

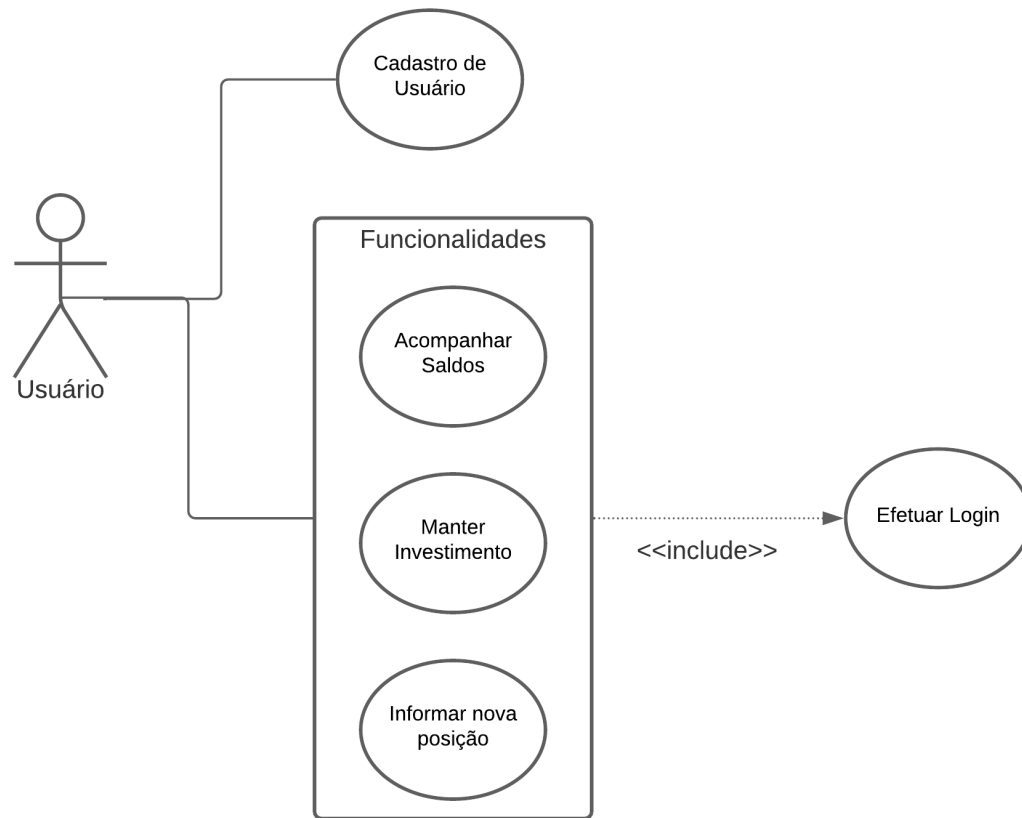
Camada Controller

Laboratório de Programação

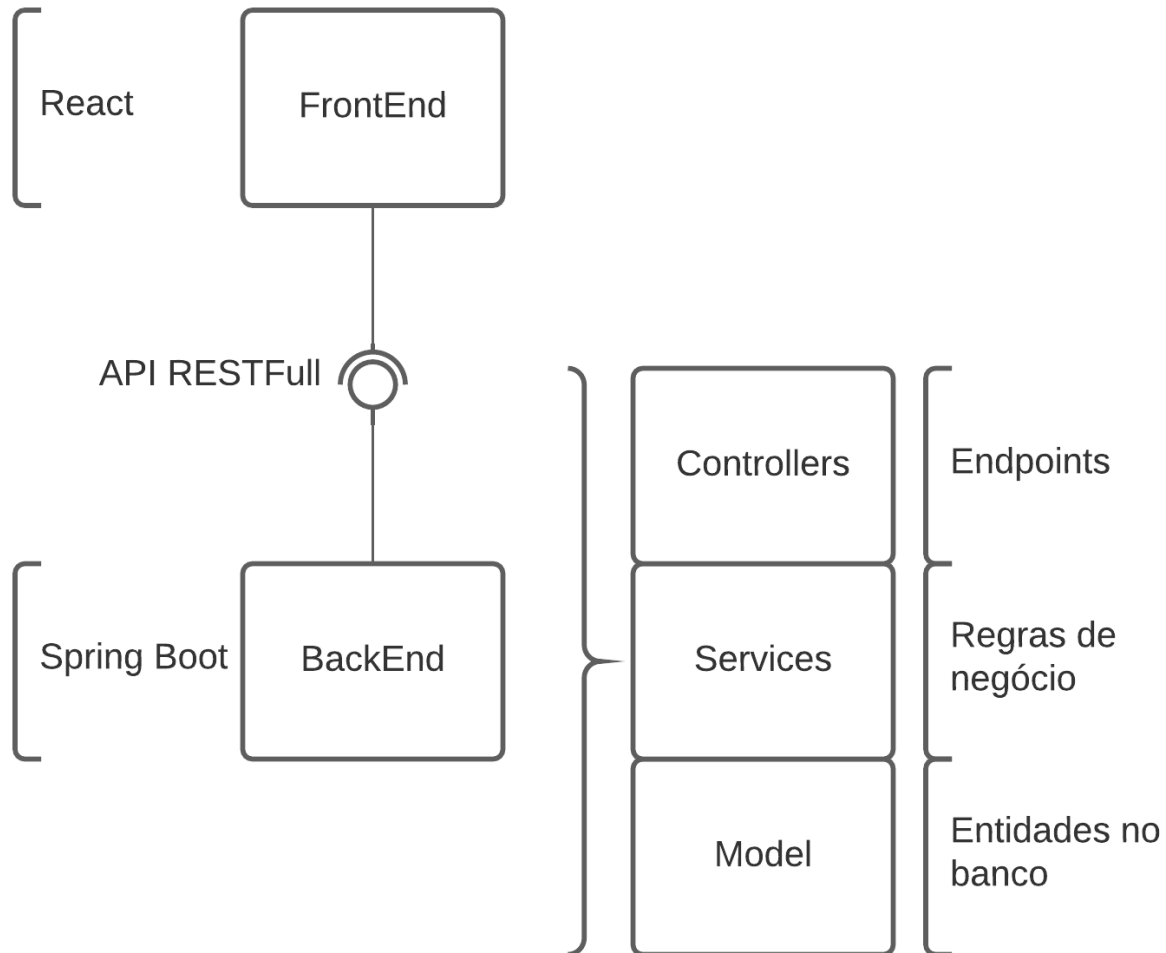
Objetivos

- demonstrar implementação dos endpoints REST
- aprender sobre modelos de resposta em HTTP

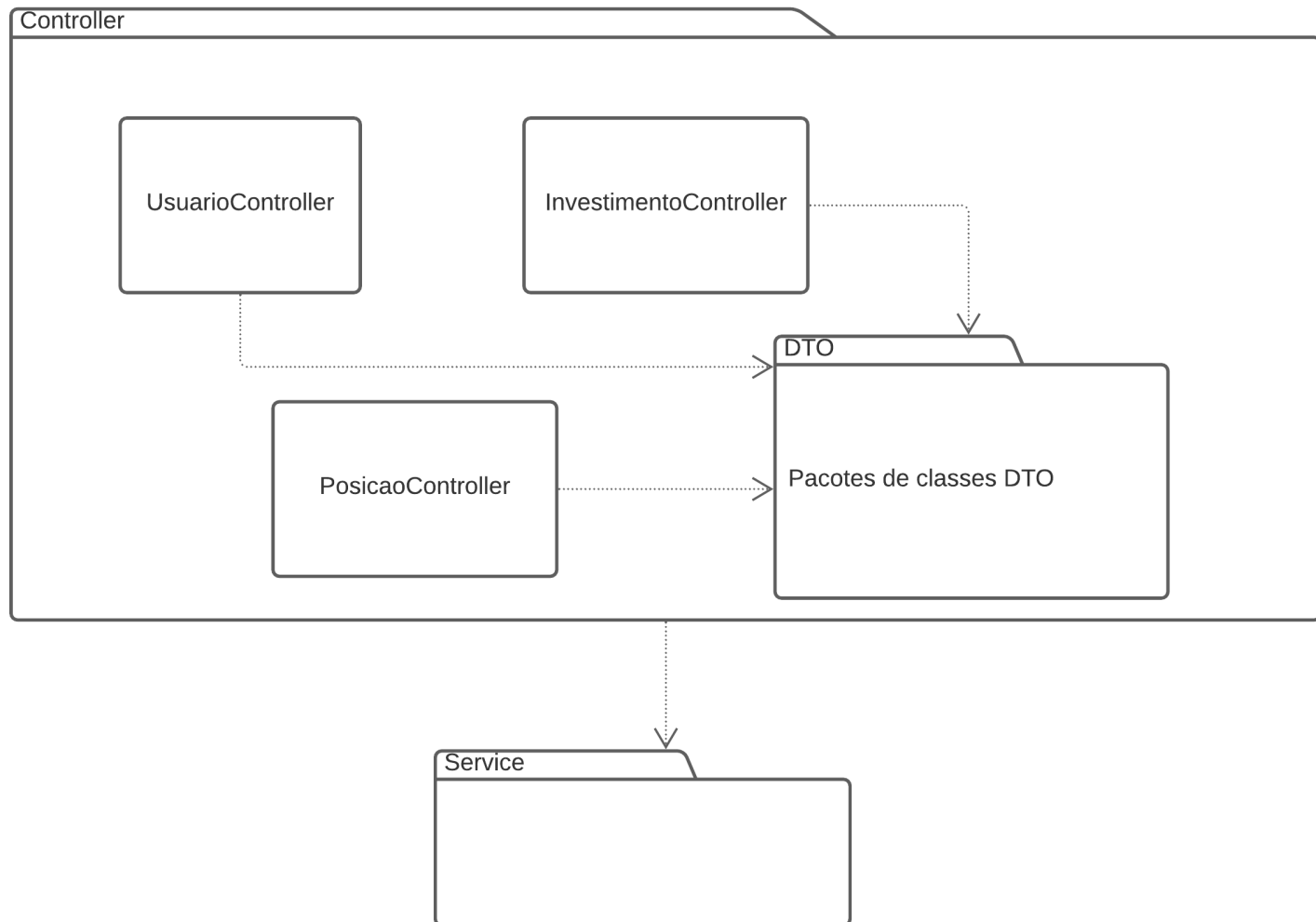
Visão de Casos de Uso



Visão Arquitetural



Visão de Classes



Implementando UserController

Endpoints de

UserController

- salvar
 - deve enviar dados do usuário a ser cadastrado
- autenticar
 - deve informar email e senha
- obterInvestimentos
 - deve informar usuario e receber investimentos com saldo

Configurando UserController

- Passo 1: criar o controller

```
package com.labprog.patrimonio.controller;  
  
public class UserController {  
  
}
```

- Passo 2: anotar como controller REST

```
@RestController  
public class UserController {  
  
}
```


Configurando

UserController

- Passo 3: definir o mapeamento da API

```
@RestController
@RequestMapping("/api/usuarios")
public class UserController {

}
```

- Para simplificar, vamos usar `api` e nome do controller `usuarios`

Configurando

UserController

- Passo 4: obter uma instância injetada de UsuarioService

```
@RestController
@RequestMapping("/api/usuarios")
public class UsuarioController {
    @Autowired
    UsuarioService service;
}
```

- Passo 5: definir endpoints
 - salvar precisa ser `post`
 - autenticar pode ser `post`
 - os dois entram em conflito, portanto terá que fazer um mapeamento separado para 1 deles
 - obter investimentos é `get`

Construindo UserController

- Construindo um endpoint `post` para salvar

```
@PostMapping
public ResponseEntity salvar(@RequestBody UsuarioDTO dto) {
}
```

- Adicionar anotação `@PostMapping`
- Necessário:
 - receber no corpo (`@RequestBody`) um json de usuário
 - aqui transformado em um DTO: `UsuarioDTO`
- Retornar:
 - `ResponseEntity`: resposta padrão no HTTP
 - Status + objetos
 - ou somente status

Ok: Anotações

- `@PostMapping`: representa que ali está sendo criado um endpoint `post`
 - se desejar, pode especificar um caminho adicional na URL
 - `@PostMapping("/novo")`
- `@ResponseBody`: converte o JSON no objeto tipo `UsuarioDTO`
 - a sequencia dos campos do JSON devem ser as mesmas que em `UsuarioDTO`

Ok: UsuarioDTO

- Objeto simples para representar o JSON da requisição

```
package com.labprog.patrimonio.model.dto;

@Data
@Builder
@AllArgsConstructor
public class UsuarioDTO {
    private String nome;
    private String email;
    private String senha;
}
```

Ok: ResponseEntity

- Resposta enviada pelo método
 - pode ser com template ResponseEntity para especificar o tipo de objeto de retorno
- Algumas opções:

```
new ResponseEntity(<objeto>, HttpStatus.CREATED);  
ResponseEntity.badRequest().body(e.getMessage());  
ResponseEntity.ok(<objeto>);  
...
```

- Os objetos são convertidos em JSON

Construindo UserController

- Construindo um endpoint `post` para salvar
 - converte o DTO em Entity

```
@PostMapping
public ResponseEntity salvar(@RequestBody UsuarioDTO dto) {
    Usuario usuario = Usuario.builder()
        .nome(dto.getNome())
        .email(dto.getEmail())
        .senha(dto.getSenha()).build();
}
```

Construindo UserController

- Construindo um endpoint `post` para salvar
 - converte o DTO em Entity
 - tenta salvar o objeto

```
@PostMapping
public ResponseEntity salvar(@RequestBody UsuarioDTO dto) {
    Usuario usuario = Usuario.builder()
        .nome(dto.getNome())
        .email(dto.getEmail())
        .senha(dto.getSenha()).build();

    try {
        Usuario salvo = service.salvar(usuario);
    }
    catch (RegraNegocioRunTime e) {
    }
}
```


Construindo UserController

- Construindo um endpoint `post` para salvar
 - converte o DTO em Entity
 - tenta salvar o objeto
 - retorna

```
@PostMapping
public ResponseEntity salvar(@RequestBody UsuarioDTO dto) {
    Usuario usuario = Usuario.builder()
        .nome(dto.getNome())
        .email(dto.getEmail())
        .senha(dto.getSenha()).build();

    try {
        Usuario salvo = service.salvar(usuario);
        return new ResponseEntity(salvo, HttpStatus.CREATED);
    }
    catch (RegraNegocioRunTime e) {
        return ResponseEntity.badRequest().body(e.getMessage())
    }
}
```

Construindo UsuarioController

- Autenticar
 - também é um post, precisamos mudar o endpoint
 - obter do DTO os campos
 - retornar

```
@PostMapping("/autenticar")
public ResponseEntity autenticar(@RequestBody UsuarioDTO dto) {

    try {
        service.efetuarLogin(dto.getEmail(), dto.getSenha());
        return ResponseEntity.ok(true);
    } catch (RegraNegocioRunTime e) {
        return ResponseEntity.badRequest().body(e.getMessage());
    }
}
```

Construindo UsuarioController

- obter Investimentos
 - necessário indicar o usuário como parâmetro
 - aqui @RequestParam: '<http://localhost:8080/api/usuarios?usuario=1>'

```
@GetMapping
public ResponseEntity obterInvestimentos(
    @RequestParam("usuario") Long idUsuario) {

    try {
        List<InvestimentoSaldo> invs =
            service.obterSaldos(idUsuario);
        return new ResponseEntity(invs, HttpStatus.OK);
    } catch (RegraNegocioRunTime e) {
        return ResponseEntity.badRequest().body(e.getMessage());
    }
}
```

Testando UserController

- Inicialmente com o **Insomnia**
- Primeiro salvar usuário
 - inicializar o banco de dados
 - inicializar projeto
 - criar nova request, tipo **POST**, e corpo **JSON**
 - inserir a URL: '**http://localhost:8080/api/usuarios**'
 - inserir como teste o seguinte JSON (conforme **UsuarioDTO**):

```
{
  "nome": "Geraldo2",
  "email": "geraldodo3@nca.ufma.br",
  "senha": "123"
}
```

Testando UserController

POST ▼ http://localhost:8080/api/usuarios Send

JSON ▼ Auth ▼ Query Header 1 Docs

```
1 {  
2   "nome": "Geraldo2",  
3   "email": "geraldodo3@nca.ufma.br",  
4   "senha": "123"  
5 }
```

Testando UserController

- clicar em enviar!

201 Created

168 ms

58 B

4 Days Ago ▼

Preview ▼

Header 3

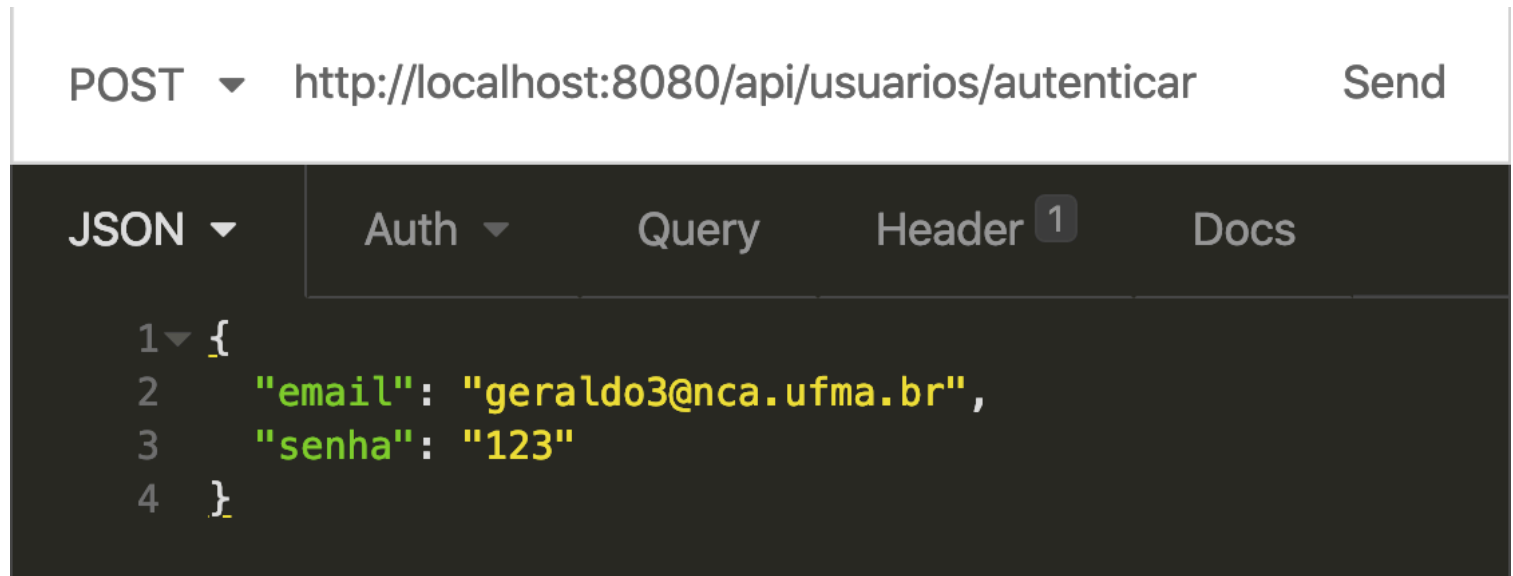
Cookie

Timeline

```
1 ▼ {  
2   "id": 58,  
3   "nome": "Geraldo2",  
4   "email": "gerald3@nca.ufma.br"  
5 }
```

Testando UserController

- Autenticar usuário



The screenshot shows a REST client interface with a dark theme. At the top, a POST request is configured to `http://localhost:8080/api/usuarios/autenticar`. Below the URL bar, there are tabs for 'JSON', 'Auth', 'Query', 'Header' (with a '1' badge), and 'Docs'. The 'JSON' tab is selected, displaying a JSON body with four lines of code: an opening curly brace, a line with `"email": "gerald3@nca.ufma.br",`, a line with `"senha": "123"`, and a closing curly brace.

POST ▼ `http://localhost:8080/api/usuarios/autenticar` Send

JSON ▼ Auth ▼ Query Header 1 Docs

```
1 {  
2   "email": "gerald3@nca.ufma.br",  
3   "senha": "123"  
4 }
```

Testando UserController

- Autenticar usuário

200 OK

23.5 ms

4 B

4 Days Ago ▼

Preview ▼

Header 3

Cookie

Timeline

1 true

Testando UserController

- Obter investimentos

The screenshot shows a REST client interface with the following components:

- Request Bar:** Method `GET`, URL `http://localhost:8080/api/usuarios`, and a `Send` button.
- Tab Bar:** Tabs for `Body`, `Auth`, `Query` (active), `Header`, and `Docs`.
- URL Preview:** A text box containing `http://localhost:8080/api/usuarios?usuario=58` and a copy icon.
- Query Parameters Table:**

	<code>usuario</code>	<code>58</code>			
	<code>New name</code>	<code>New value</code>			

Testando UserController

- Obter investimentos

200 OK

83.5 ms

130 B

4 Days Ago ▼

Preview ▼

Header 3

Cookie

Timeline

```
1 [
2   {
3     "inv": {
4       "id": 24,
5       "nome": "novo investimento2",
6     "usuario": {
7       "id": 58,
8       "nome": "Geraldo2",
9       "email": "gerald3@nca.ufma.br"
10    }
11  },
12  "valor": 610.0
13 }
```

Implementando InvestimentoController

Endpoints de

InvestimentoController

- salvar (post)
- atualizar (put)
 - atualizar investimento existente
- remover (delete)
 - remover investimento existente
- buscarInvestimento (get)
 - obter investimentos por filtro
- obterSaldoInvestimento (get)
 - obter valor total do investimento

Implementando

PosicaoController

- inicializando

```
@RestController
@RequestMapping("/api/investimentos")
public class InvestimentoController {
    @Autowired
    InvestimentoService service;
```

Implementando

InvestimentoController

- Atualizar
 - o id do investimento é pego por path `@PathVariable`
 - Ex: '<http://localhost:8080/api/investimentos/22>'
 - e o json por `@RequestBody` transformado no DTO

```
@PutMapping("{id}")
public ResponseEntity atualizar(
    @PathVariable("id") Long idInvestimento,
    @RequestBody InvestimentoDTO dto) {
    try {
        Investimento inv = Investimento.builder()
            .id(idInvestimento)
            .nome(dto.getNome())
            .usuario(Usuario.builder().id(dto.ge
            .build();

        Investimento salvo = service.atualizar(inv);
        return ResponseEntity.ok(salvo);
    }
    catch (RegraNegocioRunTime e) {
        return ResponseEntity.badRequest().body(e.getMessage());
    }
}
```

Implementando

InvestimentoController

- Remover
 - o id do investimento é pego por path `@PathVariable`
 - Ex: '<http://localhost:8080/api/investimentos/22>'

```
@DeleteMapping("{id}")
public ResponseEntity remover(
    @PathVariable("id") Long idInvestimento) {
    try {
        Investimento inv = Investimento.builder().id(idInvestimento)
            .service.remove(inv);
        return ResponseEntity.ok(HttpStatus.NO_CONTENT);
    }
    catch (RegraNegocioRunTime e) {
        return ResponseEntity.badRequest().body(e.getMessage());
    }
}
```

Implementando

InvestimentoController

- Buscar
 - usando filtros
 - cada campo num parametro
 - somente usuário é obrigatório

```
@GetMapping("/obter")
public ResponseEntity buscarInvestimento(
    @RequestParam(value="usuario", required=true)
        Long idUsuario,
    @RequestParam(value="nome", required=false)
        String nome)
{
    Investimento filtro =
        Investimento.builder()
            .nome(nome)
            .usuario(Usuario.builder()
                .id(idUsuario).build())
            .build();

    List<Investimento> investimentos = service.buscar(filtro);
    return ResponseEntity.ok(investimentos);
}
```


Implementando

InvestimentoController

- Obter Saldo

```
@GetMapping("/saldo")
public ResponseEntity obterSaldoInvestimento(
    @RequestParam("id") Long idInvestimneto) {

    Investimento inv=
        Investimento.builder()
            .id(idInvestimneto).build();

    try {
        Double valor = service.obterValorTotal(inv);
        return ResponseEntity.ok(valor);
    }
    catch(RegraNegocioRunTime e) {
        return ResponseEntity.badRequest().body(e.getMessage());
    }
}
```

Implementando PosicaoController

Endpoints de

PosicaoController

- salvar (post)
- atualizar (put)
- remover (delete)
- buscarPosicao (get)
 - obter posicoes

Implementando

PosicaoController

- Necessário DateFormat para converter strings do json em Date

```
@RestController
@RequestMapping("/api/posicao")
public class PosicaoController {

    @Autowired
    PosicaoService service;

    DateFormat formatter = new SimpleDateFormat("dd/MM/yy");
```

Implementando

PosicaoController

- Salvar

```
@PostMapping
public ResponseEntity salvar(@RequestBody PosicaoDTO dto) {
    try {
        Posicao pos = Posicao.builder()
            .investimento(Investimento
                .builder()
                .id(dto.getIdInvestimento())
                .build())
            .valor(dto.getValor())
            .data(formatter.parse(dto.getData()))
            .build();

        Posicao salvo = service.salvar(pos);
        return new ResponseEntity(salvo, HttpStatus.CREATED);
    } catch (RegraNegocioRunTime e) {
        return ResponseEntity.badRequest().body(e.getMessage());
    } catch (ParseException e) {
        return ResponseEntity.badRequest().body(e.getMessage());
    }
}
```

Implementando

PosicaoController

- Buscar
 - usando filtros
 - investimento é campo obrigatório

```
@GetMapping("/obter")
public ResponseEntity buscarPosicao(
    @RequestParam(value="investimento", required=true) Long idI

    Posicao filtro;

    filtro = Posicao.builder()
        .investimento(
            Investimento.builder()
                .id(idInvestimento)
                .build())
        .build();
    List<Posicao> posicoes = service.buscar(filtro);
    return ResponseEntity.ok(posicoes);
}
```

Demo com Insomnia