DBC
DIGITAL BUSINESS COMPANY®

VEM SER
DBC

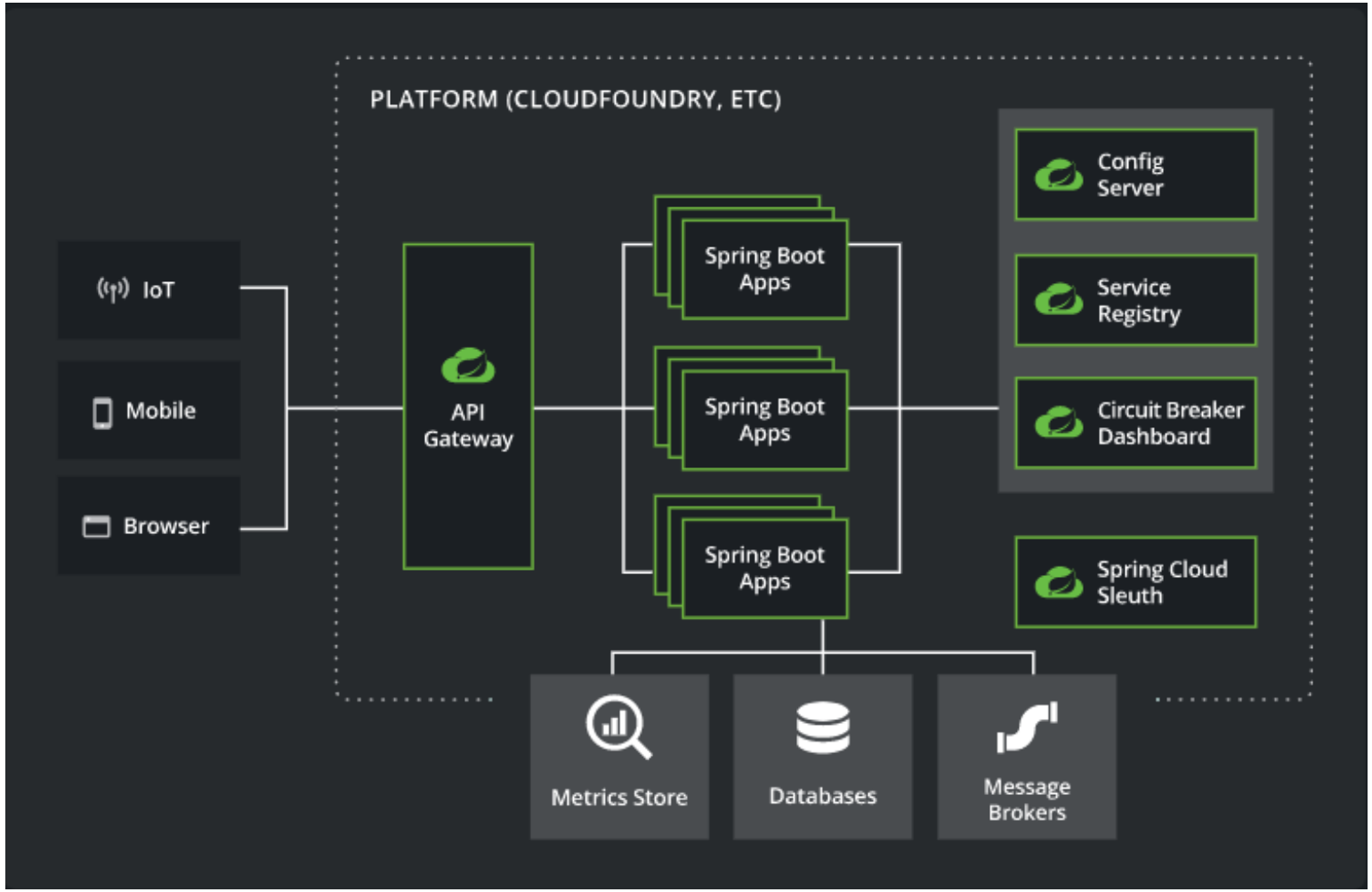
Spring Web

Aula 2 – Desenvolvimento Web com Spring

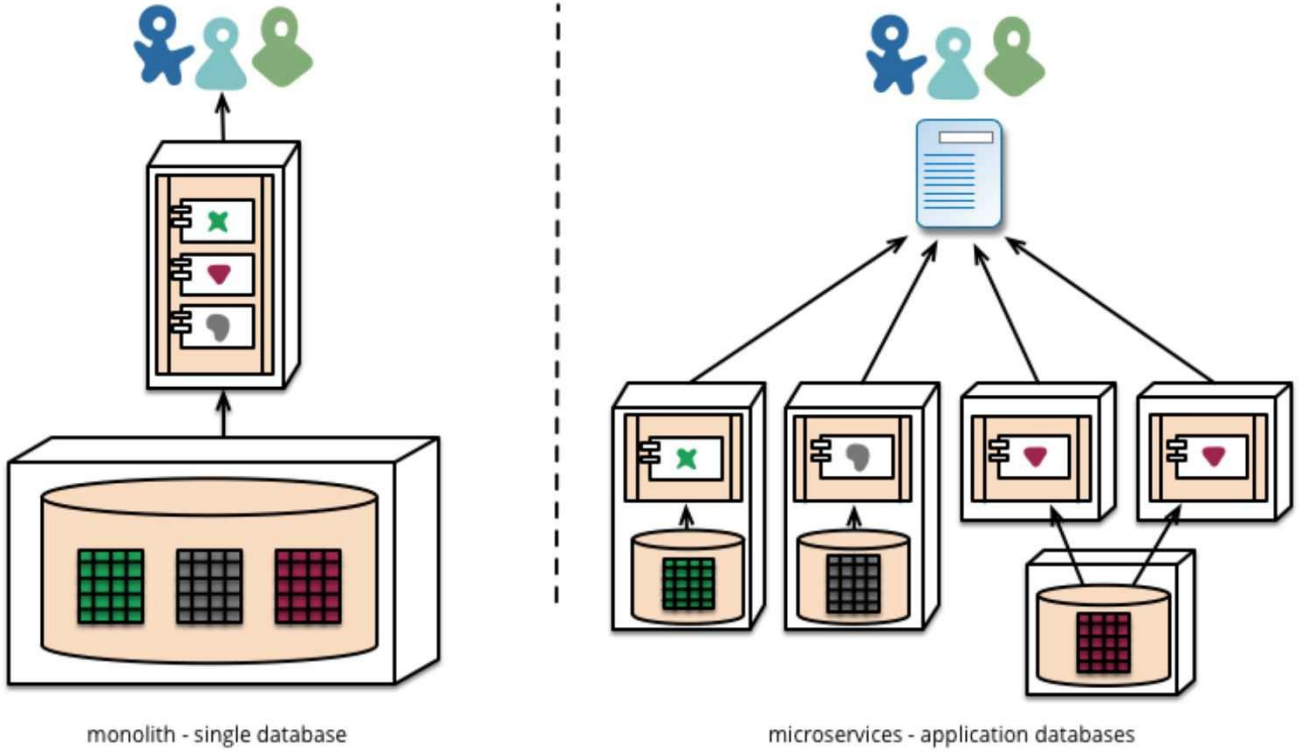
Sumário

- Microservices
- Monolith x Microservices
- Spring e Spring Boot
- Spring Initializr
- Nosso primeiro projeto no SpringBoot

Microservices



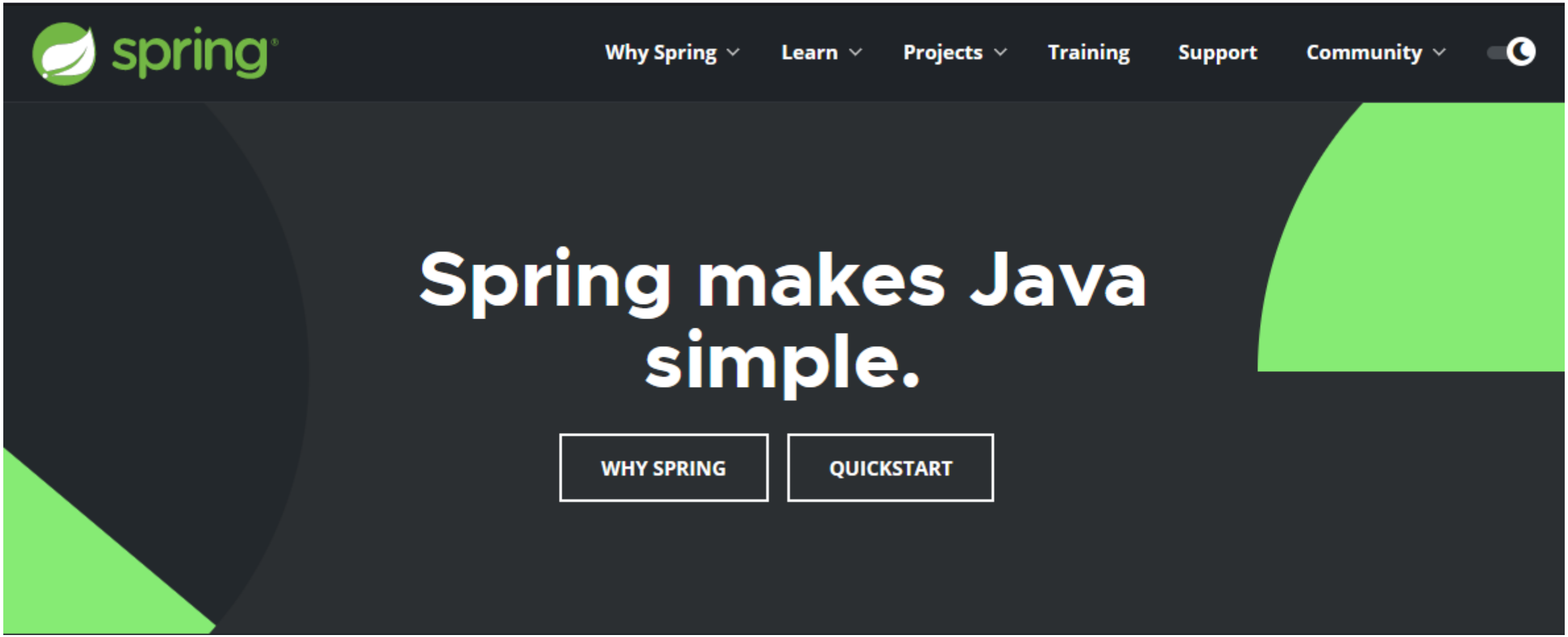
Microservices



monolith - single database

microservices - application databases

Spring


spring
Why Spring
Learn
Projects
Training
Support
Community
Spring makes Java simple.
WHY SPRING
QUICKSTART

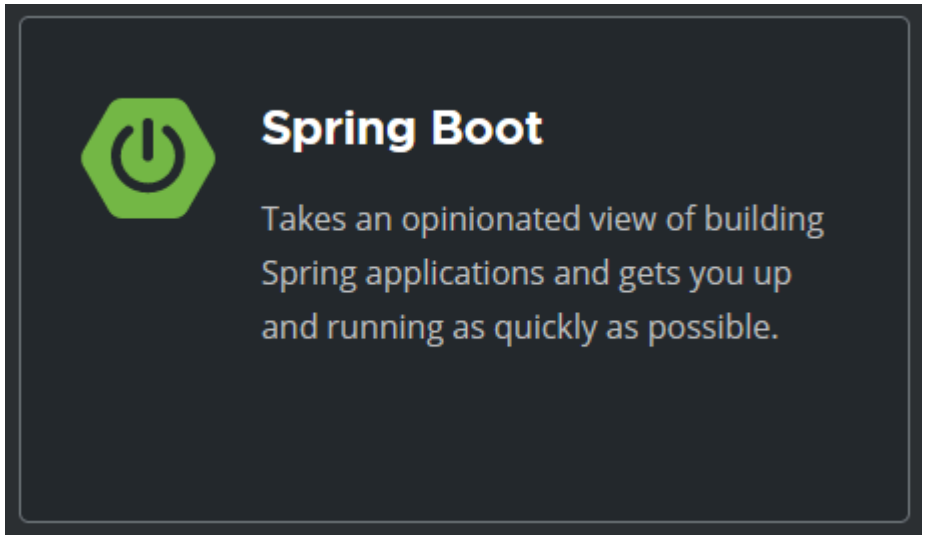
Spring

- Spring é um conjunto de frameworks Java para aplicações em qualquer tipo de plataforma

- Se concentra para que as equipes possam se concentrar na lógica de negócios no nível do aplicativo, sem vínculos desnecessários com ambientes de implementação específicos.

Spring Boot

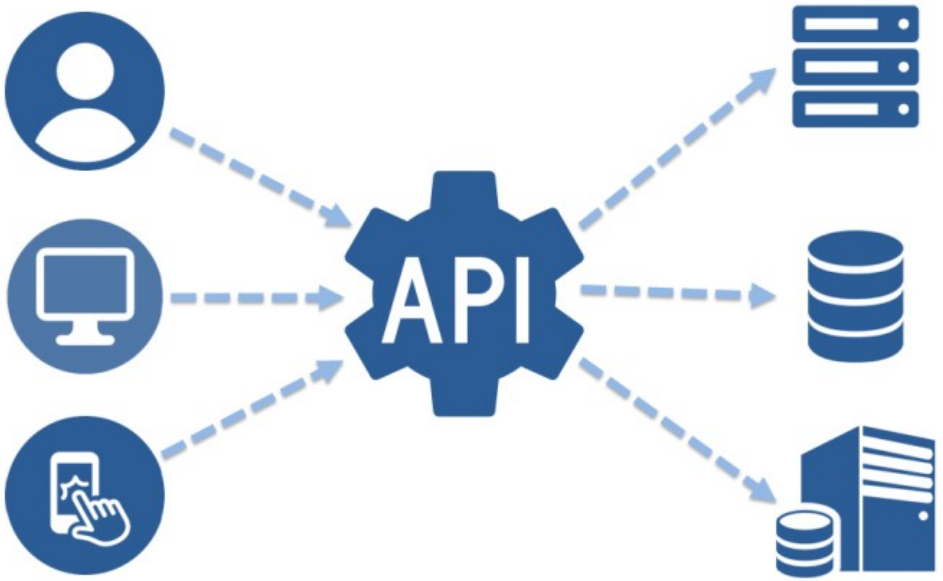
- **Spring Boot** facilita a criação de aplicações Java que utilizam o ecossistema Spring com pouca ou nenhuma configuração para executar o projeto. **Ele abstrai toda a complexidade que uma configuração completa pode trazer.**



API RESTful

- É uma API (interface application program) que utiliza requisições HTTP para extrair, inserir, postar e deletar dados.

- A API RESTful tem como base a tecnologia REST (representational state transfer), um tipo de arquitetura e comunicação muito utilizado no desenvolvimento de serviços web.



https://start.spring.io/

Vamos praticar!

Exercício #1

- Criar um projeto com o **Spring Initializr** chamado “pessoa-api”
- Abrir o projeto e executar com o Intellij

Principais anotações Spring

```
/*
 Essa anotação ativa a configuração baseado em Java,
  bem como o recurso de varredura e
  configuração automática de componentes do Spring Boot.
 */
@SpringBootApplication
```

```
@RestController // define uma classe que contém métodos para uma API RESTful.
@RequestMapping("/contato") // mapeia requisições REST.
public class ContatoController {
```

```
@DeleteMapping("/{id}") // define um endpoint do tipo HTTP DELETE (pode conter variáveis)
public void delete(@PathVariable("id") Long id) { // recupera a variável passada no URI (path)
```

```
@PostMapping("/{idPessoa}") // define um endpoint do tipo HTTP POST (pode conter variáveis)
public Contato create(@PathVariable("idPessoa") Integer idPessoa,
// recupera a variável passada no URI (path)
                      @RequestBody Contato contato) throws Exception {
// corpo da requisição em formato JSON
```

Principais anotações Spring

```
@PutMapping("/{id}") // define um endpoint do tipo HTTP PUT (pode conter variáveis)
public Contato update(@PathVariable("id") Integer id,
// recupera a variável passada no URI (path)
                      @RequestBody Contato contato) {
// corpo da requisição em formato JSON
```

```
@PutMapping("/") // define um endpoint do tipo HTTP PUT (pode conter variáveis)
public Contato update(@RequestParam("id") Integer id,
// recupera a variável por queryParam (url?id=X)
                      @RequestBody Contato contato) {
// corpo da requisição em formato JSON
```

```
@GetMapping("/{idPessoa}") // define um endpoint do tipo HTTP GET (pode conter
variáveis)
public List<Contato> listByIdCliente(@PathVariable("idPessoa") Integer
idPessoa) { // recupera a variável passada no URI (path)
```

Homework

- Utilizando o projeto “pessoa-api” criar um novo controller, **ContatoController** com a rota **“/contato”** com as seguintes operações:

  - **GET:** deve listar todos os contatos
  - **GET:** por pessoa: deve receber o id da pessoa e listar os contatos da pessoa
  - **POST:** com um id da pessoa para adicionar o contato e no corpo, receber o contato para inserir
  - **PUT:** com id do contato, deve atualizar um contato existente e receber no corpo da requisição os novos dados do contato (deve receber todos os dados)
  - **DELETE:** deve receber um id do contato e remover da lista

- Criar uma collection do POSTMAN, testar todos os métodos e exportar a collection e colocar dentro do seu projeto.

Obrigado!

DBC

DIGITAL BUSINESS COMPANY®

/dbc.company /dbccompany /dbccompany.com.br /company/dbc-company