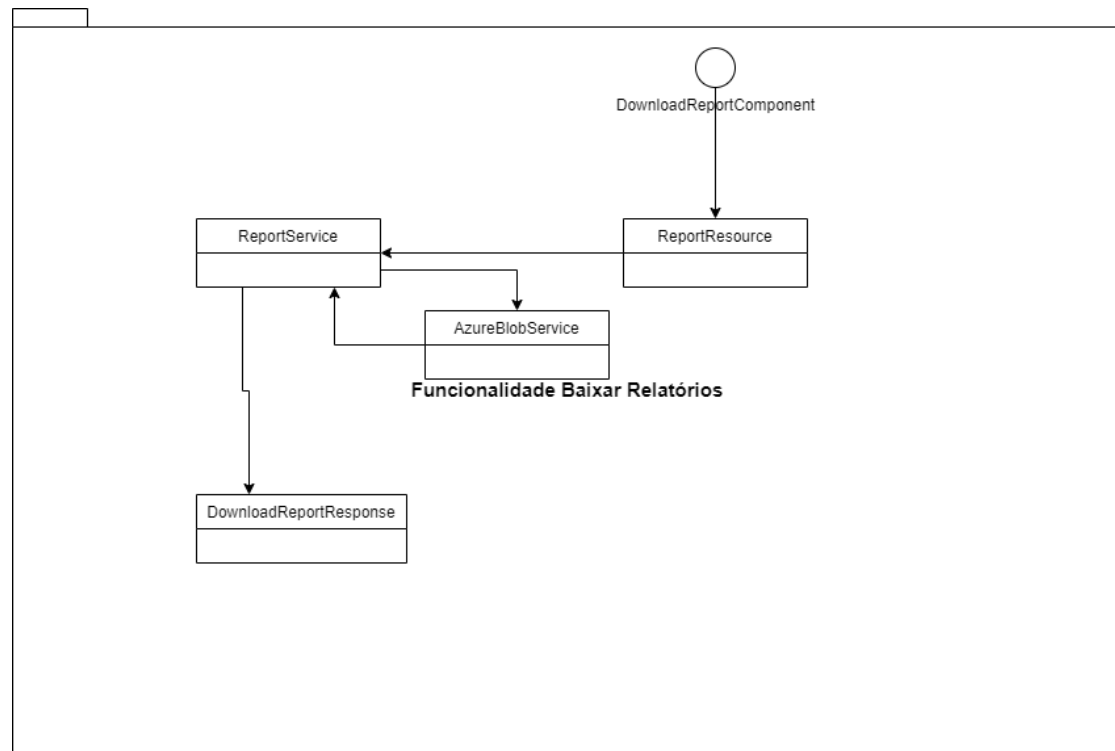


Figura 11 – Estrutura de código da funcionalidade de Download de Relatórios

Na figura 11 é detalhado o funcionamento do código da funcionalidade de download de relatórios do sistema. O fluxo inicia ao receber uma requisição no ReportResource, que chama o ReportService que faz a chamada Http Rest ao Microserviço Report, que acessa o repositório de arquivos Blob da azure, buscando o arquivo do relatório. Após receber o arquivo da azure, retorna para o ReportService, onde o arquivo é transformado em base64 e retornado ao front-end da aplicação.

Foram desenvolvidos para atender aos 3 requisitos funcionais, 3 serviços back-end feitos em Java e 1 Aplicação front-end desenvolvida com React.

Para os testes foi criado um usuário com uma credencial para ser usado no login.

Usuário: puc@gmail.com

Senha: 123456

1 – **Report:**

Realiza as operações relacionadas aos relatórios e faz acesso a base de dados

Acesso: <https://report-1665847208488.azurewebsites.net>

Repositório: <https://github.com/rafaelspereira90/investidorTools-Report-Microservice>

2 – **User:**

Realiza as operações relacionadas aos usuários e login e faz acesso a base de dados

Acesso: <https://user-1665854249598.azurewebsites.net>

Repositório: <https://github.com/rafaelspereira90/investidorTools-User-Microservice>

3 – **Facade:**

Funciona como uma fachada. Recebe as requisições do front-end e realiza as chamadas aos serviços que acessam o banco de dados.

Acesso: <https://facade-1665855219335.azurewebsites.net>

Repositório: <https://github.com/rafaelspereira90/investidorTools-Facade>

4 – **Front-end:**

Interface do sistema.

Acesso: <https://investidortools.azurewebsites.net>

Repositório: <https://github.com/rafaelspereira90/investidorTools-front-end>

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

Atributos de Qualidade	Cenários	Importância	Complexidade
Usabilidade	Cenário 1: O sistema deve ser compatível com os principais navegadores mais modernos.	A	M
Disponibilidade	Cenário 2: O sistema deve estar disponível 24horas por dia de segunda a domingo.	M	B
Performance	Cenário 3: O tempo de resposta de alguma ação do usuário nunca pode ser superior a 4 segundos.	A	M
Usabilidade	Cenário 4: O sistema deve ser apto a ser utilizado tanto por desktop como mobile. Sendo ajustado de acordo com cada tela.	M	M
Usabilidade	Cenário 5: O armazenamento de arquivos deve ser realizado na ferramenta de armazenamento da Azure.	M	M
Usabilidade	Cenário 6: O download do relatório deve ser feito em tempo inferior a 30 segundos.	M	M

6.2. Cenários

Cenário 1 – Usabilidade: A interface do sistema deve ser compatível com os principais navegadores mais modernos, tendo o mesmo comportamento e design em todos eles. Permitindo que o usuário tenha livre escolha de qual navegador usar.

Cenário 2 – Disponibilidade: O sistema deverá estar disponível para o acesso do cliente durante todos os dias tendo uma tolerância de indisponibilidade de 1%.

Cenário 3 – Performance: As requisições feitas através da interface do usuário, nunca podem exceder o tempo de 4 segundos, visando uma boa experiência.

Cenário 4 – Usabilidade: O sistema deve poder ser usado tanto por desktops como também por dispositivos móveis, se adaptando para cada tela. Permitindo que os usuários utilizem o sistema sempre que necessário mesmo que não estejam de posse de um desktop.

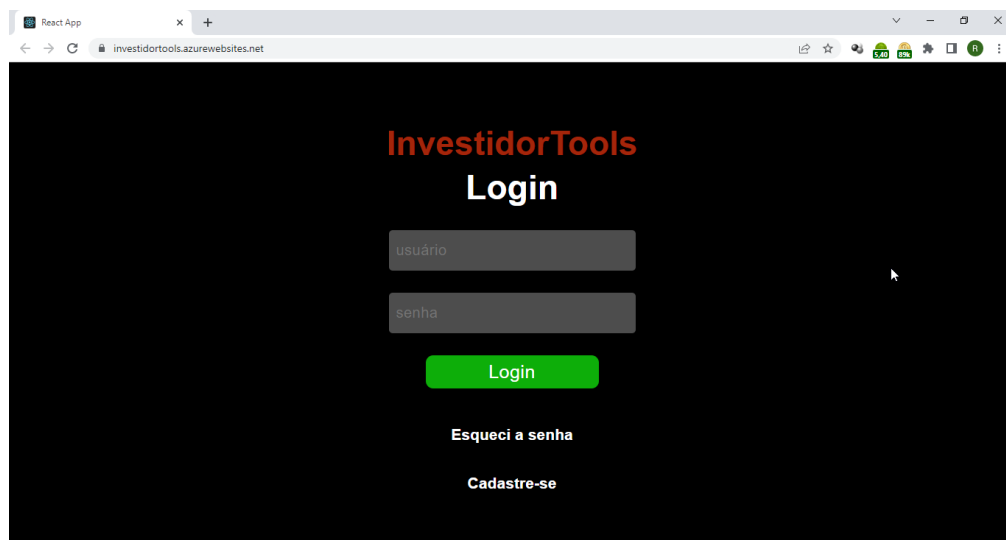
Cenário 5 – Usabilidade: O armazenamento de relatórios deve ser realizado utilizando a ferramenta azure blob, que possui bom desempenho e baixo custo de manutenção e transferência dos arquivos.

Cenário 6 – Usabilidade: O download do relatório não pode ultrapassar o tempo de 30 segundos.

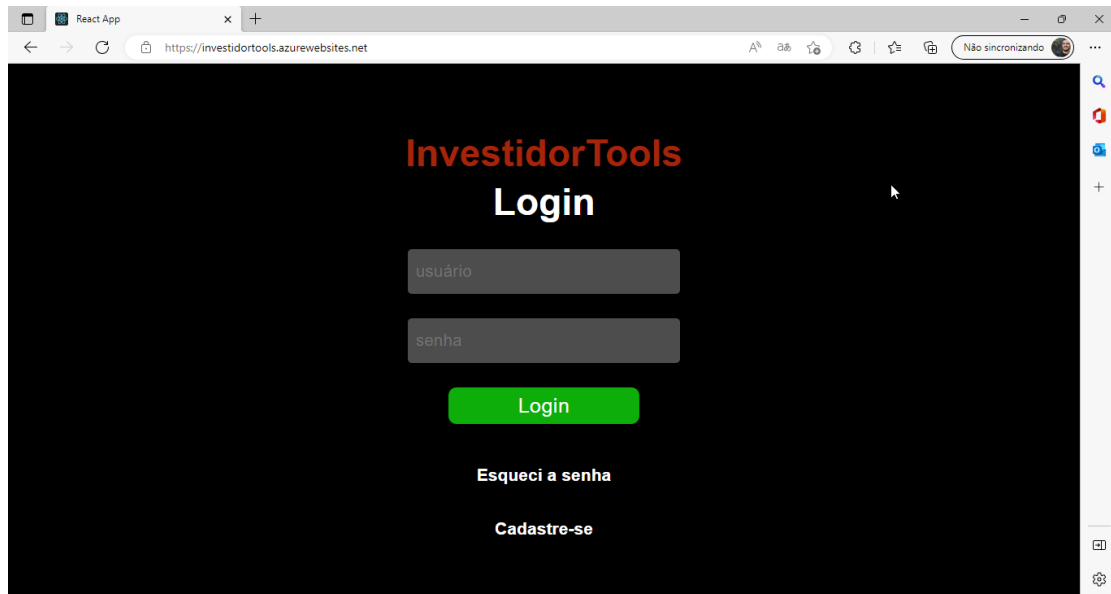
6.3. Evidências da Avaliação

Atributo de Qualidade:	Usabilidade
Requisito de Qualidade:	O sistema deve ser compatível com os principais navegadores mais modernos.
Preocupação:	
A interface do sistema deve ser compatível com os principais navegadores mais modernos, tendo o mesmo comportamento e design em todos eles. Permitindo que o usuário tenha livre escolha de qual navegador usar.	
Cenário(s):	
Cenário 1	

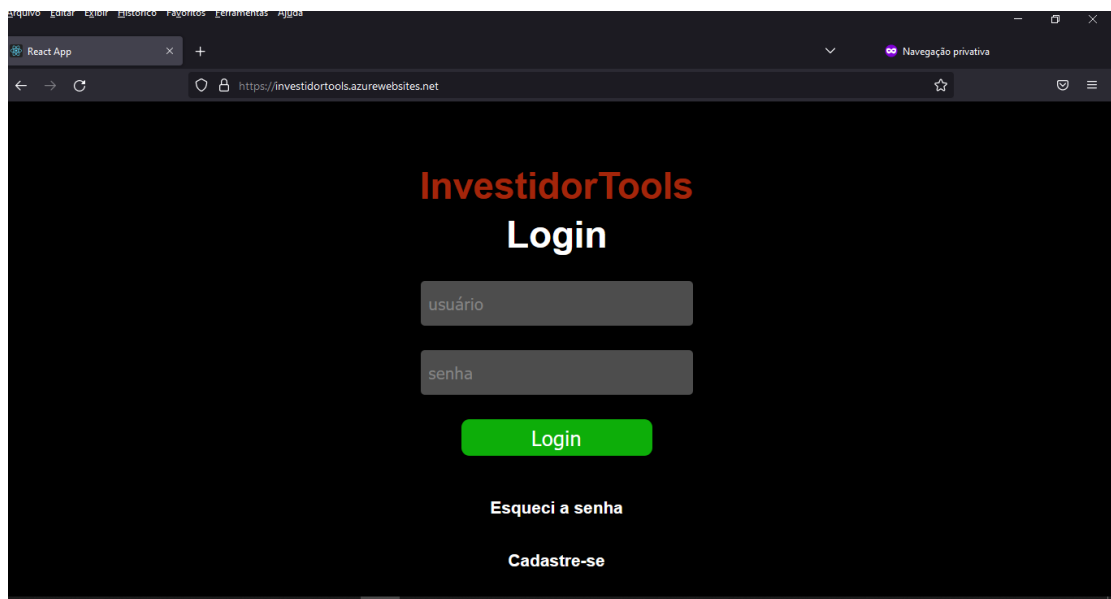
Ambiente:	
Sistema em operação normal	
Estímulo:	
Acesso ao sistema utilizando os principais navegadores do mercado.	
Mecanismo:	
Visualização da interface nos navegadores Chrome, Firefox e Edge.	
Medida de resposta:	
Retornar a interface do sistema me seu estado normal	
Considerações sobre a arquitetura:	
Riscos:	Mudanças de compatibilidade em novas versões dos navegadores podem mudar a forma que a interface é visualizada.
Pontos de Sensibilidade:	Incompatibilidade entre Navegador e biblioteca usada no front-end
Tradeoff:	Não há



Teste realizado no navegador Chrome



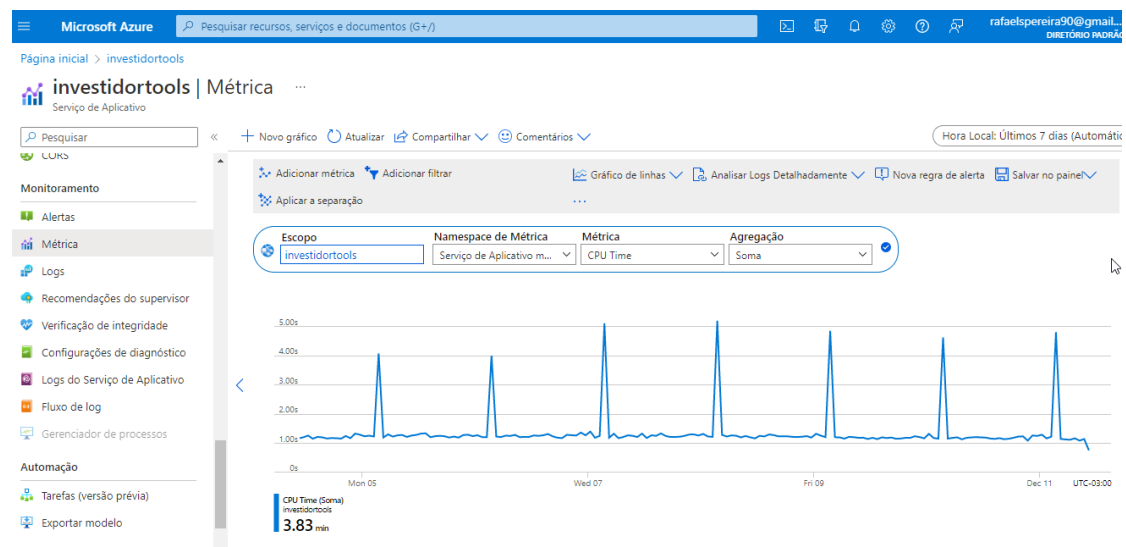
Teste realizado no navegador Edge



Teste realizado no navegador Firefox

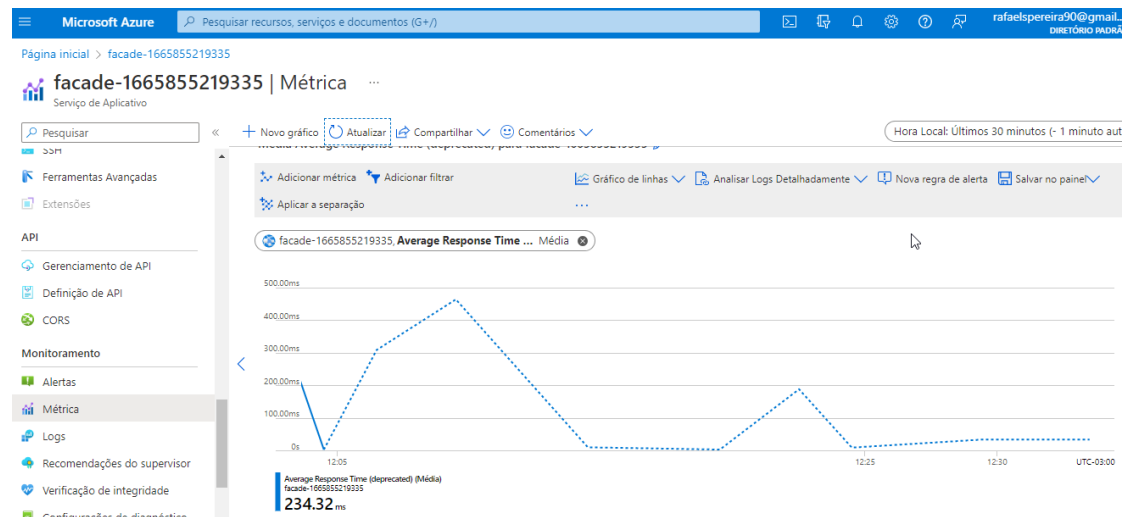
Atributo de Qualidade:	Disponibilidade
Requisito de Qualidade:	O sistema deverá estar disponível para o acesso do cliente durante todos os dias tendo uma tolerância de indisponibilidade de 1%.
Preocupação:	

O usuário deve conseguir acessar o sistema em qualquer dia e qualquer horário. O sistema deve estar disponível sempre, com uma tolerância de indisponibilidade de 1% do tempo.	
Cenário(s):	
Cenário 2	
Ambiente:	
Sistema em operação normal	
Estímulo:	
Acesso ao sistema em diferentes horários.	
Mecanismo:	
Visualização do dashboard da azure que monitora o serviço filtrando pelos últimos 7 dias.	
Medida de resposta:	
O dashboard deve mostrar o uso de cpu.	
Considerações sobre a arquitetura:	
Riscos:	Indisponibilidade geral do provedor de cloud pode deixar o sistema inoperante e frustrar os usuários.
Pontos de Sensibilidade:	Não há
Tradeoff:	Não há



Dashboard da Azure de uso da CPU pela aplicação nos últimos 7 dias.

Atributo de Qualidade:	Performance
Requisito de Qualidade:	As requisições feitas através da interface do usuário, nunca podem exceder o tempo de 4 segundos, visando uma boa experiência
Preocupação:	
O sistema deverá enviar respostas rápidas, para que problemas de conexões não causem problemas de usabilidade e confiabilidade do sistema.	
Cenário(s):	
Cenário 3	
Ambiente:	
Sistema em operação normal	
Estímulo:	
Chamandas nas APIs	
Mecanismo:	
Visualização do dashboard da azure que monitora o serviço filtrando pelos últimos 7 dias.	
Medida de resposta:	
O dashboard deve mostrar o tempo de resposta das requisições.	
Considerações sobre a arquitetura:	
Riscos:	Uma demanda aos endpoints acima do esperado pode comprometer a performance e não cumprir esse requisito em alguns momentos.
Pontos de Sensibilidade:	Não há
Tradeoff:	A capacidade dos microserviços pode ser aumentada para responder de forma mais satisfatória, o que aumentaria os custos.



Dashboard da Azure medição do response time pela fachada da aplicação.

6.4. Resultados Obtidos

Abaixo apresento os resultados obtidos nos testes da arquitetura.

Requisitos Não Funcionais	Teste	Homologação
RNF01: O sistema deve ser compatível com os principais navegadores mais modernos.	OK	N.A
RNF02: O sistema deve estar disponível 24horas por dia de segunda a domingo.	OK	N.A.
RNF03: O tempo de resposta de alguma ação do usuário nunca pode ser superior a 4 segundos.	OK	N.A.

7. Avaliação Crítica dos Resultados

A arquitetura proposta e implementada nesse projeto se mostrou acertada e atingiu o objetivo dos requisitos não funcionais definidos anteriormente. As tecnologias utilizadas permitiram que a usabilidade do sistema fosse satisfatória para todos os usuários e entregaram bom desempenho nas funcionalidades disponibilizadas nessa primeira versão.

A escolha da linguagem React, utilizada no front-end junto com o flex-box no css permitiram que o sistema ficasse acessível em todos os principais navegadores da atualidade e que também se readaptasse a diferentes tamanhos de telas.

A escolha de hospedar os sistemas em uma empresa de cloud confiável e bastante intuitiva comparada as outras, também foi uma ótima escolha. Foi economizado tempo no processo de publicação das APIs graças a essa facilidade em seu uso, além disso, também permitiu alta disponibilidade utilizando os servidores e hardware da azure.

Com a utilização de cache na aplicação, foi possível também diminuir o tempo de resposta em requisições mais repetitivas, porém, nos testes apesar do tempo de resposta ficar abaixo do limite de 4 segundos, identificamos que a aplicação possivelmente sofreria com um tempo de resposta acima do aceitável em um cenário de múltiplas requisições simultâneas. Isso aconteceria devido a termos utilizado a configuração mais básica disponível das máquinas da azure, portanto seria algo que poderia ser resolvido sem complexidade.

8. Conclusão

Esse projeto teve o objetivo de apresentar uma arquitetura moderna e escalável para suportar um sistema fictício e que pode ser incrementado e melhorado futuramente. Dessa forma a arquitetura evoluiria conforme a necessidade e sucesso desse projeto.

A elaboração deste relatório se mostrou de imensa importância para uma arquitetura robusta e preparada. Sem ele, muitas informações e preocupações importantes acabariam não sendo notadas, o que causaria prejuízos e retrabalho. É muito importante também a participação de outros membros da equipe em todas as discussões levantadas no relatório.

Os diagramas desenhados durante o processo nos ajudam a ter uma visão mais clara e rápida de toda a arquitetura e nos permite tomar decisões com base neles. Além disso, ele é muito importante também para a apresentação para novos integrantes que a equipe do projeto possa ter.

Os testes e avaliações foram cruciais para um refinamento e para a entrega em produção de um sistema com o menor nível de erro possível.

Com base em todos esses pontos descritos, avalio que o objetivo foi alcançado com sucesso, e para o futuro do projeto existem melhorias a serem implementadas. Primeiramente, adotar o uso das ferramentas de CI/CD e de versionamento e hospedagem de código da azure, que permitirá maior organização, controle e histórico das evoluções desenvolvidas no projeto. Outro ponto que será importante é o upgrade do hardware contratado junto a azure. Foi contratada a configuração mais básica e com o sistema disponível para diversos usuários, ela não será suficiente. Será necessário mais poder de processamento para processar múltiplas requisições simultâneas.

Referências

BRASIL BOLSA BALCÃO. **B3 atinge 5 milhões de contas de investidores em renda variável em janeiro.** São Paulo, 04 de fevereiro de 2022. Disponível em: https://www.b3.com.br/pt_br/noticias/5-milhoes-de-contas-de-investidores.htm.

Acesso em: 03 de Março de 2022.

RABINOVICI, Marcelo. **Dificuldades e mentalidade de investidor.** 19 de março de 2020. Disponível em: <https://dicadehoje7.com/educacao-financeira/dificuldades-e-mentalidade-de-investidor>.

Acesso em: 03 de Março de 2022.