**Projeto de Módulo**

**Arquitetura de Microsserviços e Mobile**

Sumário

[Introdução 4](#_Toc51606590)

[Etapa de Análise utilizando *UML* 5](#_Toc51606591)

[*Problem Statement (mini-mundo)* 5](#_Toc51606592)

[Identificação do Problema e Solução 6](#_Toc51606593)

[Glossário 7](#_Toc51606594)

[Lista de Requisitos 9](#_Toc51606595)

[Modelo de Casos de Uso 10](#_Toc51606596)

[Diagrama de Contexto 10](#_Toc51606597)

[Diagrama de Casos de Uso do Módulo de Gestão de Clientes 11](#_Toc51606598)

[Diagrama de Casos de Uso do Módulo de Gestão de Vagas 12](#_Toc51606599)

[Diagrama de Casos de Uso do Módulo de Gestão de Processos Seletivos 13](#_Toc51606600)

[Diagrama de Casos de Uso do Módulo de Busca de Oportunidades 14](#_Toc51606601)

[Diagrama de Casos de Uso do Módulo de Busca de Candidatos 15](#_Toc51606602)

[Modelo de Classes de Análise 16](#_Toc51606603)

[Diagrama de Classes de Domínio 16](#_Toc51606604)

[Etapa de Projeto utilizando ***UML***, ***C4 Model***, ***Domain-Driven Design*** e outras abordagens 17](#_Toc51606605)

[Microsserviços necessários 17](#_Toc51606606)

[*Roadmap* 17](#_Toc51606607)

[*Modelo de Classes de Projeto (UML)* 18](#_Toc51606608)

[Diagrama de Classes do PerfilMicroservice 18](#_Toc51606609)

[Diagrama de Classes do VagaMicroservice 19](#_Toc51606610)

[Modelo de Comunicação e Implantação 20](#_Toc51606611)

[Diagrama de Comunicação e Implantação dos componentes (*UML*) 20](#_Toc51606612)

[Microservice Canvas 21](#_Toc51606613)

[*C4 Model* 23](#_Toc51606614)

[Cartão *CRC* 27](#_Toc51606615)

[Exemplos de Implementação 28](#_Toc51606616)

[Justificativa 28](#_Toc51606617)

[Código 28](#_Toc51606618)

[Arquitetura e Design do Código 28](#_Toc51606619)

[Documentação de Uso dos Microsserviços 34](#_Toc51606620)

[Implantação 34](#_Toc51606621)

[Comunicação 34](#_Toc51606622)

[I AM (autenticação e autorização) 34](#_Toc51606623)

[Conclusão 35](#_Toc51606624)

[Referências 36](#_Toc51606625)

# Introdução

A proposta deste trabalho é apresentar maneiras de se documentar softwares baseados em arquitetura de Microsserviços.

Para tal foi utilizada parte da *UML* na fase de Análise para entendimento das necessidades e delinear a próxima fase. Na fase de projeto, foi utilizada uma combinação de técnicas de modelagem além da *UML* como *C4 Model*, *Domain-Driven Design* e outras abordagens.

Para representar a proposta de solução, além da documentação será criada uma implementação em Java utilizando a documentação proposta, cujo código será disponibilizado na plataforma de repositórios de código [*GitHub*](https://github.com/cirilojunior/infnet-mit-arquitetura-bloco-microservicos).

# Etapa de Análise utilizando *UML*

## *Problem Statement (mini-mundo)*

A *IT Job Hunters* Ltda é uma empresa que presta serviços de Recursos Humanos terceirizados para diversas empresas do mercado da Região Sudeste e possui mais de 20 anos de experiência com foco na seleção de candidatos para vagas para empresas de Tecnologia da Informação, tais como Analistas de Sistemas, Desenvolvedores, Administradores de Banco de Dados, assim como perfis de gestão ou processos como Gerentes de Projetos, *Product Owners*, *Scrum Master* e Agile *Coaches*.

Com a crescente demanda por mão de obra qualificada, houve um aumento substancial de clientes, assim como algumas aquisições ou fechamento de parcerias com outras empresas do ramo, expandindo sua atuação para outras regiões do Brasil. Além disso, a empresa passou a realizar a intermediação de vagas para empresas estrangeiras, tanto para expatriação quanto para trabalho remoto.

Apesar de pioneira em informatização de seus processos, possui um grande legado de sistemas especializados e não integrados, além de sistemas criados para necessidades específicas de alguns clientes. Tais sistemas já não suportam a carga de trabalho da empresa e as necessárias mudanças de processos, pondo em risco seu protagonismo em seu nicho e preocupando todo o *board* de gestão.

Para superar esses desafios, a empresa utilizará tanto equipe de TI interna, mas também realizará algumas contratações temporárias de acordo com a etapa em questão, mas com possibilidade de contratação definitiva ao final do processo. Essa dinâmica será inclusive utilizada, sempre que possível, como piloto dos próprios módulos desenvolvidos.

Todo o processo será revisto. Os problemas começam na captação de bons candidatos, na prospecção de oportunidades nos clientes, seguido do *match* entre as necessidades dos clientes e a base de currículos que ainda é manual, demorado e depende muito da experiência do analista que atua em determinado cliente, e deverá ser um dos pontos de maior destaque nessa nova empreitada.

O novo processo que vem sendo discutido internamente passa por módulos de captação de candidatos com detalhe para seus skills e experiências técnicas, de área de negócio, empresas (conhecimento de cultura organizacional).

## Identificação do Problema e Solução

|  |  |
| --- | --- |
| O problema | Processos de negócios lentos e difícil escala |
| Afeta | A velocidade da área fim da empresa |
| Cujo impacto é | Falha na identificação de oportunidades nos clientes e perda do *time* de alguns negócios. |
| Tendo como solução | A criação de uma nova solução de *software* baseada em arquitetura de microsserviços e utilizando uma série de novas práticas de desenvolvimento empiricamente comprovadas pelo mercado. |

## Glossário

1. Mercado – Região do cliente ou vaga.
2. Seleção de Candidatos – Processo de seleção de candidatos.
3. Vaga – Posto de Trabalho requerido por um cliente. Pode ser presencial ou remota.
4. Oportunidade – Necessidade identificada pelos nossos analistas em um determinado cliente que poderá se tornar uma vaga ou encontrada pelos processos automáticos.
5. Cliente – cliente da ***IT Job Hunters***.
6. Perfil – perfil de vaga e perfil de candidato. Usado nas rotinas de match.
7. Processo – Processo de Negócio e Processo Seletivo.
8. Processo de Negócio – Processos internos da empresa.
9. Processo Seletivo – Processos de busca de candidatos para vagas.
10. Analista – colaborador que atua na identificação de necessidades dos clientes, busca e avaliação por candidatos e processo de contratação.
11. Gestão – colaborador que ocupa cargo de gestão e interage em funções de aprovação e acompanhamento por relatórios.
12. Aquisição – Aquisição de empresas do ramo que passam a fazer parte do grupo.
13. Fechamento de Parcerias – Parcerias com outras empresas do ramo em mercados ainda não atingidos, aplicando o processo interno da empresa (marca).
14. Expatriação – Processo de seleção que envolve a mudança do candidato para o país da vaga. A empresa auxilia nesse processo como um produto extra pago pelo cliente.
15. Sistemas especializados – Sistemas especializados em determinados processos de negócio, que agora seriam módulos de um novo sistema.
16. Integrados – Os sistemas antigos não eram integrados ou possuíam algum nível de integração rudimentar, como integração direto pela base de dados. A ideia agora é existir uma integração mais bem orquestrada e aderente aos novos padrões arquiteturais praticados no mercado de desenvolvimento de software. A empresa é focada em profissionais da área e precisa dar exemplo.
17. Contratação Temporária – Vaga com tempo definido para execução de um projeto específico.
18. Contratação Permanente – Vaga cujo perfil interessa ao cliente como parte definitiva da equipe (tempo indeterminado).
19. Piloto – Execução do processo no próprio sistema sendo desenvolvido (módulos existentes) e que pode evoluir para o conceito de demanda / projeto interno de seleção de vagas para um cliente.
20. Captação – Processo de busca por candidatos.
21. Candidato – Aspirante a uma vaga.
22. Prospecção – Processo de análise constante de necessidades internas dos clientes que demandem contratações.
23. Match – evento de descoberta de candidatos para vagas.
24. Base de Currículos – Base com informações de candidatos e seus perfis (habilidades, experiências e senioridade).
25. Negócio – *Core* da empresa que são os Processos Seletivos.
26. Colaborador – Profissional que participa do processo seletivo intermediando contatos entre Candidato e Cliente.

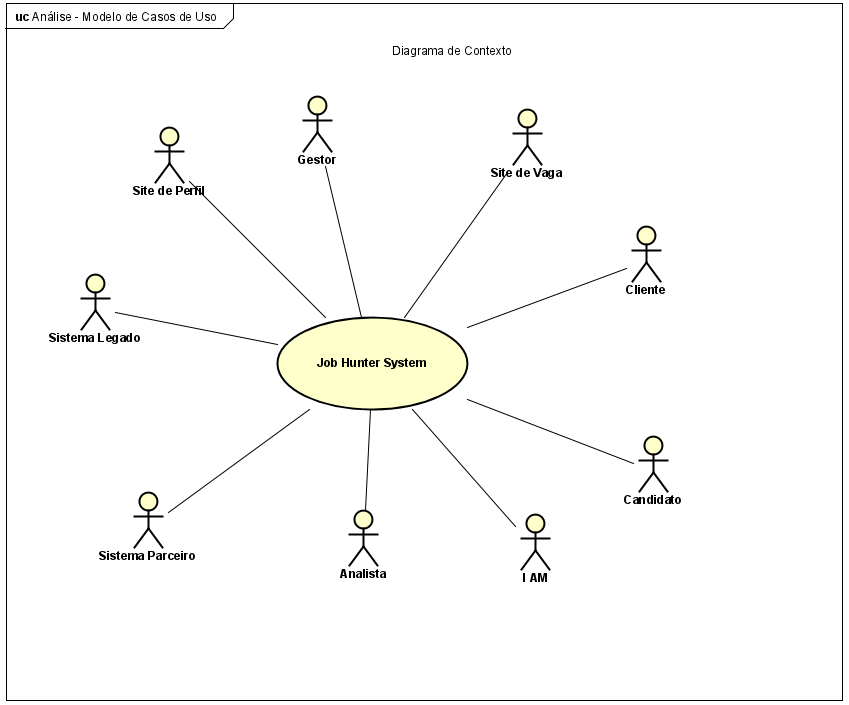
## Lista de Requisitos

* Gestão de Clientes
  + Cadastro de Clientes
  + Cadastro de Mercados
* Gestão de Vagas
  + Cadastro de Perfis
    - Cargo
    - Nível
    - Habilidades
    - Experiências
  + Cadastro de Vagas
    - Presencial (padrão)
    - Remota
    - Aprovação pelo Cliente
* Gestão de Processos Seletivos
  + Cadastro de Candidatos
    - Cargo
    - Nível
    - Habilidades
    - Experiências
  + Cadastro de Contratação
    - Confirmação do Candidato
* Busca de Oportunidades
  + Sites de vagas conhecidos (LinkedIn, APInfo, Vagas.com.br, etc)
  + Sites e bases de clientes
  + Identificadas por Analista
* Busca Candidatos
  + Sites de perfis conhecidos (LinkedIn, APInfo, etc)
* Gestão de Parcerias \*
  + Integração com empresas de mercados onde a *IT Job Hunters* não possui escritório e atua em parceria com outras empresas da área
* Gestão de Aquisições \*
  + Integração rápida pós-aquisição e adaptação de processos com os sistemas de empresa adquirida

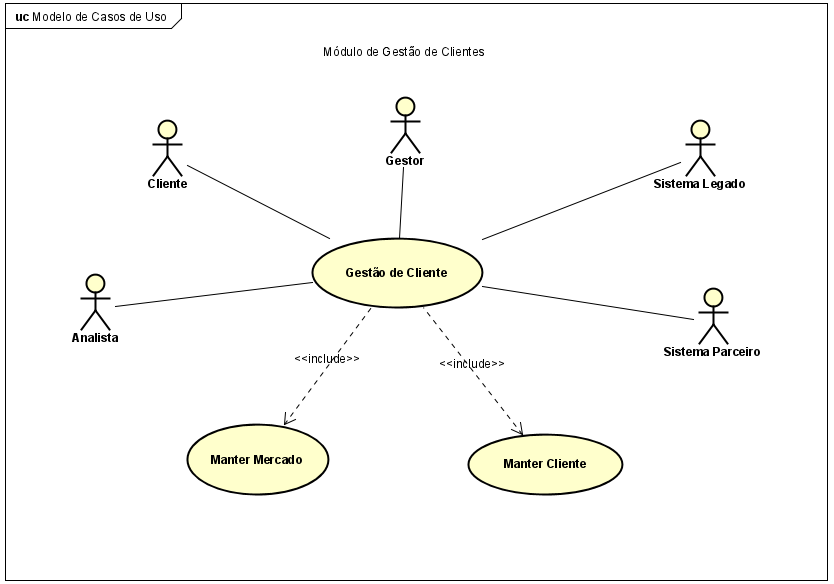
\* Descritos porque fazem parte da contextualização do trabalho, mas como são áreas muito específicas e fogem ao senso comum de atividades de um processo seletivo não serão abordados em detalhes e demais seções do trabalho.

## Modelo de Casos de Uso

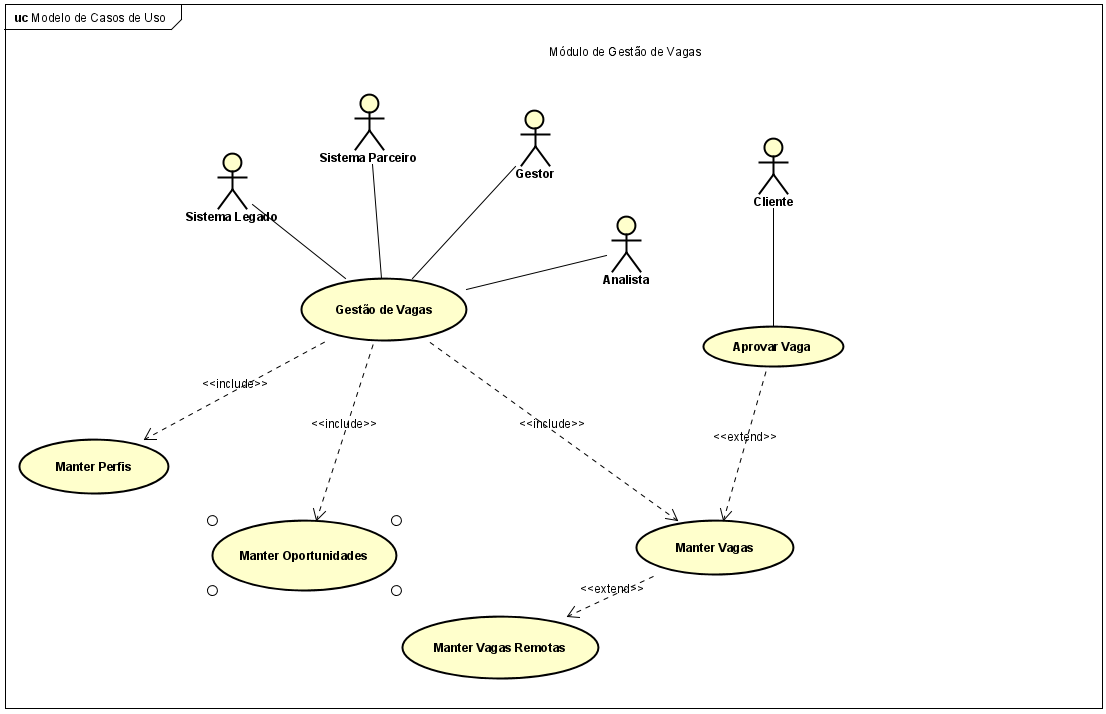
### Diagrama de Contexto



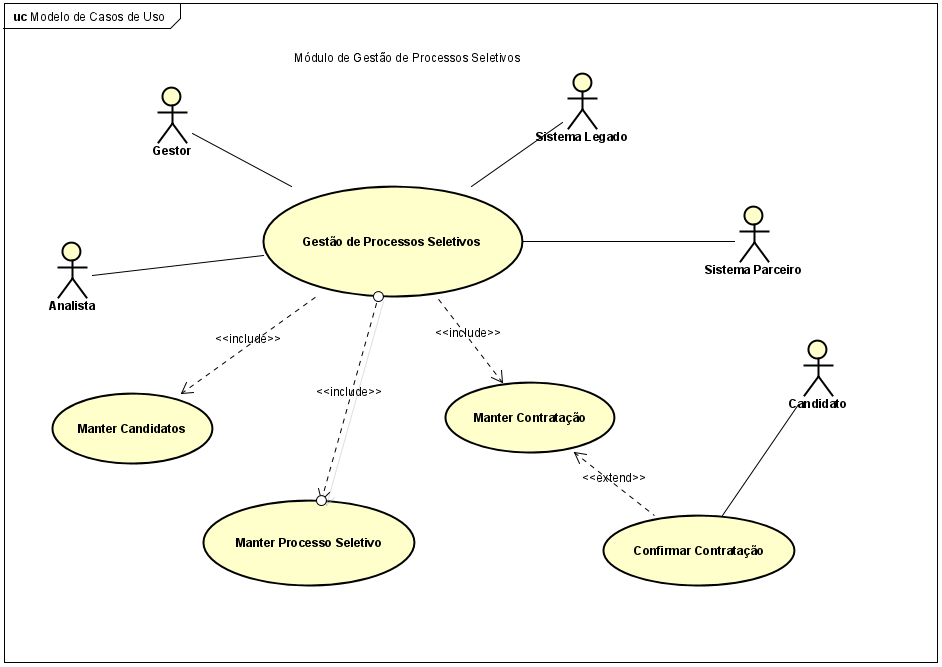
### Diagrama de Casos de Uso do Módulo de Gestão de Clientes



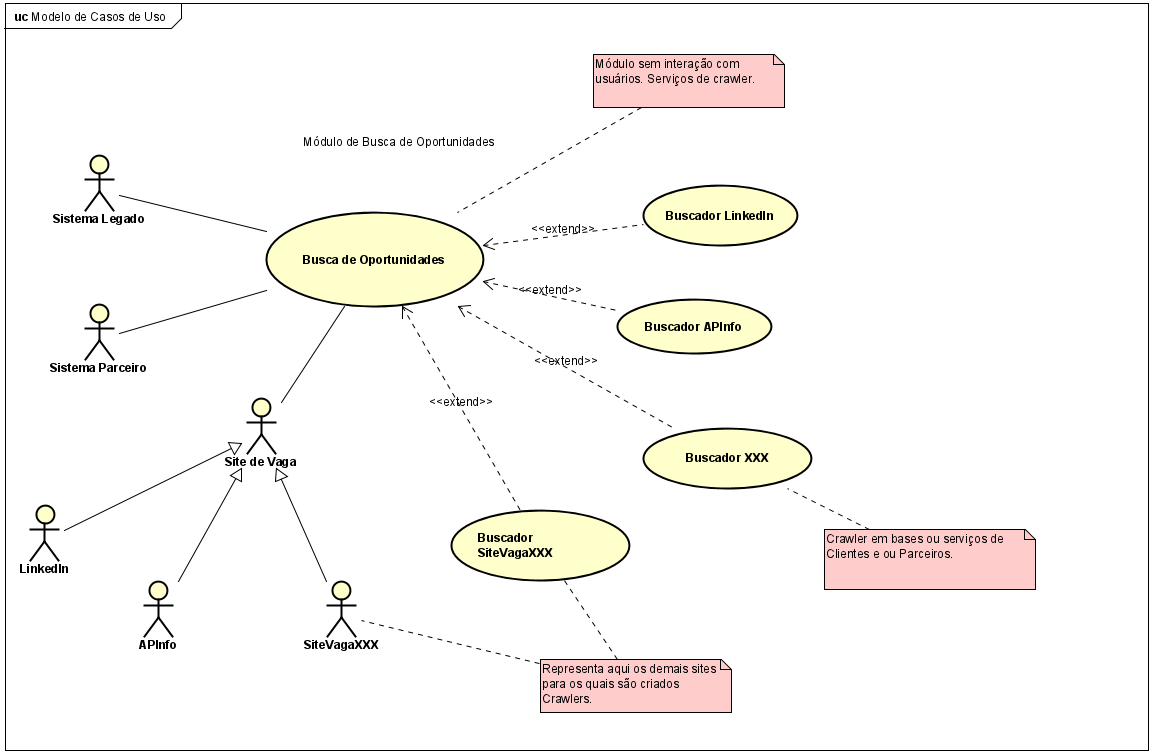
### Diagrama de Casos de Uso do Módulo de Gestão de Vagas



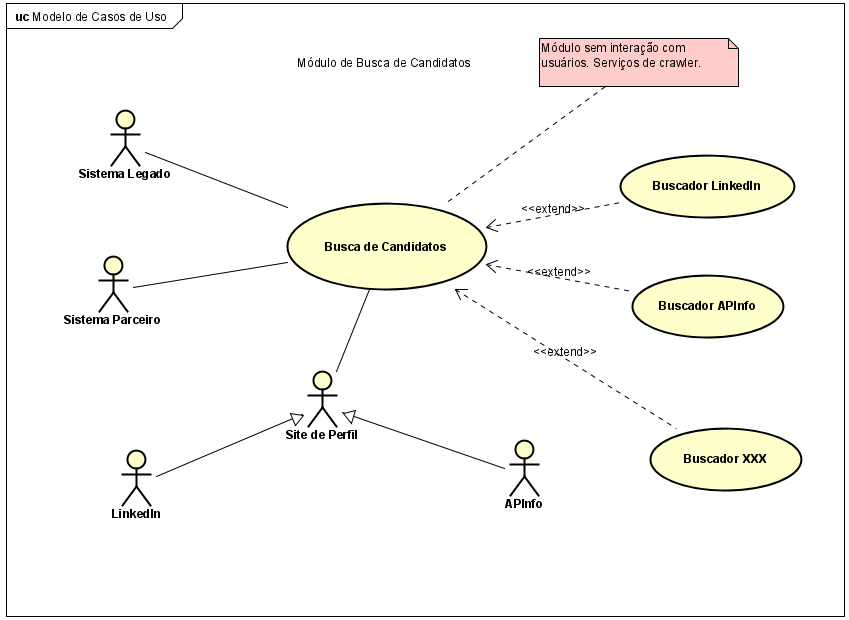
### Diagrama de Casos de Uso do Módulo de Gestão de Processos Seletivos



### Diagrama de Casos de Uso do Módulo de Busca de Oportunidades



### Diagrama de Casos de Uso do Módulo de Busca de Candidatos



## Modelo de Classes de Análise

### Diagrama de Classes de Domínio

# Etapa de Projeto utilizando ***UML***, ***C4 Model***, ***Domain-Driven Design*** e outras abordagens

A proposta é combinar o que existir de melhor em cada uma das abordagens para criar uma documentação útil e rápida para criação, simples para consultas e de fácil manutenção ou descarte em alguns casos.

## Microsserviços necessários

Serão necessários os seguintes microsserviços para implementar o novo sistema:

* Módulo de Gestão de Clientes
  + MercadoMicroservice
  + ClienteMicroservice
* Módulo de Gestão de Vagas
  + PerfilMicroservice
  + OportunidadeMicroservice
  + VagaMicroservice
* Módulo de Gestão de Processos Seletivos
  + ProcessoSeletivoMicroservice
  + CandidatoMicroservice
  + ContrataçãoMicroservice
* Módulo de Busca de Oportunidades
  + BuscadorOportunidadesMicroservice
* Módulo de Busca de Candidatos
  + BuscaCandidatosMicroservice

A representação dos Microsserviços no processo de análise reflete-se nos diagramas de Casos de Uso por módulo. Para todo Caso de Uso do tipo *include* será criado um Microsserviço. Já para os Casos de Uso do tipo *extend* entende-se que, ou estão contemplados como endpoints do Microsserviço principal ou são implementações de estratégias (*Strategy pattern - GoF*) para a estrutura principal de uma determinada rotina (no caso, os buscadores).

## *Roadmap*

Com foco em *ROI\** e análise das pessoas que já foram integradas à equipe para execução do projeto piloto (com possível contratação) foi pensado o seguinte *Roadmap*:

Módulo de Gestão de Vagas >> Módulo de Gestão de Processos Seletivos >> Módulo de Busca de Candidatos >> Módulo de Busca de Oportunidades >> Módulo de Gestão de Clientes.

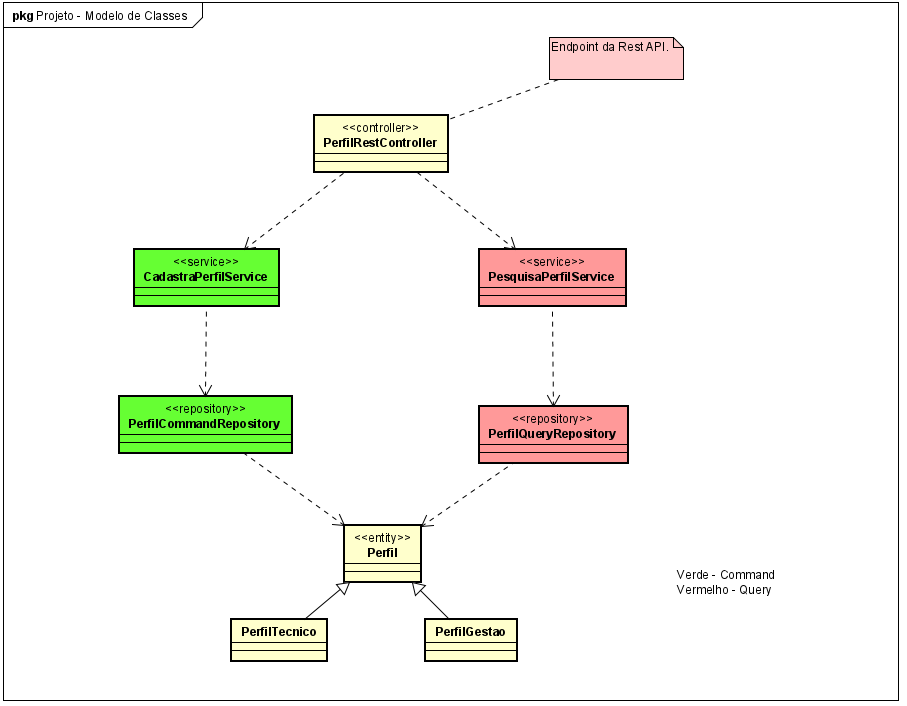
Todas as informações necessárias e já existentes em alguma outra fonte deverão carregadas no novo banco ou lidas, quando for a melhor solução de momento, nos bancos legados.

\* *Return of Investment*.

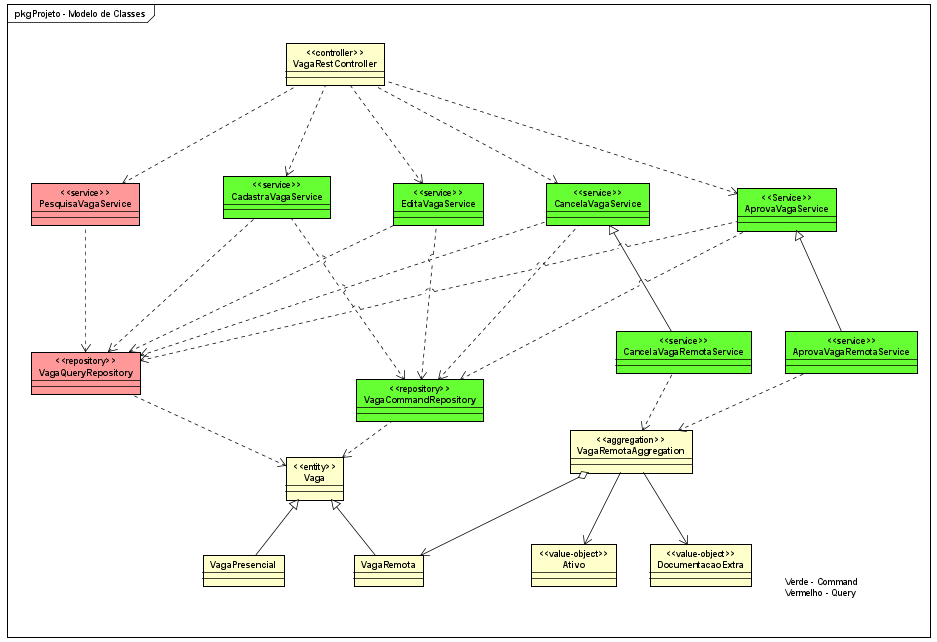
## *Modelo de Classes de Projeto (UML)*

Utilizando os conceitos de *CQS* para separar as responsabilidades de classes que fazem apenas leitura e atualização na camada de persistência.

### Diagrama de Classes do PerfilMicroservice



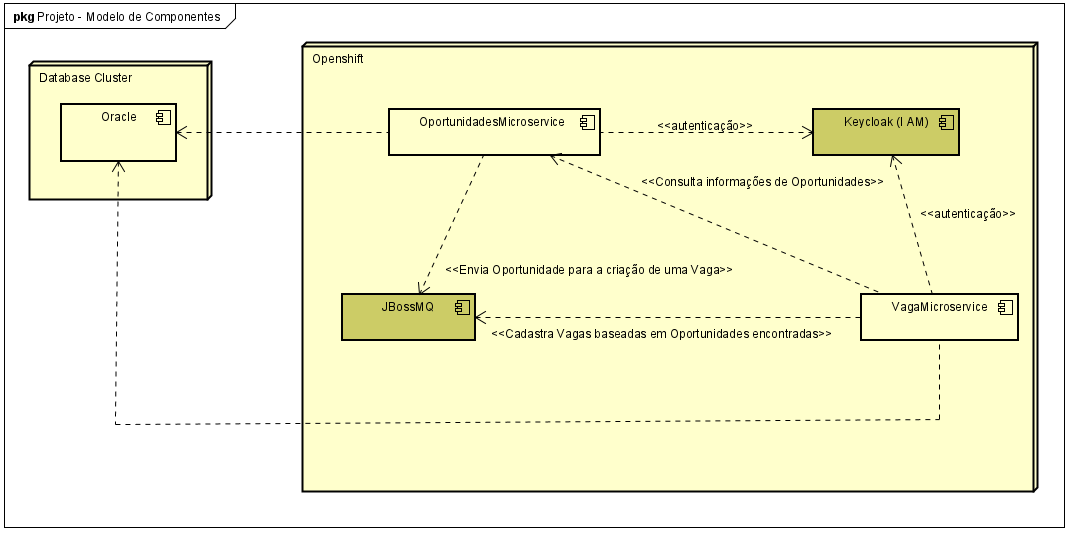
### Diagrama de Classes do VagaMicroservice



## Modelo de Comunicação e Implantação

### Diagrama de Comunicação e Implantação dos componentes (*UML*)

Utilização do Diagrama de *Deployment* da UML para representar a comunicação dos componentes arquiteturais e locais onde executam. Vale ressaltar a preocupação com a visão de dependências entres os componentes e o quanto isso define a qualidade da solução em termos de acoplamento e coesão (referência ao Princípio de *Hollywood*: *Tell, don’t ask*!).



### Microservice Canvas

Simplificação textual proposta em adaptação de outro Canvas (*Microservice Design Canvas*), trata-se de um Canvas que visa documentar todos os aspectos de um serviço. Segue um exemplo do serviço de Vagas:

|  |  |  |
| --- | --- | --- |
| **Name** | VagaMicroservice | |
| **Description** | O serviço de Vagas provê uma API para manter as informações identificadas por Analistas ou rotinas automáticas e que poderão tornar-se Processos Seletivos. | |
| **Capabilities** | | |
| * Captação de Oportunidades * Cadastro de Vagas * Aprovação de Vagas * Disparar a criação de Processos Seletivos | | |
| **Service API** | | |
| Commands | Queries | Events Published |
| Synchronous | * Consulta de Vagas em que o Analista está responsável * Consulta de Vagas por Cliente * Consulta de Vagas Aprovadas para iniciar Processo Seletivo | * Vagas Aprovadas * Vagas Canceladas |
| * Consulta de Vaga(s) * Cadastro de Vaga * Atualização de Vaga * Cancelamento de Vaga |
| Asynchronous |
| * Criação de Vaga através de Oportunidade identificada |
| Non-functional requirements | * Disponibilidade 99,9% \* | |
| **Observability** | | |
| Key metrics | | |
| * Vagas criadas * Vagas aprovadas * Vagas canceladas | | |
| Health check endpoint | /actuator/health (*Spring Boot Actuator*) | |
| **Implementation** | | |
| * *Design*   + Baseado em modelo de domínio criado com práticas de *DDD, CQS, CQRS, S.O.L.I.D* e *Tiny Types* * Lógica   + Fluxo de criação, aprovação ou cancelamento de vagas * Dados   + Informações de Oportunidades para criação de Vagas   + Informações sobre Vagas   + Informações base para Processos Seletivos * Qualidade   + Volume de alto de consumo * Segurança   + Permissões via autenticação e autorização de Token | | |
| **Dependencies** | | |
| Invokes | Subscribes to | |
| * I AM   + Autorização * OportunidadeMicroservice   + Recupera informações detalhadas de Oportunidade | * Broker   + Evento de Vaga criada   + Evento de Vaga cancelada | |

\* Essas definições são baseadas em Bechmarks e SLAs, sendo apoiadas por ferramentas de coletas de métricas e alguns cálculos específicos.

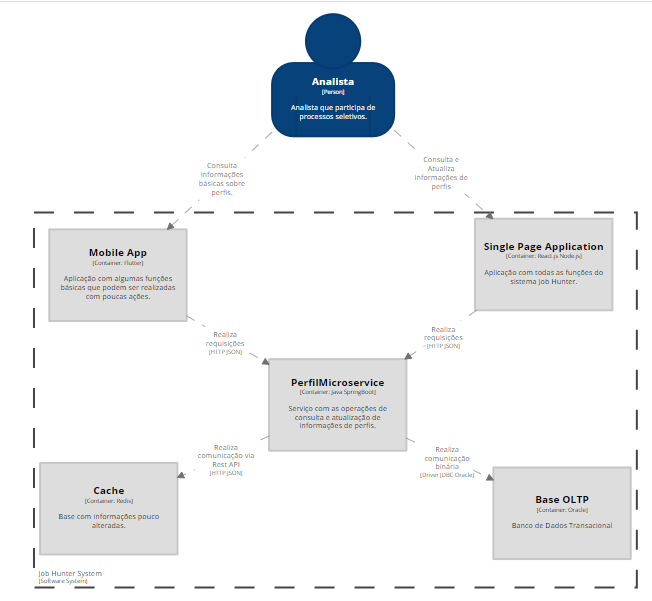
### *C4 Model*

Utilização dessa abordagem de diagramação simples e focada na comunicação entre Arquitetos e Desenvolvedores. Modelo que pode ser criado por ferramenta gráfica, via *DSL* ou *APIs* nativas para linguagens de programação. No repositório de códigos utilizado no trabalho foi criado um exemplo de geração e os outros criados na ferramenta *Structurzr* do autor do modelo.

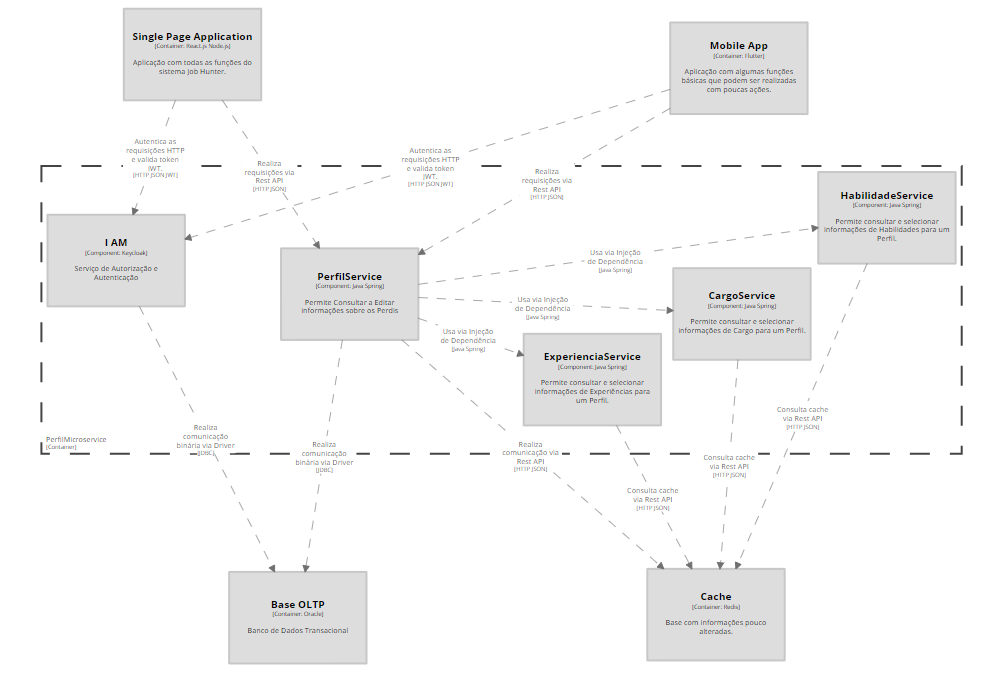
São listados abaixo alguns recortes do sistema nos 4 aspectos propostos pelo modelo:

#### 1º Nível – Contexto

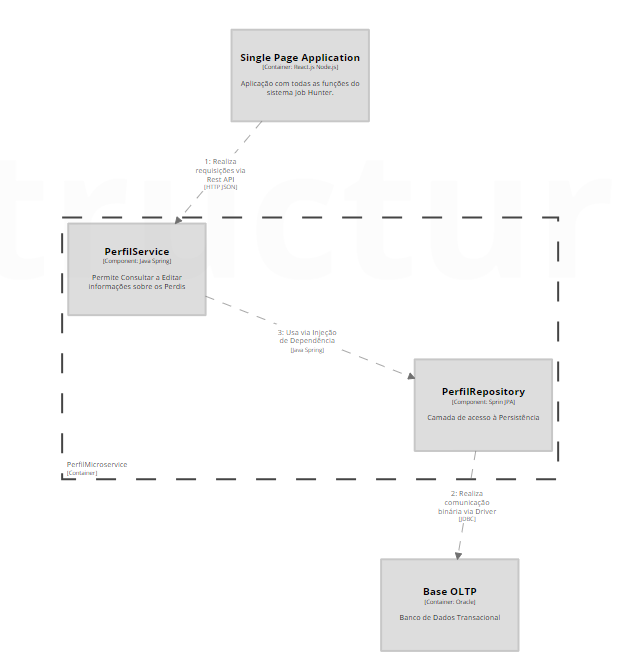
#### 2º Nível – Containers



#### 3º Nível – Componentes



#### 4º Nível – Código



### Cartão *CRC*

|  |  |
| --- | --- |
| PesquisaPerfilService | |
| * Recuperar Perfis desejados pelos clientes * Listar Perfis para criação de Vagas | * Cliente * Vaga |

# Exemplos de Implementação

## Justificativa

O foco do trabalho foi em documentação e pesquisa de boas práticas que guiem a criação de bons microsserviços.

Apesar de não ter sido criado um microsserviço de ponta-a-ponta com todos os seus recursos e publicação em plataforma de nuvem, foram implementados algumas propostas de solução de design de código que servem de guia para todas as demais definições, tanto de mais alto (arquitetura), quanto de mais baixo (classes) nível.

### Código

O código implementado para validar a proposta desse trabalho encontra-se no GitHub do autor: <https://github.com/cirilojunior/infnet-mit-arquitetura-bloco-microservicos>.

Toda documentação produzida também se encontra no repositório.

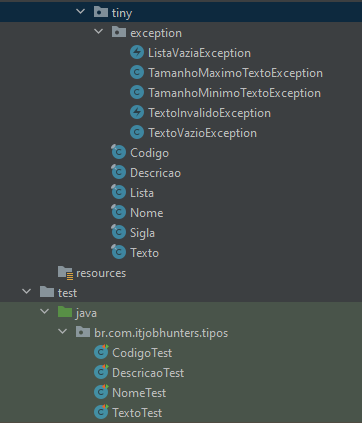
### Arquitetura e Design do Código

Os exemplos iniciam com a criação de um componente (*lib* Java) de software base, mas que acabou evoluindo para uma espécie de *framework*.

A codificação começou com a definição de uma *lib* de *tiny types* que seria utilizado por todos os novos softwares da empresa. Apesar de um esforço inicial, se justifica pela expressividade alcançada nos códigos de negócio e pela não repetição (*DRY*) de códigos *boillerplates, como por exemplo: tratamentos de* exceções e validações de nulo ou vazio.

Com a evolução da pesquisa, foram sendo incorporados classes base baseadas nos *building blocks* do *DDD*, assim como nos conceitos de *CQS* que são base para a construção serviços aderentes ao padrão *CQRS*.

#### Tiny Types

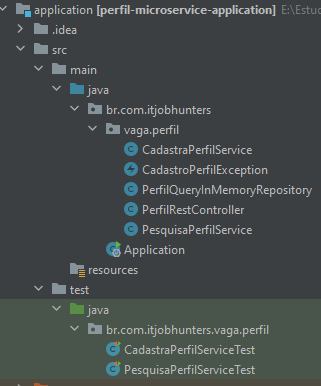




#### Domain-Driven Design, S.O.L.I.D, CQS e CQRS

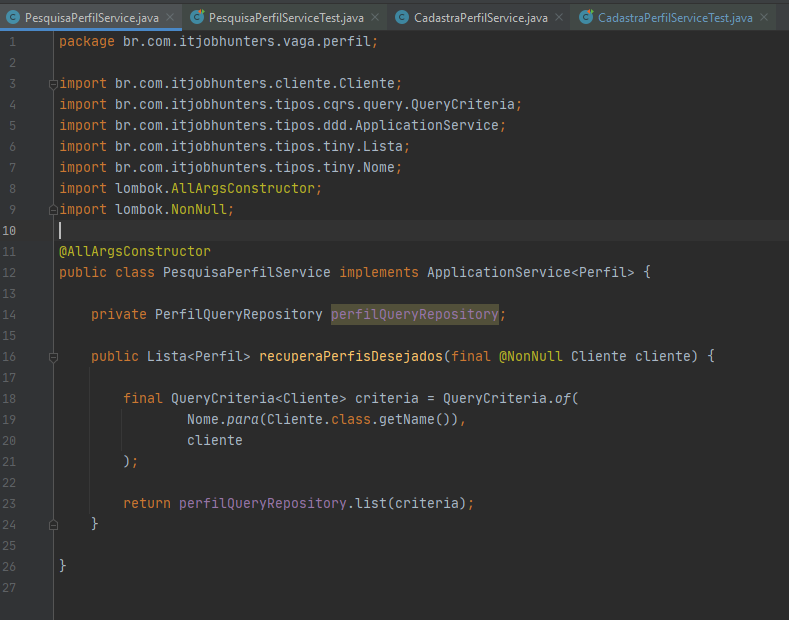
Foram utilizados esses conceitos no *design* das classes e nos *endpoints* dos microsserviços. Os conceitos se mesclaram no design das classes, evoluindo por exemplo, o *Repository* do *DDD* em *QueryRepository* e *CommandRepository*. É possível perceber também a influência de *S.O.L.I.D*, uma vez que encontramos classes com responsabilidade única, inversão de dependência (permitindo a injeção em classes e permitindo desacoplamento fácil entre microsserviços).

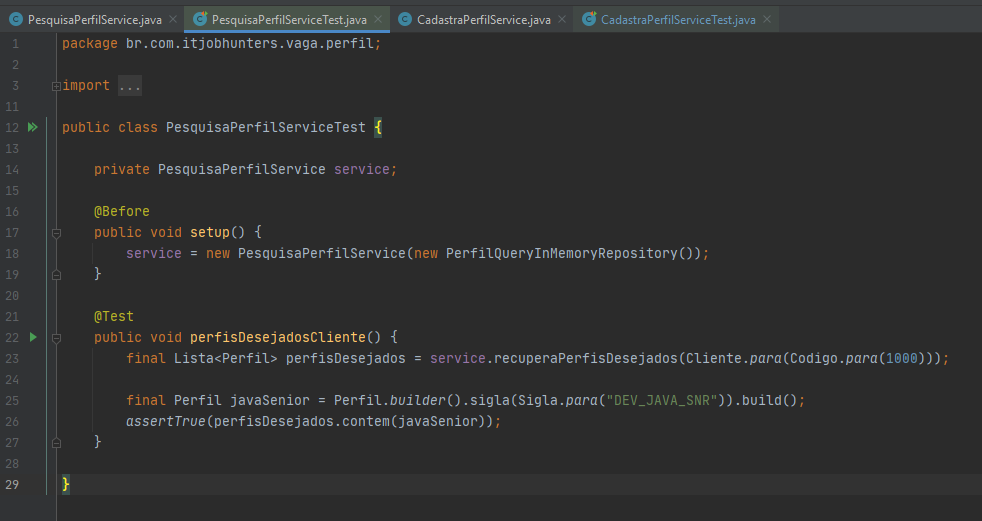
As classes herdam ou implementam\* abstrações que marcam sua responsabilidade influenciando em comportamentos e na própria organização e documentação do código. Um exemplo é que a camada de aplicação importa a camada de domínio e injeta suas implementações de *Repositories* (por exemplo, *Spring Data JPA*) em *Services*.



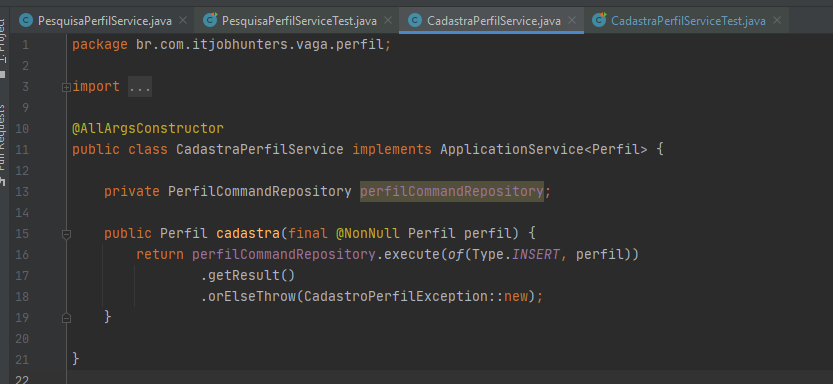
\* Outra opção seria a criação de Anotações.

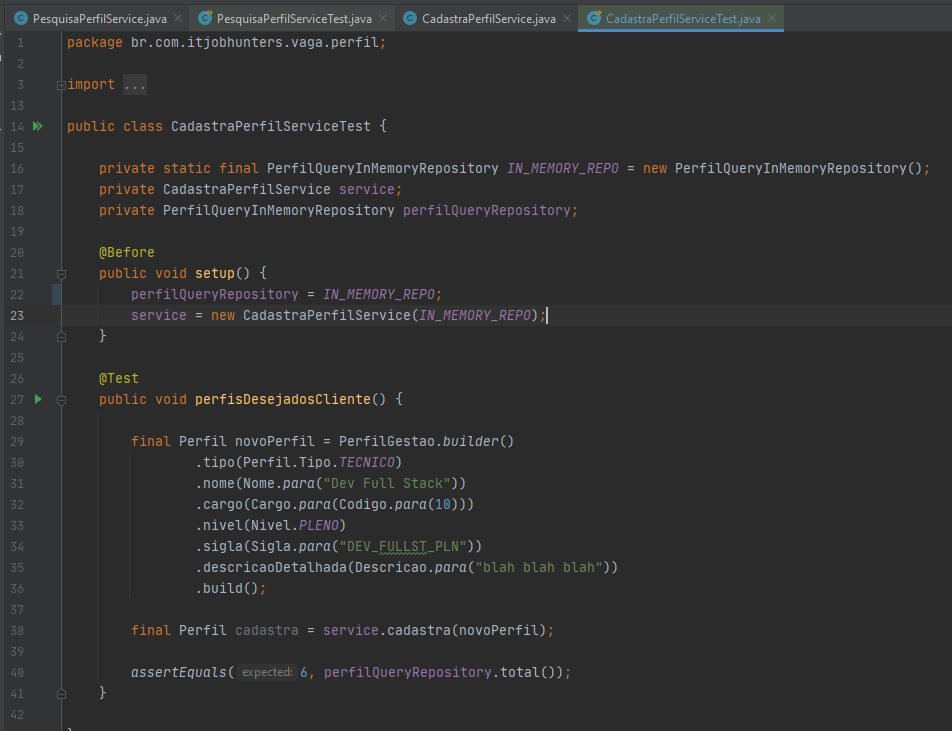
##### Classe de Serviço de Pesquisa de Perfis utilizando os padrões abordados





##### Classe de Serviço de Pesquisa de Perfis utilizando os padrões abordados





### Documentação de Uso dos Microsserviços

Para documentar as APIs para utilização será utilizado *Swagger* que mantém uma página de documentação com possibilidade de execução dos métodos dos serviços.

### Implantação

Será utilizada a plataforma *Openshift* para *deploy* das aplicações, inicialmente *on-premise* na versão *Community*. Após a finalização de todos os módulos inicialmente planejados, existe um plano de contratação do serviço em um dos planos de nuvem\*. A migração das aplicações nesse contexto fica facilitada e reduz o *vendor lock-in* de provedores de nuvem. A dependência do próprio *Openshift* se justifica por todas as facilidades que ele oferece sobre a administração e gestão das aplicações por uma equipe com cultura *DevOps\*\**.

\* nas referências existe um *link* para as modalidades utilizando AWS, Google, Azure ou IBM.

\*\* A plataforma simplifica bastante todo o processo de *Continuous Integration*, *Continuous Delivery* através de um modelo de *build pipeline* obtido diretamente do repositório de código e criada através um catálogo opções de *containers* previamente configurados.

### Comunicação

A comunicação assíncrona entre os microsserviços se dará pelo *broker* padrão oferecido pela plataforma *Openshift*: *JBossMQ* (baseado no Apache Artemis).

Num primeiro momento, conforme já informado, algumas informações serão obtidas das bases corporativas atuais e na medida em que o novo sistema for sendo incrementado, essa comunicação será desligada.

Não será feita a implementação, apenas representação do componente arquitetural.

### I AM (autenticação e autorização)

Como solução de *Identity Server*, foi selecionado o produto Keycloak também fornecido pela plataforma *Openshift*.

Não será feita a implementação, apenas representação do componente arquitetural.

# Conclusão

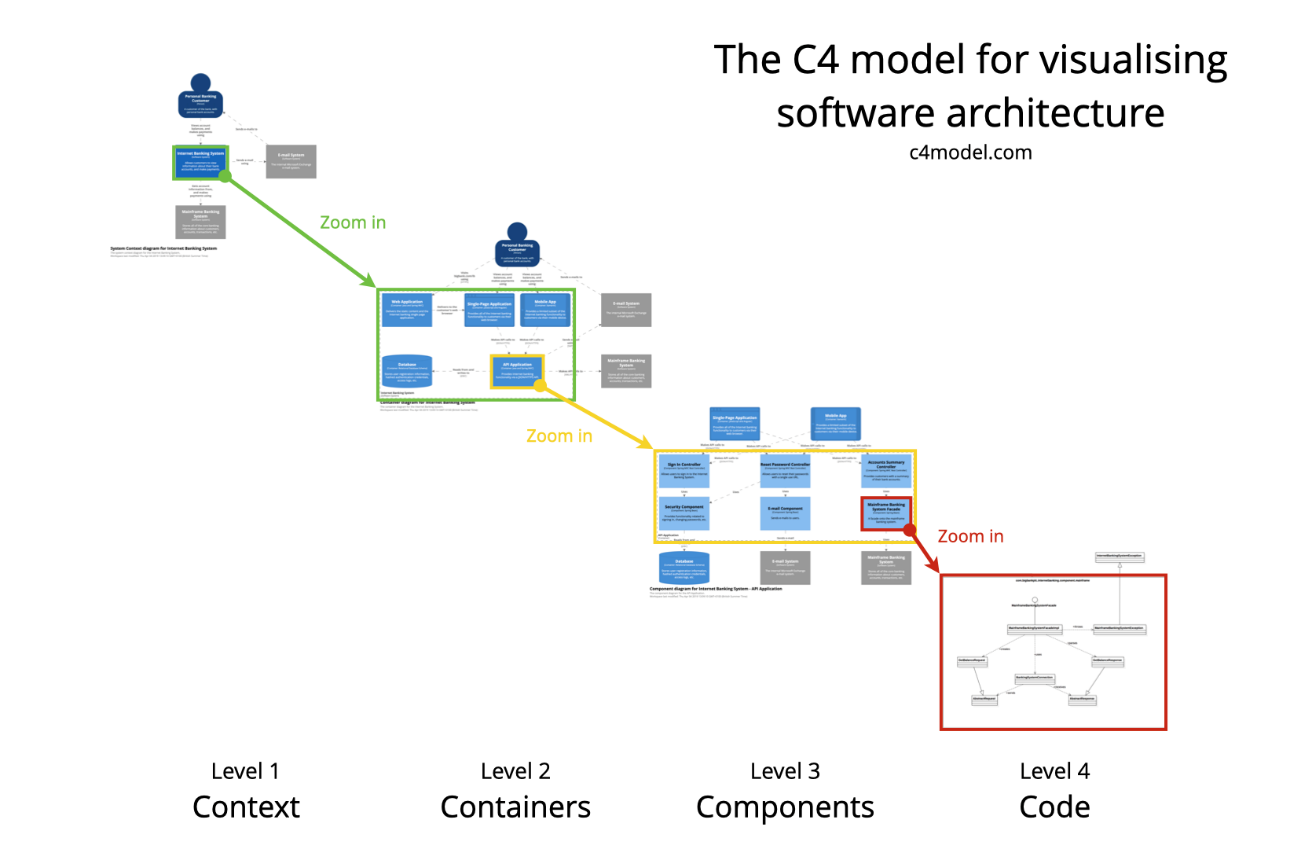
Neste trabalho abordamos formas de documentação de especificação de microsserviços, assim como toda a base de código nos níveis inferiores e concluímos que se prender a um determinado padrão ou método, além de tornar o trabalho mais burocrático, tira o principal valor da documentação que é facilitar a comunicação entre os envolvidos no desenvolvimento guiando-os de forma segura, sem ambiguidades e fácil de ajustar quando necessário.

Foram aplicadas várias técnicas de engenharia de softwares de acordo com os contextos de análise, projeto e implementação, além da criação de código que exemplifiquem as conclusões alcançadas.

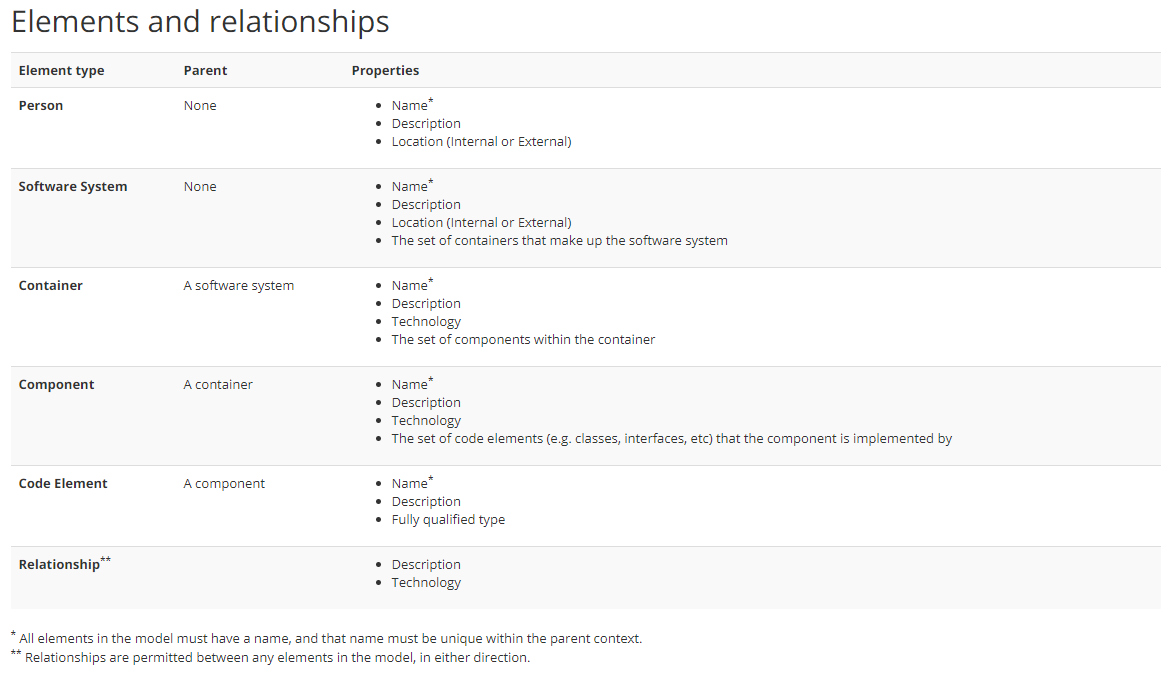
Esse trabalho contribuiu para o conhecimento de novas técnicas que serão aplicadas em futuros desenvolvimentos.

# Referências

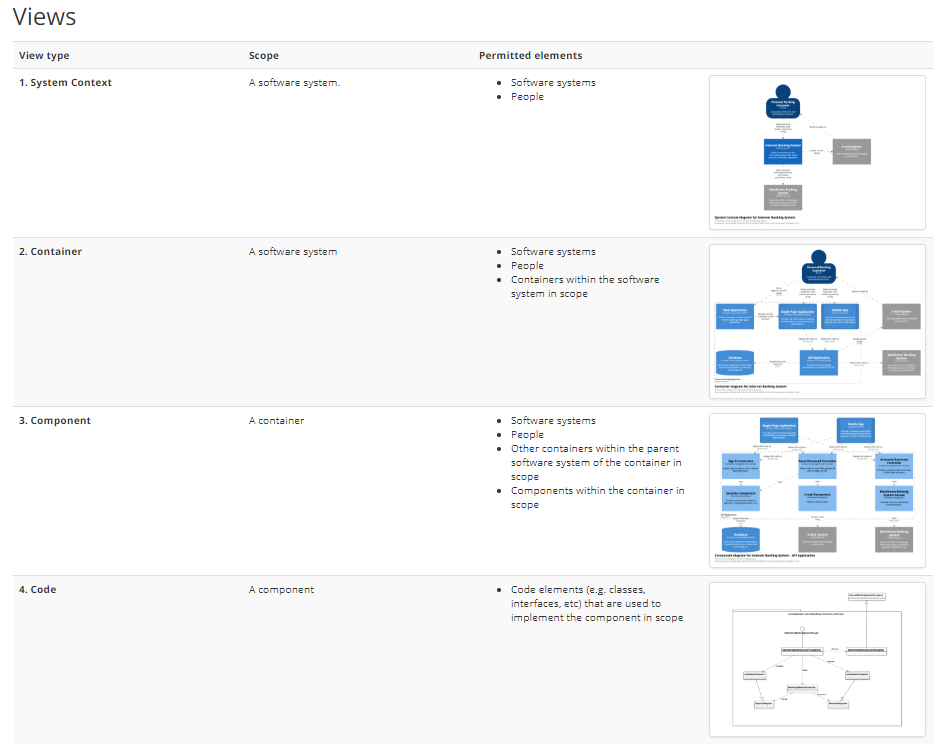
* Microsserviços
  + <https://chrisrichardson.net/post/microservices/general/2019/02/16/whats-a-service-part-1.html>
  + Microservice Canvas:
    - <https://chrisrichardson.net/post/microservices/general/2019/02/27/microservice-canvas.html>
    - <https://dzone.com/articles/streamlined-microservice-design-in-practice>
* Modelagem de *Software* com *RUP* e *UML*.
  + RUP
    - Entendimento das fases de um processo de desenvolvimento de *software*.
    - Fonte: <https://www.devmedia.com.br/rup-rational-unified-process/4574>
  + Documento de Visão e *Problem Statement*:
    - <https://www.ibm.com/support/knowledgecenter/en/SSWMEQ_4.0.6/com.ibm.rational.rrm.help.doc/topics/r_vision_doc.html>
  + UML as Sketch
    - Uso de *UML* de forma menos burocrática e mais produtiva com foco na solução e não na documentação em si.
    - Fontes:
      * <https://martinfowler.com/bliki/UmlMode.html>
      * <https://martinfowler.com/bliki/UmlAsSketch.html>
* *C4 Model*
  + Site oficial: <https://c4model.com/> (fonte das imagens)
    - Resumo
      * Visualização do Modelo:



* + - * *Metamodel*:



* + - * *View*:



* + *Structurizr*
    - <https://structurizr.com/>
    - Ferramenta para a criação de modelos no padrão *C4 Model*
* *GoF Patterns*
  + <https://www.devmedia.com.br/design-patterns-padroes-gof/16781>
  + Utilizado o *pattern Strategy* como guia para a solução dos buscadores.
* I AM
  + <https://www.keycloak.org/>
* Comunicação assíncrona entre serviços
  + <https://www.redhat.com/pt-br/technologies/jboss-middleware/amq>
* Openshift
  + <https://www.openshift.com/>
  + <https://www.openshift.com/products>
  + Plataforma de *containers* de aplicação *opensource* da [*Red Hat*](https://www.redhat.com/en).
* Lombok
  + <https://projectlombok.org/>
  + *Lib* utilizada na geração de código comum que permite que o fonte fique menos poluído.
* *CQS*
  + <https://martinfowler.com/bliki/CommandQuerySeparation.html>
  + Padrão de *design* que define a separação de métodos de Consulta e Alteração de forma clara.
* *CQRS*
  + <https://martinfowler.com/bliki/CQRS.html>
  + Padrão arquitetural baseado no *CQS*, mas agora pensando na questão arquitetural, ou seja, relacionado à requisições via rede.
* *S.O.L.I.D*
  + <https://blog.caelum.com.br/principios-do-codigo-solido-na-orientacao-a-objetos/>
  + Acrônimo de características desejadas para que o código obtenha um bom padrão de qualidade, sendo mais fácil de passar por evoluções.
* *Tiny Types*
  + <https://blog.caelum.com.br/pequenos-objetos-imutaveis-e-tiny-types/>
  + Estilos de codificação que visa expressividade e onde tentar-se criar os tipos o mais próximo possível do conceito representado, através de classes *wrappers* para os tipos comuns da linguagem, como por exemplo: String e List.
* *DRY (Don’t Repeat Yourself)*
  + <http://www.macoratti.net/16/04/net_dry1.htm>
  + Princípio que busca a não repetição de código.
* *Boillerplate*
  + <https://pt.stackoverflow.com/questions/10575/o-que-%C3%A9-boilerplate-code>
  + Código repetido pela estrutura da linguagem.
* Princípio de *Hollywood*
  + <https://www.devmedia.com.br/as-leis-do-mundo-dos-objetos-melhores-praticas-em-orientacao-a-objetos/26588>
  + Prática relacionadas à inversão de dependências.
* *CRC Cards*
  + <http://agilemodeling.com/artifacts/crcModel.htm>
  + Estilos de documentação de classes (*Class*, *Resposabilities* e *Collaborators*).
* Requisitos não funcionais:
  + <https://www.altexsoft.com/blog/non-functional-requirements/>
  + <https://www.scaledagileframework.com/nonfunctional-requirements/#:~:text=Nonfunctional%20Requirements%20(NFRs)%20define%20system,system%20across%20the%20different%20backlogs.&text=They%20ensure%20the%20usability%20and%20effectiveness%20of%20the%20entire%20system>.
  + <https://uptime.is/>
* Spring Boot Actuator
  + <https://docs.spring.io/spring-boot/docs/current/reference/html/production-ready-features.html>
* *Swagger*
  + <https://swagger.io/>