Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

CELIO ROBERTO BALANCIERI

MICRO SERVIÇOS COM SPRING BOOT

Assis/SP
2021

Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

CELIO ROBERTO BALANCIERI

MICROSERVIÇOS COM SPRING BOOT

Projeto de pesquisa apresentado ao curso do Instituto Municipal de Ensino Superior de Assis – IMESA e a Fundação Educacional do Município de Assis – FEMA, como requisito para a obtenção do Certificado de Conclusão.

Orientando: Celio Roberto Balancieri
Orientador: Domingos de Carvalho Villela Junior

Assis/SP
2021

FICHA CATALOGRÁFICA

B171m BALANCIERI, Celio Roberto

Microsserviços com spring boot / Celio Roberto Balancieri. – Assis, 2021

42p

Trabalho de conclusão do curso (Análise e Desenvolvimento de Sistemas). – Fundação Educacional do Município de Assis - FEMA

Orientador: Esp. Domingos de Carvalho Villela Júnior

1.Microsserviços 2.Spring boot

CDD005.133

MICRO SERVIÇOS COM SPRING BOOT

CELIO ROBERTO BALANCIERI

Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de Ensino Superior de Assis, como requisito do Curso de Graduação, avaliado pela seguinte comissão examinadora:

Orientador: Domingos de Carvalho Villela Junior

Examinador: Osmar Aparecido Machado

Assis/SP
2021

DEDICATÓRIA

Dedico este trabalho a minha amada e querida esposa Ângela e aos meus queridos filhos Kailaine e Miguel amigos e familiares.

AGRADECIMENTOS

Em primeiro lugar agradeço a Deus, meus familiares e amigos que acreditaram e me deram força para chegar até o fim desta jornada e os professores com sua paciência e dedicação em ensinar.

De um modo especial e carinhoso quero agradecer a minha esposa que presenciou neste decorrer de tempo as minhas dificuldades, mas sempre ao meu lado me dizendo você vai conseguir não desista iremos vencer juntos.

Aos meus filhos agradeço pela compreensão paciência de não poder em certos momentos de estudo estar presente em suas atividades ou brincadeiras.

Aos meus pais que sempre me conduziram para o bom caminho e ensinou que dedicação aos estudos sempre edifica o nosso caráter.

Também quero agradecer em especial o professor Drº Prof. Almir Rogério Camolesi que deixa como exemplo de vida que devemos sempre superar nossas dificuldades.

Viva como se fosse morrer amanhã. Aprenda como se fosse viver para sempre

Mahatma Gandhi

RESUMO

No cenário que estamos vivenciando nos dias atuais a crescente demanda por tecnologia em todo momento cresce a necessidade de consultas ao bancos de dados e maior tanto na abstração de uma informação ou para gravar um registro, a este contexto responsabilidade e do profissional de tecnologia da informação solucionar as constantes e infinitas variáveis nas equações decorrentes do consumo de tantos serviços relacionados na arquitetura e desenvolvimento de programas capazes de trazer a informação desejada para seu usuário.

A esta demanda grandiosa por serviços e processamento de informações as tecnologias evoluem a um ritmo acelerado e o profissional da área deverá ser capaz de identificar as tendências e atualizações para o desenvolvimento de seus projetos.

Para tal situação iremos abordar o Framework Spring Boot, no desenvolvimento de aplicação capaz de trabalhar com micro serviços descompactados no contexto Rest Full a fim de explorar um pouco mais amplo a ferramenta usando a tecnologia JAVA no seu desenvolvimento.

;

Palavras-chave: MICROSSERVIÇOS; SPRING BOOT;

ABSTRACT

I the scenario we are experiencing thease days, the growing demand for technology at all times, the need for database queries graws, and greater both in the abstraction of infornation or to record a record, in this contexto, ther responsibility of the information technology prefissional to solve the constant and infinite variables in the equations resulting from the consumpion of so many services related to the architecture and development of programs capable of briging the desired information to its user.

To this great demand for services and information processing Technologies evolve at a fast pace and the professional in the área should be able to indentity treands and updates for the development of their projects.

For this situation, we will approach the Spring Boot Framework, in the development of na application capable of working with umconpressed micro services in the Rest Full contexto, in order to explore the tool a little winder using JAVA technology in its development.

Keywords: MICROSSERVECES; SPRING BOOT;

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

SOAP – Simple Object Access Protocol

HTML – Hypertext Markup Language

XML – Extensível Markup Language

JASON - Java Script Object Notation.

HTTP – Hyper Text Transfer Protocol

SUMÁRIO

1. INTRODUÇÃO

A evolução na arquitetura e desenvolvimento de sistemas tem passado por constante evolução ao um ritmo muito acelerado ao que possamos imaginar talvez pela percepção humana, diante deste cenário cada vez mais e exigíveis profissionais da área de tecnologia que sejam capazes de avaliar e propor soluções mais eficazes escaláveis e de baixo custo presente e funcionando em diversos e variados tipos de aparelhos eletrônicos computadores, celulares, notebooks, tabletes, televisão possivelmente vinte e quatro horas por dia sete dias por semana.

A demanda da necessidade deste tipo e serviço e de grande escala diante do cenário que vivemos hoje no mundo porem, precisamos despertar nosso foco de estudo e pesquisa para a seguinte colocação a evolução da Arquitetura Monolítica para Micro Serviços.[1]

Além da evolução da arquitetura de software também cresce juntamente os métodos e as ferramentas de desenvolvimento na produção e integração dos softwares tendo hoje no mercado uma ampla e variada oferta de tecnologias para desenvolvimento de aplicações neste momento focaremos o estudo do uso e Frameworks[2] exemplificando o uso destes métodos para construir e usar micro serviços na rotina diária e funcionamento destas aplicações.

[1] Disponível em < https://www.opus-software.com.br/micro-servicos-arquietura-monolitica/>
[2] Disponível em https://pt.wikipedia.org/wiki/Framework >

1.1. OBJETIVOS

Elaborar um estudo de caso para explorar o uso conceitual e prático da metodologia de micro serviços e uso do framework Spring Boot juntamente com o JAVA no desenvolvimento de sistemas com grande escalabilidade dinâmicos e capazes de ser adaptado e integrado com a evolução tecnológica decorrente do contexto deste século revolucionário área da Tecnologia da Informação e Desenvolvimento de Softwares.

Com este objetivo e esperado que este trabalho venha incentivar e encorajar os profissionais da área a se atualizar com as tecnologias existentes hoje no mercado de hoje para área de informática.

1.2. PUBLICO ALVO

Este trabalho tem como público alvo profissionais da área de Desenvolvimento de Softwares estudantes para que possam aprofundar os seus conceitos com novas tecnologias e quebrar o paradigma e dificuldade na integração de novas tecnologias.

1.3. JUSTIFICATIVA

A justificativa para abordagem escolhida neste trabalho, do estudo de caso no desenvolvimento de softwares com a utilização micro serviços e framework Spring Boot e para mostrar e desertar a iniciativa na adoção de práticas ao desenvolvimento mais ágil descompactado de sistemas monolíticos que na maioria dos casos são engessados e não permitem

muitas alterações e customização com integração rápida e fácil a novas tecnologias.

1.4. ESTRUTURA DO TRABALHO

Este trabalho está dividido em capítulos. O primeiro capítulo apresenta introdução, objetivo, público alvo e justificativa para o desenvolvimento deste trabalho.

O segundo capítulo apresenta as tecnologias e ferramentas utilizadas para o

Desenvolvimento do trabalho.

O terceiro capítulo destinada a análise e a especificação do sistema, o levantamento de requisitos, mapa mental, lista de eventos, e a modelagem

No quarto capítulo conclusão e apresentação de trabalhos futuros.

2. TECNOLOGIA FERRAMNETAS PARA DESENVOLVIMENTO.

2.1 JAVA

A Sun anunciou o Java formalmente em uma conferência do setor em maio de 1995. O Java chamou a atenção da comunidade de negócios por causa do enorme interesse na Web. O Java agora e utilizado para desenvolver aplicativos corporativos de grande porte, aprimorar as funcionalidades de servidores Web (os computadores que fornece o conteúdo que vemos em nossos navegadores da Web), fornece aplicativos para dispositivos voltados

para o consumo popular (como telefones celulares, pages e PDAs) e outros propósitos (DEITEL, 2010, p.06

2.2 POSTGRE SQL

E um sistema gerenciador de banco de dados objeto relacional (SGBD), desenvolvido como projeto de código aberto.

O PostgreSQL é um dos resultados de um, um dos pioneiros a ampla evolução que se iniciou com o projeto Ingres, desenvolvido na Universidade de Berkeley, Califórnia. O líder do projeto, Michael Stonebraker, um dos pioneiros em banco de dados relacionais.[3]

2.3 INTELLIJ IDEA

IntelliJ IDEA é um ambiente de desenvolvimento integrado (IDE) escrito em Java para desenvolvimento de software de computador. E desenvolvida pela JetBrais (anteriormente conhecida como intelliJ), e está disponível como uma edição da comunidade licenciada pela Apache e em uma edição comercial proprietária. Ambos podem ser usadas em desenvolvimento comercial.[4]

[3] Disponível em https://pt.wikipedia.org/wiki/PostgreSQL

[4] Disponível em https://en.wikipedia.org/wiki/IntelliJ_IDEA

2.4 MICRO SEVIÇOS

Quem já trabalhou com o Java 5 ou com versões anteriores provavelmente lembra das aplicações grandes e pesadas que engoliam o hardware do servidor e, por uma pequena falha em uma parte da aplicação, comprometendo todo o sistema. Em uma aplicação nova não podemos mais pensar desta maneira, más e interessante entender e aprender com os erros do passado.

Antigamente as empresas que adotavam em massa um tal de JAVA. Ele conversava com qualquer banco de dados e até mainframe, fazia transações distribuída e tornava qualquer plataforma confiável para executar os seus sistemas. Depois do decimo sistema feito por diferentes equipes, foi descoberto que todos tinham o mesmo cadastro de clientes. O que começou a surgir nessas empresas foi o velho (e como é velho) problema do reaproveitamento de código.

Que tal isolar essa parte comum em um sistema único?

Assim na virada do século, os serviços (internos e externos) começaram a usar um padrão de comunicação via XML, chamado Object Acces Protocol (SOAP) ou, em português, protocolo de acesso a objetos simples. Com ele, sistemas começaram a trocar informações, inclusive de diferentes linguagens e sistemas operacionais. Foi sem dúvida uma revolução.

Começava a era da arquitetura orientada a serviços, conhecida como SOA, que padronizava essa comunicação entre diferentes serviços.

O problema do padrão SOAP, é a complexidade em fazer qualquer coisa, como, por exemplo realizar um serviço de consulta que retorne qualquer valor simples. Isso tem muita abstração envolvida com o servidor de um lado e obrigatoriamente um cliente do serviço do outro, trafegando XML para

ambos os lados. E tudo isso em cima de um protocolo usado na internet (HTTP).

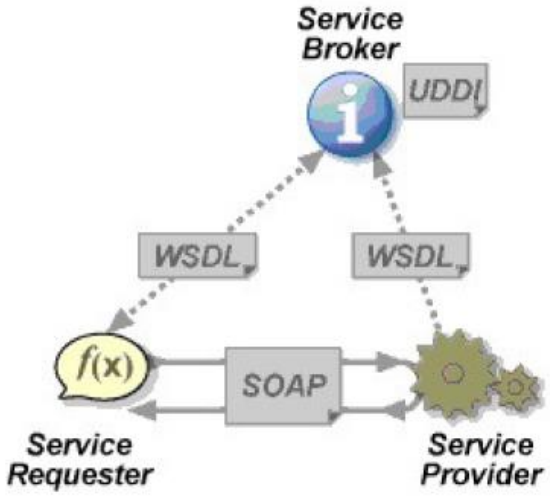

Figura 1: Spring Boot Acelere o Desenvolvimento de Microsserviços (In: BOAGLIO et al.; 2021 p 14)

No decorrer do tempo, muitos serviços SOAP, usados por vários sistemas, começaram a aparecer rapidamente como a principal causa de lentidão, obrigando os programadores a procurar alternativas, como trocar o serviço SOAP por um acesso direto ao banco de dados ou a um servidor Active Directory. Com esse cenário, Roy Thomas, um dos fundadores do projeto Apache HTTP, defendeu uma tese de doutorado apresentando uma alternativa bem simples: o famoso Representational State Transfer (Transferência de Estado Representacional), ou simplesmente REST (ou ReST). Essa simples alternativa ao SOAP aproveita os métodos existentes no protocolo HTTP para fazer as operações mais comuns existentes nos serviços, como busca de cadastro.

Assim, por sua simplicidade e rapidez, esse padrão foi adotado rapidamente pelo mercado.

Abaixo representação gráfica do REST.

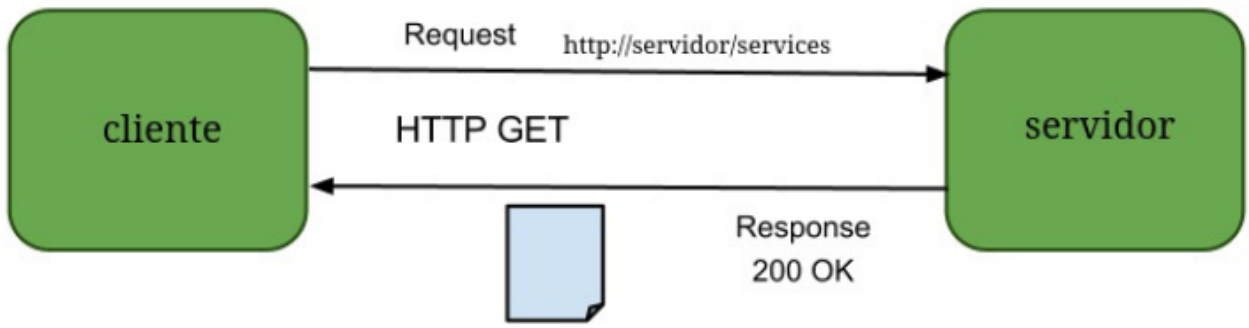

Figura 2: Spring Boot Acelere o Desenvolvimento de Microsserviços (In: BOAGLIO et al.; 2021 p 15)

Mesmo com os serviços ReST, as aplicações continuaram a empacotar todas as funcionalidades em um só lugar, sendo classificadas como aplicações monolíticas. Fazendo uma analogia de funcionalidades com um brinquedo, independente da demanda. Em um cenário com muitas crianças que só brincam com bonecas, sobrarão muitos aviões.

Para resolver esse problema destaca-se um subconjunto da arquitetura SOA chamado micros serviços (microservices), que abraça a solução ReST com o objetivo de fornecer uma solução separada e independente para um problema.

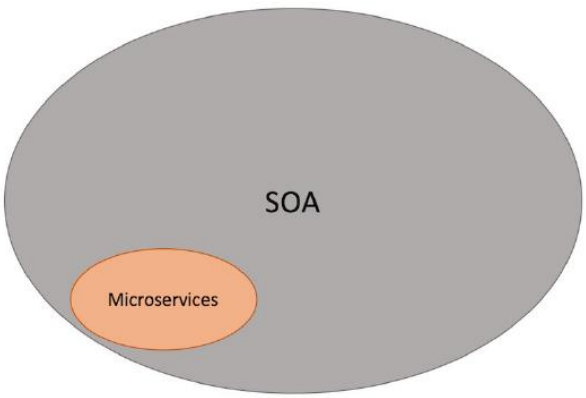

Figura 3: Spring Boot Acelere o Desenvolvimento de Microsserviços (In: BOAGLIO et al.; 2021 p 17)

O ideal de uma aplicação é separar suas funcionalidades e se adequar ao cenário. Com o conceito de microsserviços, cada funcionalidade é independente e podemos crescer a sua quantidade conforme a demanda de nosso cenário. (BOAGLIO,2021, p.13, 14, 15.)

2.5 SPRING BOOT FRAMEWORK

E uma maneira eficiente e eficaz de criar uma aplicação em Spring e facilmente coloca-la no ar, funcionando sem depender de um servidor de aplicação. O Spring Boot criou um conceito novo, não existe ate o momento na especificação JEE. (BOAGLIO,2021, p.17,18.)

2.6 FRAMEWORK SWAGGER

O Swagger é um framework composto por diversas ferramentas que, independente da linguagem auxilia a descrição, consumo e visualização de serviços de uma API REST.

Uma boa documentação de uma API REST é crucial para desenvolvedores, testadores mantenedores entenderem claramente o comportamento fornecido por ela. Então em questão relevante e: De como fazer esta documentação da melhor forma possível.[5]

[5] Disponível em http://www2.decom.ufop.br/terralab/documentando-sua-api-rest-com-swagger/

2.7 DOCKER

Docker é um conjunto de produtos de plataforma como serviço (PaaS) que utilizam virtualização de nível de sistema operacional para entregar software em pacotes chamados contêineres. Os contêineres são isolados uns dos outros e agrupam seus próprios softwares, bibliotecas e arquivos de configuração. Eles podem se comunicar uns com os outros por canais bem definidos. Todos os containers são executados por um único kernel do sistema operacional e, portanto, usam menos recursos do que as maquinas virtuais.[6]

3. FUNDAMENTAÇÃO TEORICA

Neste capitulo será apresentado a base teórica a qual a Proposta de Trabalho se embasa, afim de possibilitar uma melhor compreensão do que será exposto posteriormente no trabalho.

3.1 INTRODUÇÃO A ARQUITETURA DE MICROSERVICES COM SPRING BOOT.

Uma das funções mais importantes e desafiadoras de um arquiteto de softwares é a definição da arquitetura para o sistema a ser desenvolvido.

[6]Disponívelemhttps://pt.wikipedia.org/wiki/Docker_(software)#:~:text=Docker%20%C3%A9%20um%20conjunto%20de,bibliotecas%20e%20arquivos%20de%20configura%C3%A7%C3%A3o.

Iniciado entre as primeiras fases do projeto, esta etapa, é fundamental para o bom andamento do mesmo. Para se ter uma noção mais precisa, uma decisão incorreta pode ocasionar atrasos na entrega do sistema, comprometer a qualidade, a segurança ou até mesmo inviabilizar o desenvolvimento.

Dada a importância dessa fase para a fundamentação do sistema, inúmeros padrões de desenvolvimento, frameworks e modelos arquiteturais são elaborados para desenvolver problemas comuns utilizando melhores práticas. Neste cenário, um dos padrões arquiteturas que mais tem se destacado é a arquitetura de microsserviços, justamente pelos benefícios adquiridos ao adotar esta solução.

Sistemas construídos sobre o padrão arquitetural monolítico são compostos basicamente por módulos ou funcionalidades agrupadas que, juntas compõem o software. Podemos citar como exemplo de aplicações que fazem o uso do modelo monolítico ERP (Enterprise Resorse Planing) como o SAP ou Protheus. Estes possuem, normalmente módulos para controle do setor financeiro, contábil, compras entre outros agrupados numa única solução que acessa o banco de dados.[7]

[7] Disponível em <https://www.devmedia.com.br/introducao-a-arquitetura-de-microservices-com-spring-boot/33703>

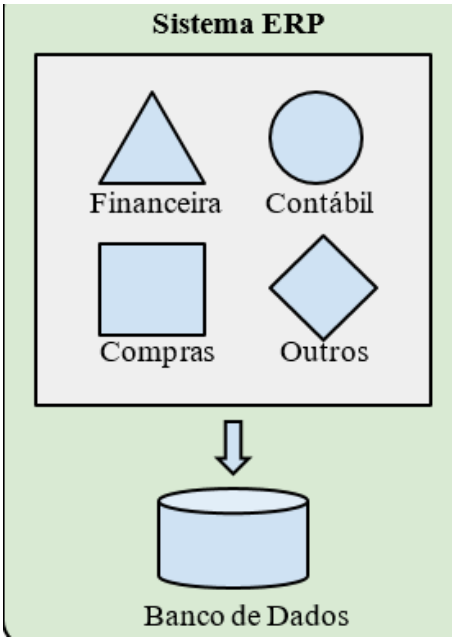

Figura 4: Exemplo: Arquitetura Monolitica **(** Acessado dia 10/07/2019**,** https://www.devmedia.com.br/introducao-a-arquitetura-de-microservices-com-spring-boot/33703.

Introduzida a mais tempo que o padrão de microservices, a arquitetura monolítica tambem é comumente utilizada no desenvolvimento de aplicaçoes web. No entenato, assim ele prove de vantagens e desvantave

3.2 MICROSERVICES

Em suma, o estilo arquitetural de microservices consiste no desenvolvimento de uma aplicação como um conjunto de pequenos serviços, a cada um executando em seu próprio processo. Os serviços podem ser construídos de forma modular com base na capacidade de negócio da organização em questão.

Pelo fato de os serviços serem construídos de forma independente, a implantação e a escalabilidade podem ser tratados de acordo com a

demanda. Essa independência também permite que os serviços sejam escritos em diferentes linguagens de programação e diferentes tecnologias, visando atender a necessidade especificas da melhor maneira. Outro ponto importante e que cada serviço pode ser gerenciado por diferentes times de desenvolvimento, possibilitando a formação de equipes especializadas.[8]

Microservices possuem uma grande vantagem em relação aos padrões (monolíticos e SOA), porque permite que os serviços sejam implementados de forma heterogenia, ou seja, cada serviço pode ser implementado com tecnologia distinta (C, Python, Rust), porem traz consigo a desvantagem, que é complexidade empregada. Por serem independentes entre si, o debug da aplicação fica difícil de ser realizado, e exige conhecimento das tecnologias usadas.[9]

3.3 SPRING BOOT

Desde 2003, o ecossistema Spring cresceu muito, o que do ponto de vista do desenvolvedor é bom, pois aumenta a gama de opções para usar, e ele mesmo não precisa implementar. Por exemplo: para adicionar uma autenticação na aplicação, podemos usar o Spring Security; para autenticar no Facebook ou Google, podemos usar o Spring Social. Já se existir uma necessidade de criar muitos processos com horário agendado, temos o Spring Batch. E essa lista e enorme.

Entretanto, esse crescimento do Spring trouxe alguns problemas: com muitos módulos, vieram muitas dependências, e a configuração já não e tão simples assim como antes.

[8] Disponível em < https://www.devmedia.com.br/introducao-a-arquitetura-de-microservices-com-spring-boot/33703>

[9] Disponivel em <https://www.embarcados.com.br/microservices/>

O Spring Boot além de impulsionar o desenvolvimento para microsserviços, também ajuda na configuração importando e configurando todas as dependências. Às vezes, ele e confundido com um simples framework, mas na verdade ele e um novo conceito de criar aplicações Web.

No conceito de Java Web Container, temos o framework Spring controlando as suas regras de negócio empacotadas em um JAR, que deverá obedecer aos padrões (servlet, filter, diretório WEB-INF etc.). BOAGLIO,2021, p.09,10.)

3.4 SUA ARQUITETURA

A arquitetura do Spring Boot é formada pelos componentes:

- **CLI** – O Spring Boot CLI é uma ferramenta de linha de comando que facilita a criação de protótipos através de scripts em Groovy.
- **Starters** – Conjunto de componentes de dependências, que podem ser adicionadas aos nossos sistemas.
- **Autoconfigure** - Configura automaticamente os componentes carregados.
- **Actuador** - Ajuda a monitorar e gerenciar as aplicações publicadas em produção.
- **Tools** – Uma IDE customizada para desenvolvimento com Spring Boot.
- **Samples** – Dezenas de exemplos de implementação disponíveis para uso. BOAGLIO,2021, p.11,12.)

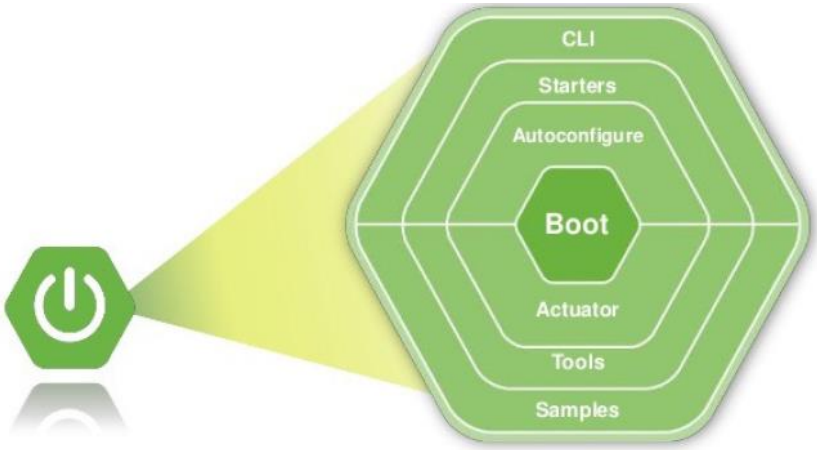

Figura 5: Spring Boot Acelere o Desenvolvimento de Micros serviços (In: BOAGLIO et al.; 2021 p 1

Abaixo representação gráfica para estudo de caso implementação de micro serviços.

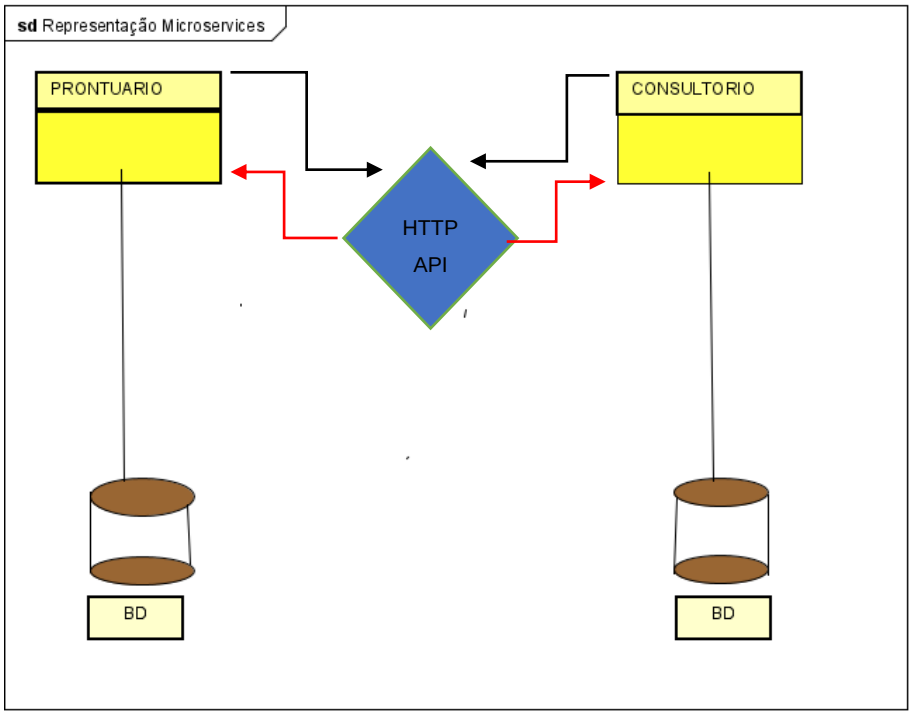

Figura 6: Proposta do Trabalho e implementação Conceito Microsservice.

A proposta e o estudo deste trabalho com base na literatura apresentada e estudos em andamento propõe explorar o uso framework Spring Boot para

desenvolver aplicações orientadas a microsserviços, e servir de base para futuras consultas relacionadas ao uso das tecnologias abordadas.

Acima representado pela figura seis, temos o seguinte problema para solucionar, está sendo solicitado a implementação de um Prontuário On Line. No conceito de sistemas Monolíticos poderia ser construído apenas uma base de dados, as tabelas respectivas ao banco, as implementações SQL de rotina o desenvolvimento da aplicação na linguagem escolhida e resolver o caso.

Na solução proposta do uso de microsserviços, a aplicação será dividida em aplicações distintas na metodologia do conceito Rest, com comunicação HTTP.

Em nosso estudo teremos PRONTUARIO x CONSULTORIO acessando seus bancos distintos enviando requisições CLIENTE x SERVIDOR através do protocolo HTTP.

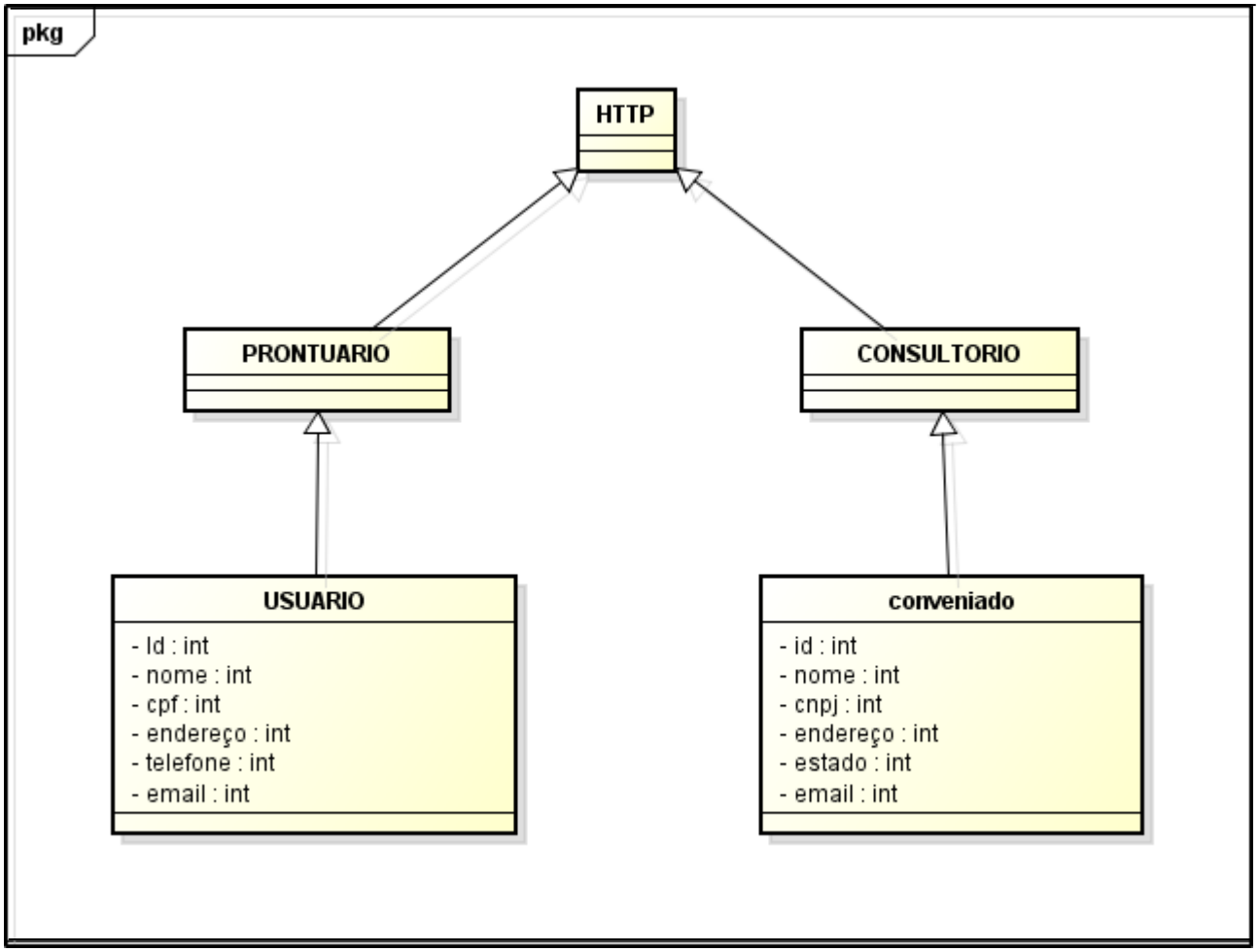

Figura 7: Representação Diagrama de Classes Aplicação.

4. ESTUDO DE CASO DE USO.

4.1 STARTANDO O SPRING BOOT

Neste capitulo, será abordado o estudo do uso do framework Spring Boot, que associado em conjunto com ferramentas específicas, e integradas ao projeto auxilia, a desenvolver e ter uma produção de código mais automatizada, organizada e de ampla integração com as tecnologias envolvidas na demanda.

O Spring Boot deverá ser usados em conjunto com um ambiente Java, independentemente de qualquer que seja a ferramanta escolhida para o desenvolvimento.

Neste projeto será usado IntelleJ IDEA, o Spring Boot poderá ser inicializado e configurado dentro do ambiente da IDE, ou gerado no site https://start.spring.io e importado para o projeto, será disponibilizado um arquivo compactado.

Na figura abaixo está sendo apresentada a tela de starter do Spring Boot

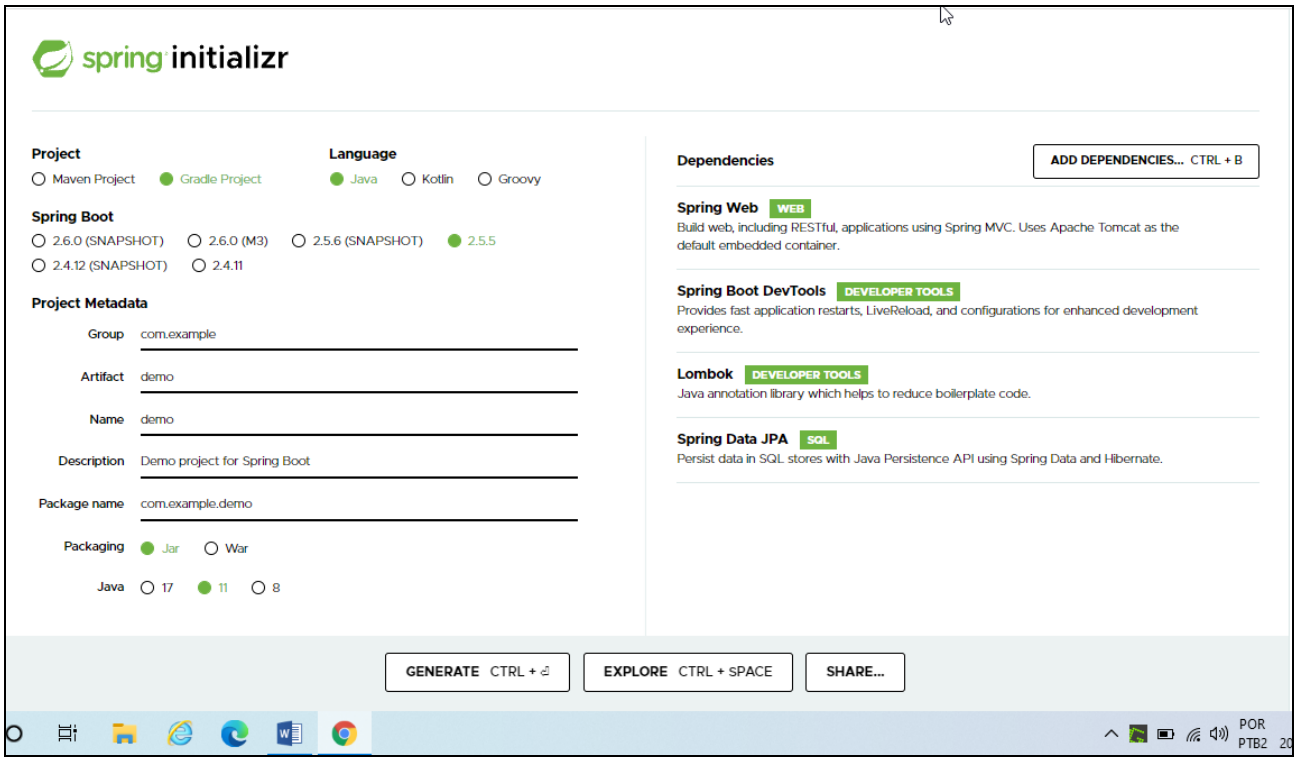

Figura 8: Tela de inicialização e configuração Spring Boot.

Nesta etapa, é necessário fazer as configurações iniciais do Projeto onde destaca-se as seguintes etapas:

Project MAVEN ou GRADLE PROJECT, são responsáveis pelo gerenciamento das dependências e artefatos do projeto garantindo sua funcionalidade e integração com os plug-ins necessários para start e build da aplicação.

Language, neste caso e JAVA, Spring Boot é a versão do (SNAPSHOT) já vem por padrão a última, Project Metadata nome projeto e pacotes da

estrutura e organização do mesmo, Pakaging pacote Jar ou War e Java escolha versão.

Dependências, esta etapa e muito importante pois e onde faremos as escolhas das dependências necessárias para desenvolvimento do nosso projeto, e no momento que escolhemos e orquestramos, o que será necessário para desenvolver o projeto conforme o estudo de caso.

Antes de prosseguir com as etapas e continuidade do projeto, será apresentado e destacado referências a documentação do Spring Boot https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle o documento está disponível neste endereço para consultas relacionadas a sua manipulação e emprego de suas ferramentas.

Figura 9: Tela de inicialização a documentação Spring Boot.

É de ampla importância que o desenvolvedor conheça a ferramenta que irá utilizar, um dos objetivos deste documento é ajudar a entender o Ecossistema da aplicação para que se possa executar com êxito as integrações no projeto.

A fim de estudos apenas citamos a existência da documentação para posterior consultas em meios mais detalhados, não é nosso foco no presente momento aprofundar em cada tópicos existentes por se tratar assuntos longos que exige um tempo maior hábil para estudo e aprendizado.

Antes de se iniciar o desenvolvimento com Spring Boot, é necessário realizar um análise criterioso da escalabilidade da aplicação em relação ao projeto e desenvolvimento para adequar juntamente com a equipe de Engenharia de Software a integração correta do ecossistema de desenvolvimento.

4.2 CONFIGURAÇÃO DO AMBIENTE SPRING BOOT.

A ferramenta IDE para desenvolvimento deste projeto escolhida, foi o IntelliJ desta forma ao término de sua geração o arquivo e importado será demostrado na próxima figura.

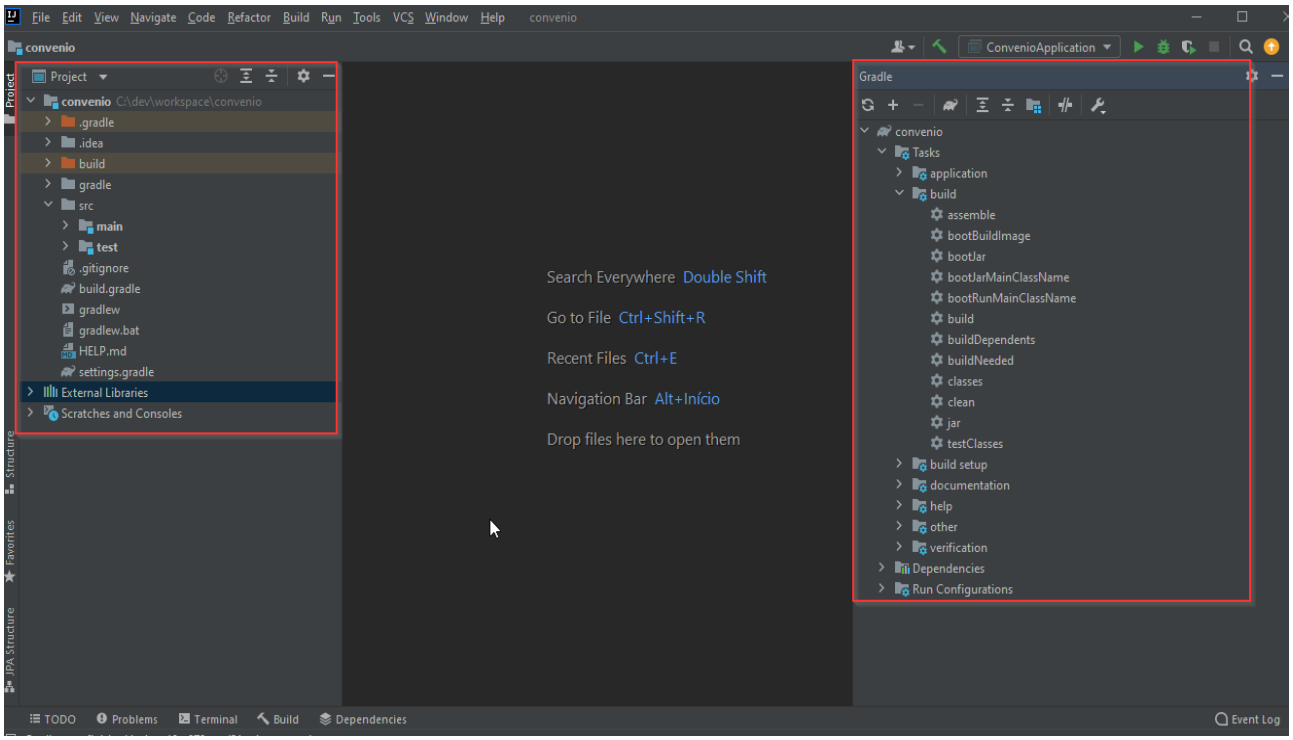

Figura 10: Tela inicial da importação projeto para ferramenta desenvolvimento.

Ao lado esquerdo em destaque temos as pastas do projeto criada pelo Spring Boot onde encontra-se a pasta principal do projeto SRC, com as sub pastas necessárias para build do projeto.

Ao lado direto está destacado o gerenciador de dependências GRADLE para este, foi optado a não usar o MAVEN para explorar o uso nesta aplicação.

Gradle e responsável por gerenciar as dependências do projeto e um de sistema de automatização de compilação de código aberto que se baseia em conceitos de Apache Ant e Apache Maven e introduz uma linguagem de domínio especifico (DSL) baseada em GROOVY em vez do XML, usado pelo Apache Maven para declarar configurações de projeto. Gradle foi projetado para multi – projetos que possam ter alto crescimento, e suporta compilações incrementais quando inteligentemente determina quais partes da árvore estão atualizadas, de modo que qualquer tarefa dependente dessas partes não precisa ser executada.[10]

Após a importação do projeto a ferramenta de desenvolvimento faz a checagem no projeto, baixa e atualiza todas as dependências necessárias, para configurações otimizando e as incorporando na aplicação. Não existe obrigatoriedade em usar nenhuma em especifico nos dias atuais nenhuma tecnologia pode estar presa uma a outra porem capaz de se integrar e crescer em grande escalabilidade para acompanhar a velocidade da necessidade cada vez de aplicações hibridas e capaz de atender a grande demanda de dados para extração de informações em grande escala.

Dentro da pasta rasoures contém um arquivo/ aplication.yml onde encontra-se a configuração de acesso ao banco de dados para este estudo de caso foi

[10] Disponível em < https://pt.wikipedia.org/wiki/Gradle>

usado o banco de dados POSTGRES em um container encapsulado dentro do ficando ele responsável pelo gerenciamento e execução da aplicação DOCKER não foi instalado no sistema operacional da máquina usada para o desenvolvimento assunto que trataremos em tópico especifico.

Abaixo, segue a figura com representação da localização do arquivo mencionado.

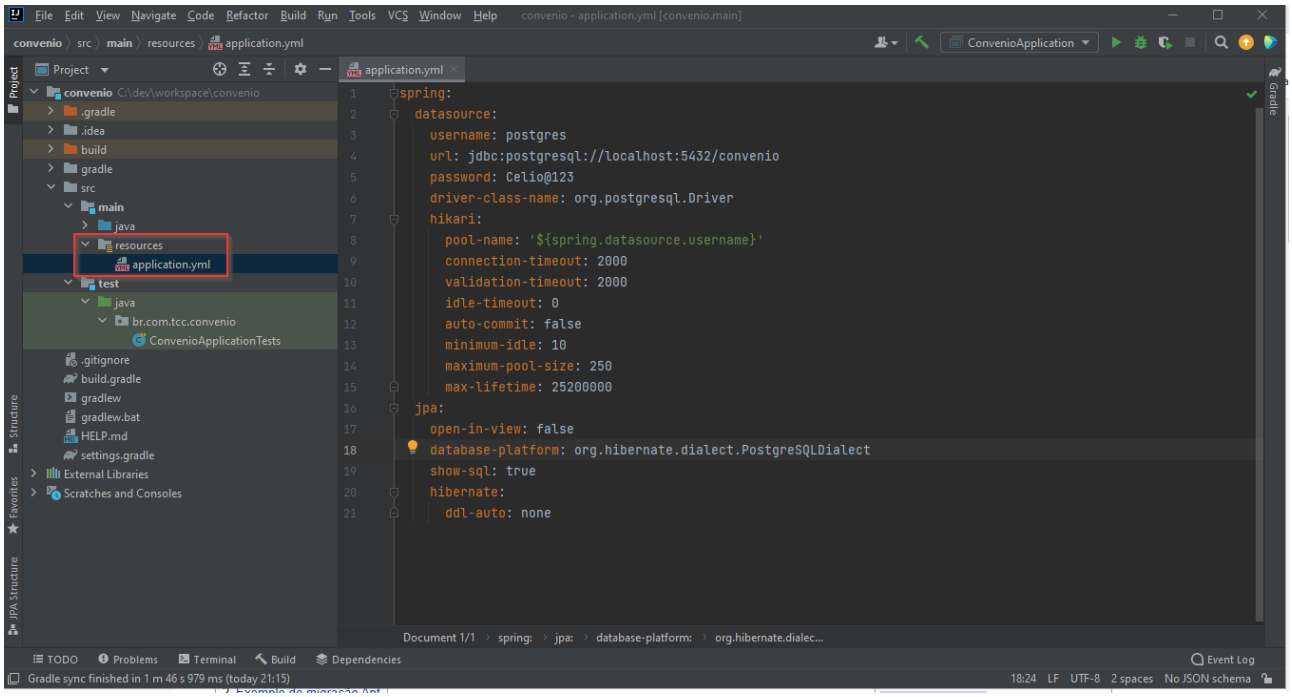

Figura 11: Representação da localização do arquivo de configuração acesso ao banco dados Spring Boot.

Para este estudo de caso foi adotado o uso DOCKER para expandir um pouco mais de tecnologias abrangente neste projeto para diversificar as possibilidades no desenvolvimento e despertar o conhecimento para o que existe hoje no mercado para e uma das ferramenta que está sendo muito requisitada para desenvolvimento de aplicações. É uma plataforma que compõe um conjunto de produtos como serviços que usam a virtualização nível de sistema operacional para entregar software em pacotes chamados contêiner.

Na figura abaixo segue o banco POSTEGRES com o serviço ativo é mapeado para uso no projeto em desenvolvimento de microsserviços.

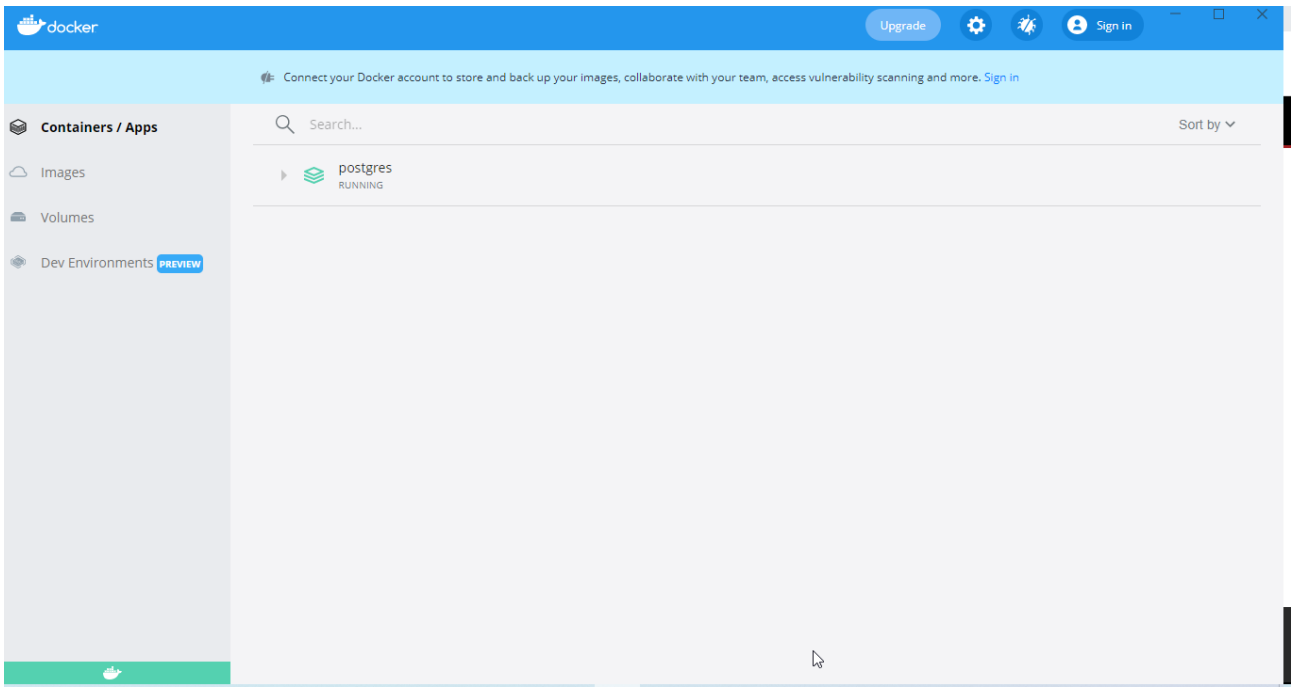

Figura 12: Serviço ativo no Docker banco dados Postegres.

O gerenciamento do banco e realizado pela ferramenta Pg Admin, é um software gráfico utilizado para administração do SGBD Postegre SQL disponível para Windows e Linux.

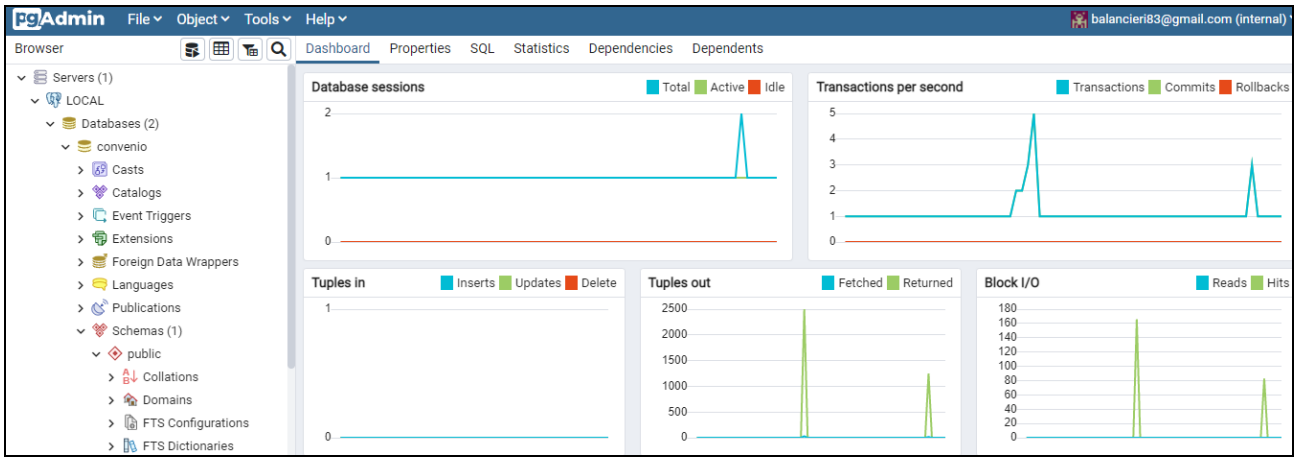

Figura 13: Serviço ativo no Docker banco dados Postegres.

Para auxiliar este trabalho, terá o uso da ferramenta Swagger para documentar os passos desenvolvimento dos microsserviços e testes para tanto e necessário a configuração do mesmo no ambiente e suas respectivas dependências.

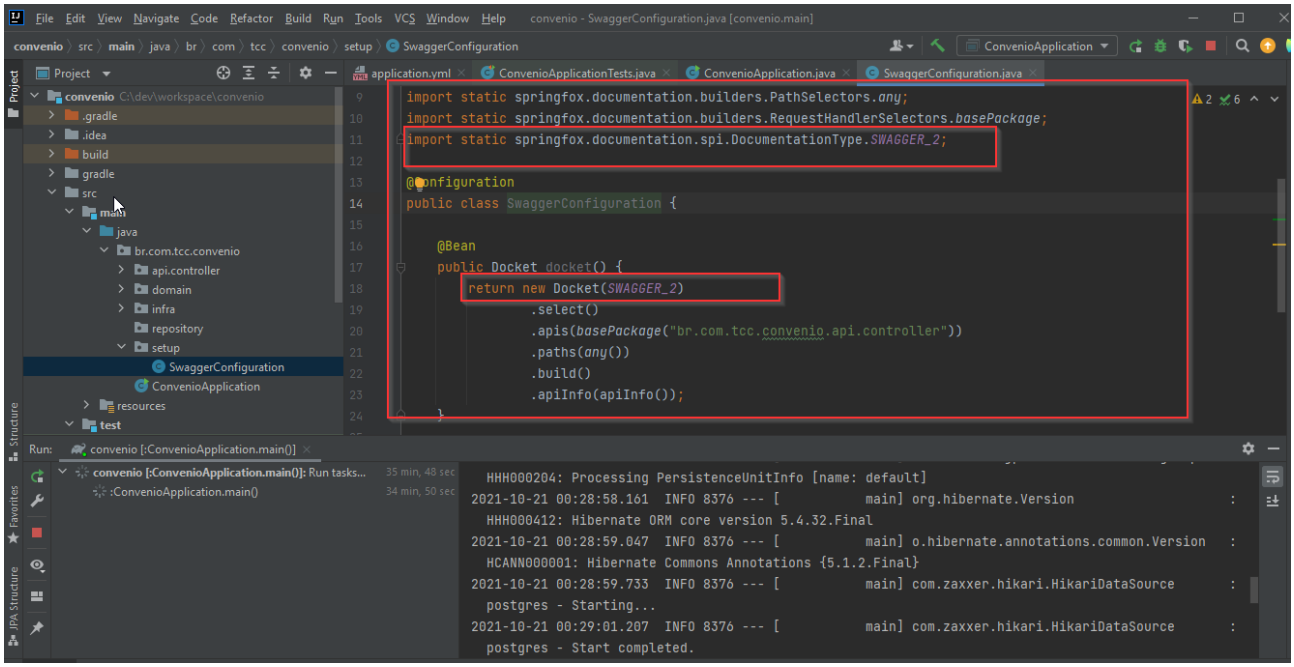

Figura 14: Configuração Swagger juntamente com o Spring Boot.

O swagger será responsável por gerenciar a documentação da API dentro do projeto onde sua funcionalidade e detalhar o que cada microsserviço fará dentro da aplicação de maneira clara e ágil auxiliando outro desenvolvedor realizar alterações mais segura, no momento do desenvolvimento é necessário cautela para colocar legendas corretas em cada método, um diferencial nesta ferramenta que podemos realizar testes em sua própria tela enviar e receber requisições.

Na figura abaixo a configuração estando correta será exibia a tela abaixo.

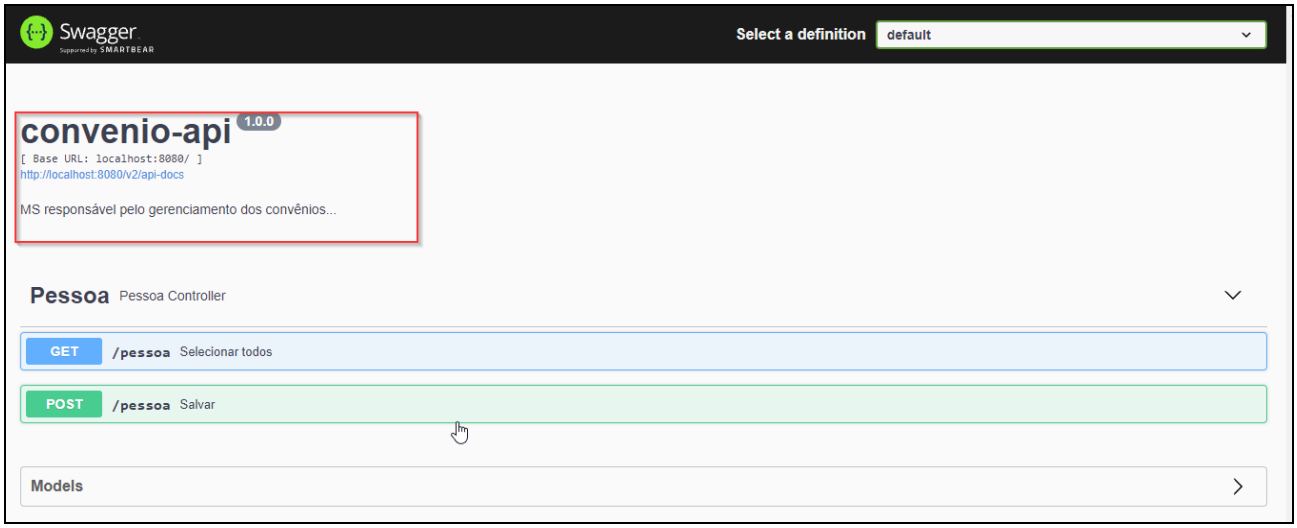

Figura 15: Swagger em execução após a configuração.

A ferramenta além de gerenciar a documentação da API, possibilita o desenvolvedor testar se ocorre comunicação entre as partes da aplicação. Enviado e recebendo requisições com suas respostas de acordo com cada método criado.

4.3 CRIANDO MICROSSERVIÇO COM SPRING BOOT

Para criar um serviço REST, precisamos apenas anotar uma classe como um serviço. Isso será feito utilizando a anotação @RestController. Mas e a URL do serviço? Para isso, usaremos a anotação @Mapping do Spring, uma para cada verbo HTTP desejado. Outro detalhe muito importante que se deve levar em consideração é a maturidade do REST que exige um pouco mais de conhecimento desenvolver aplicações baseadas em microsserviços.

Na figura abaixo tela configuração da aplicação com verbos HTPP e suas respectivas anotações.

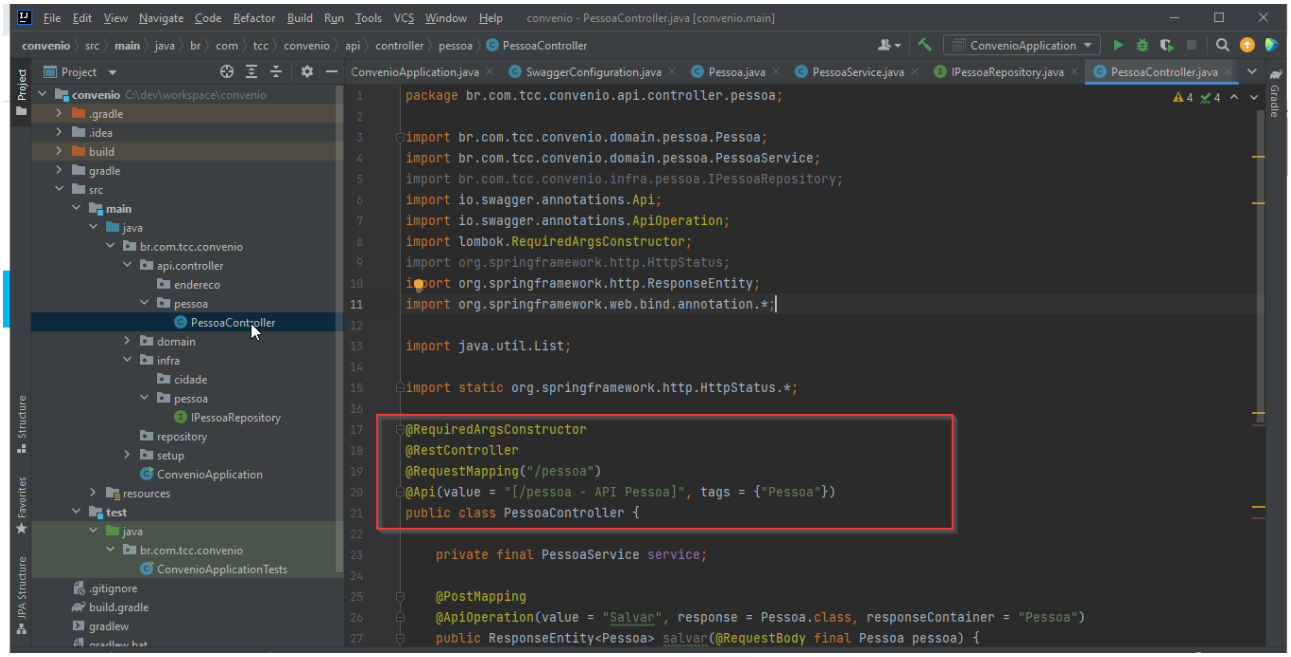

Figura 16: Figura de configuração Spring Boot para conclusão de serviços.

Para fins de estudos os pontos abordados foram os principais e cruciais para starter do projeto, não significa que as funcionalidades do Spring Boot estão todas em uso neste trabalho pois e um tema grande foi destacado os que compõem nosso estudo de Caso.

5. CONCLUSÃO FINAL.

Este trabalho abordou o estudo relacionado ao uso do Spring Boot destacando como e onde usar as funcionalidades, existe muitos profissionais que não conhece a ferramenta e vale ressaltar que para tal situação exige um estudo mais detalhado de sua documentação um maior domino e execução de projetos, para incrementar ainda mais o nosso caso de uso acrescentou micro serviços para uma performance ainda mais indagativa ao despertar a curiosidade e aguçar cada vez mais o desejo de busca de conhecimento.

Nada e fácil aprendizado e conhecimento são virtudes que se conquista com esforço e dedicação, perfeição e uma colheita de frutos plantados na jornada traçada por cada um individualmente.

6. CRONOGRAMA

CRONOGRAMA	ABR	MAI	JUN	JUL	AGO	SET	OUT	NOV	DEZ
Pre Projeto	■								
Mapa Mental	■								
Cronograma	■								
Levantamento Requisitos		■							
Analise de Requisitos		■							
Validação Requisitos		■							
Diagrama de Caso Uso			■						
Dagrama Classes			■						
Diagrama de Atividades			■						
Diagrama de Sequencia			■						
Diagrama de E-R			■						
Apresentação da Qualificação				■					
Implementação Banco Dados					■	■	■		
Implementação do Sistema					■	■	■		
Apresentação Banca Final								■	
Correçoes Finais									■

Figura 17: Cronograma

7. REFERÊNCIAS

BOAGLIO,Fernando. **SPRING BOOT Acelere o Desenvolvimento de microsserviços**. 2 Ed. Fernando, 2021.

DEITEL, Paul . **JAVA Como Programar.** 8. Ed. Pearson, 2010.

DIAS, Wesley. **Documentando sua API com Swegger** Disponível em <http://www2.decom.ufop.br/terralab/documentando-sua-api-rest-com-swagger/> Acessado 19/09/2021.

WILKIPÉDIA, a enciclopédia. **Framework.** Disponível em < https://pt.wikipedia.org/wiki/Framework>. Acessado em 16/08/2021.

WILKIPÉDIA, a enciclopédia. **PostgreSQL.** Disponível em <https://pt.wikipedia.org/wiki/PostgreSQL. Acessado em 16/08/2021.

WILKIPÉDIA, a enciclopédia **Intellij.** Disponível em < https://en.wikipedia.org/wiki/IntelliJ_IDE> Acessado 16/08/2021.

WILKIPÉDIA, a enciclopédia. **Docker.** Disponível em < https://pt.wikipedia.org/wiki/Docker_(software)#:~:text=Docker%20%C3%A9%20um%20conjunto%20de,bibliotecas%20e%20arquivos%20de%20> Acessado 19/09/2021.

DEVMEDIA, **Introdução à arquitetura de microservices com Spring Boot.** Disponível em <https://www.devmedia.com.br/introducao-a-arquitetura-de-microservices-com-spring-boot/33703>. Acessado em 28/09/2021.

SILVA, Cristiano, **Microservices.** Disponível em <https://www.embarcados.com.br/microservices/>. Acessado em 06/10/2021.

WILKIPÉDIA, a enciclopédia. **Gradle.** Disponível em < https://pt.wikipedia.org/wiki/Gradle>. Acessado em 07/10/2021.

SOFWARE, Opus. **Microserviços qual a diferença para arquiterura monolitica.** Disponivel em <https://www.opus-software.cohttps://www.opus-software.com.br/micro-servicos-arquietura-monolitica/m.br/micro-servicos-arquietura-monolitica/>. Acessado em 15/08/2021.