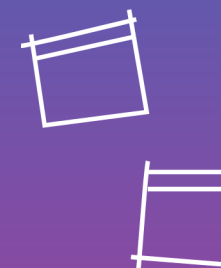




 **VEM SER**
DBC



Spring Web

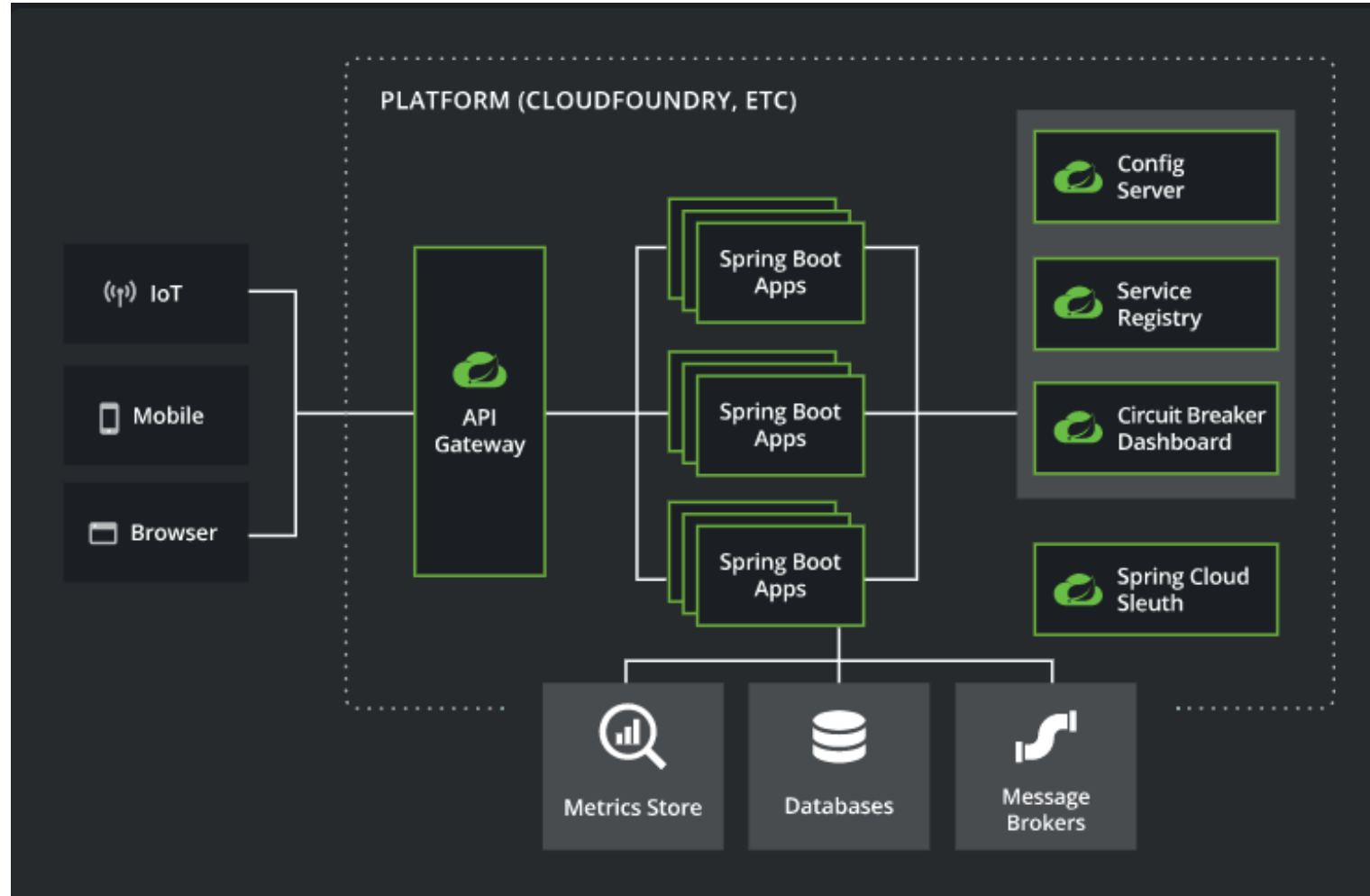
Aula 2 – Desenvolvimento Web com Spring



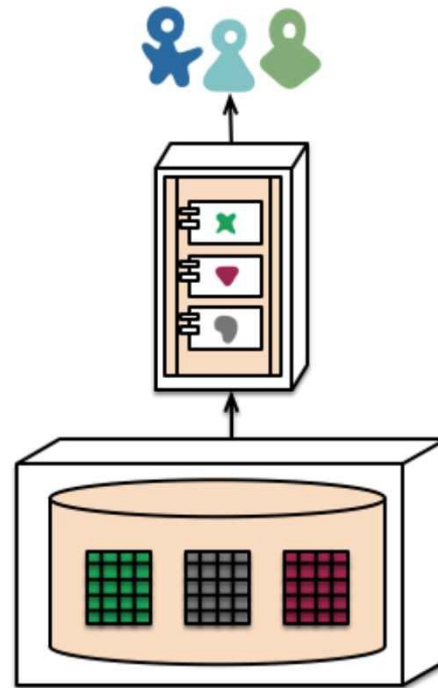
Sumário

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

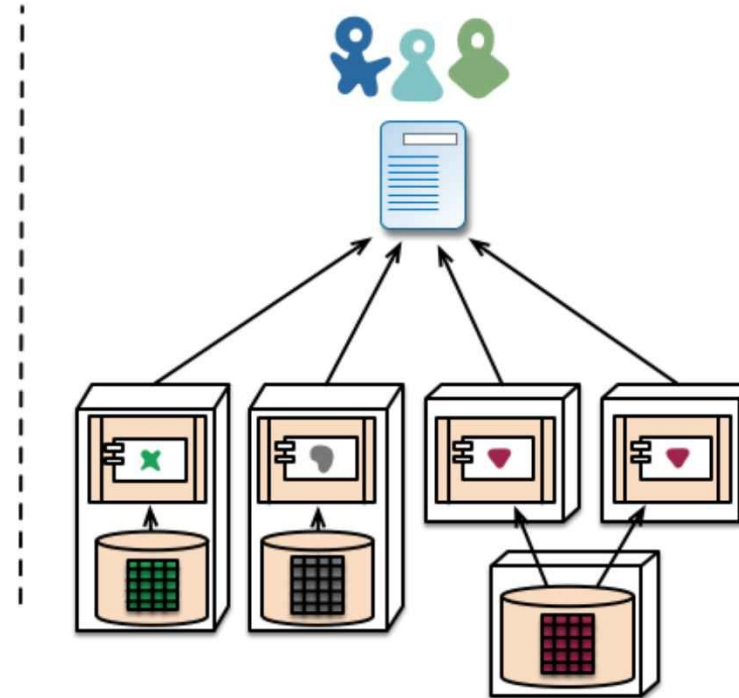
Microservices



Microservices

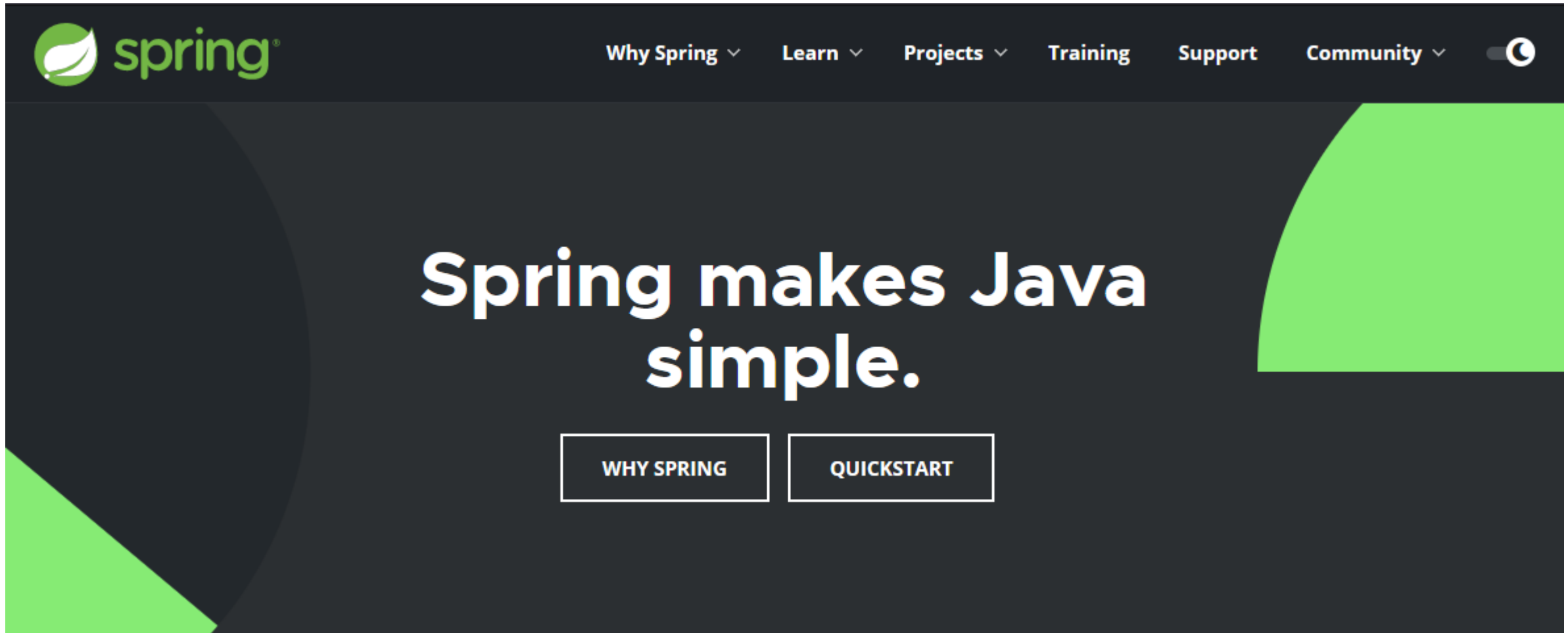


monolith - single database



microservices - application databases

Spring



Spring

- 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

Takes an opinionated view of building Spring applications and gets you up and running as quickly as possible.



Spring Framework

Provides core support for dependency injection, transaction management, web apps, data access, messaging, and more.



Spring Data

Provides a consistent approach to data access – relational, non-relational, map-reduce, and beyond.



Spring Cloud

Provides a set of tools for common patterns in distributed systems. Useful for building and deploying microservices.



Spring Cloud Data Flow

Provides an orchestration service for composable data microservice applications on modern runtimes.



Spring Security

Protects your application with comprehensive and extensible authentication and authorization support.

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.**

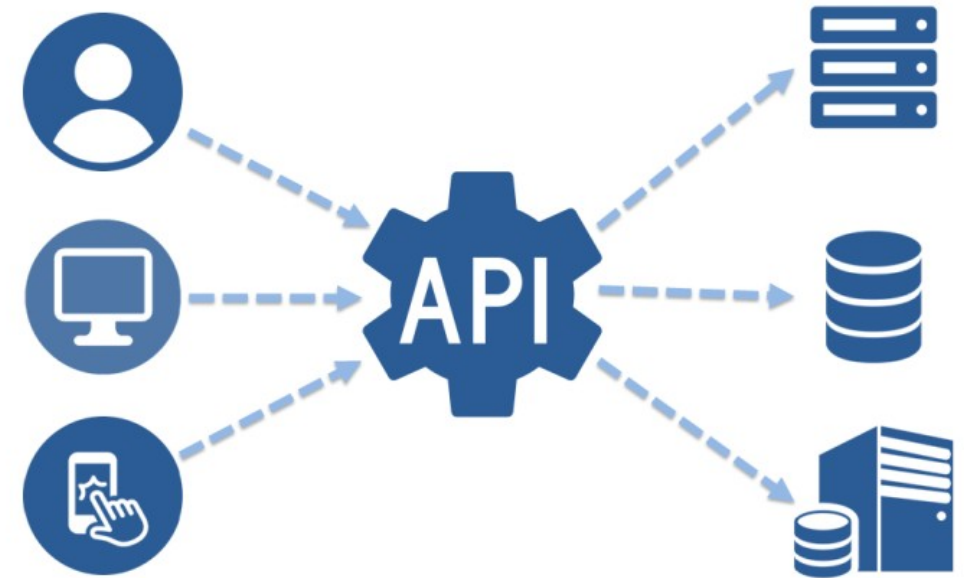


Spring Boot

Takes an opinionated view of building Spring applications and gets you up and running as quickly as possible.

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”;
- O pacote padrão deve ser “br.com.dbc.vemser.pessoaapi”;
- 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

```
@PostMapping("/{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
```

```
@PostMapping("/") // define um endpoint do tipo HTTP PUT (pode conter variáveis)
public Contato update(@RequestParam("id") Integer id,
// recupera a variável por queryParams (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)
```



Vamos praticar!



Task

- 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 “PessoaAPI” no POSTMAN para os métodos criados e testados e compartilhar esta no classroom, pública.

Obrigado!

DBC

DIGITAL BUSINESS COMPANY®



 /dbc.company

 /dbccompany

 /dbccompany.com.br

 /company/dbc-company