



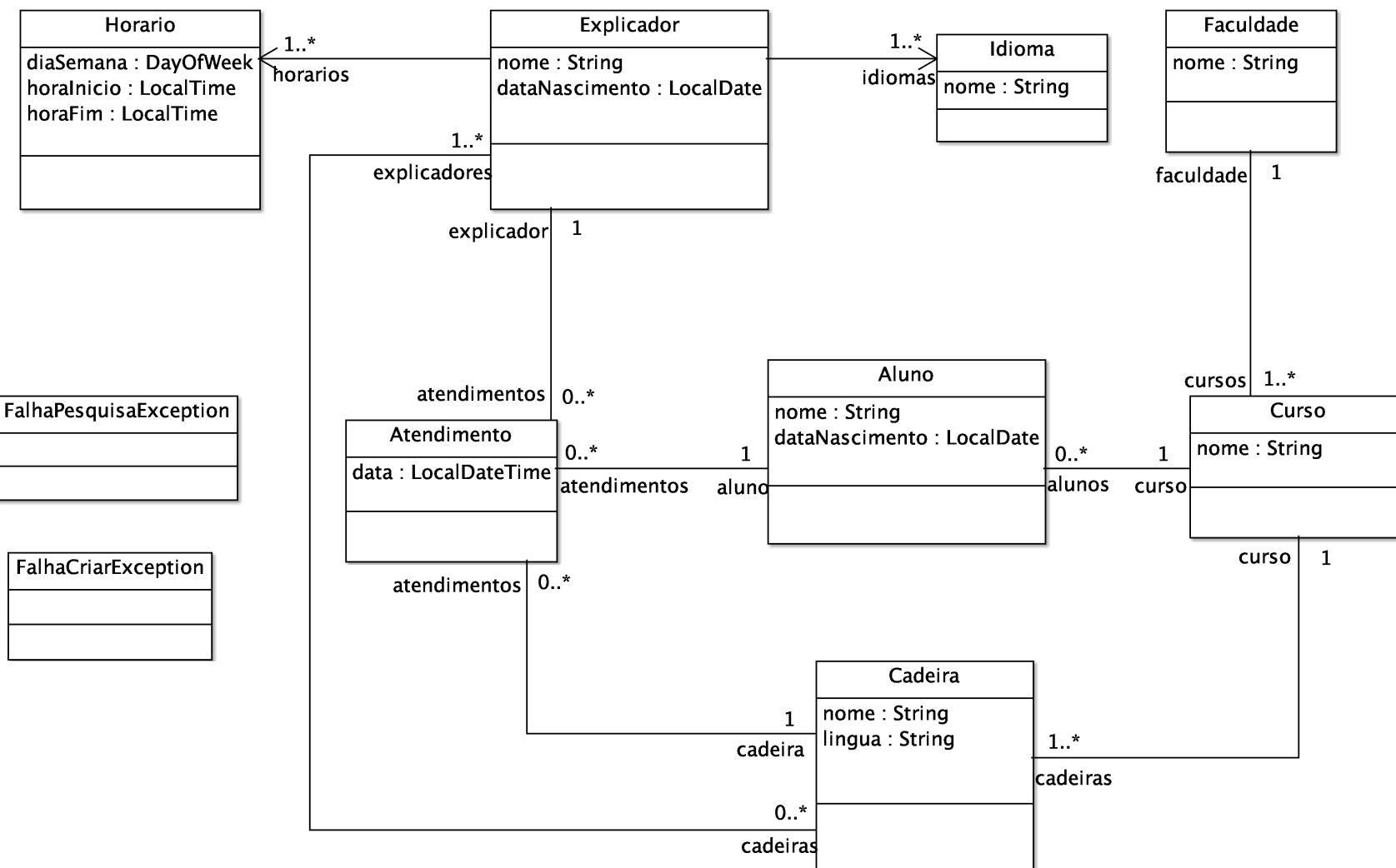
UNIVERSIDADE
FERNANDO ■■■
PESSOA ■■■

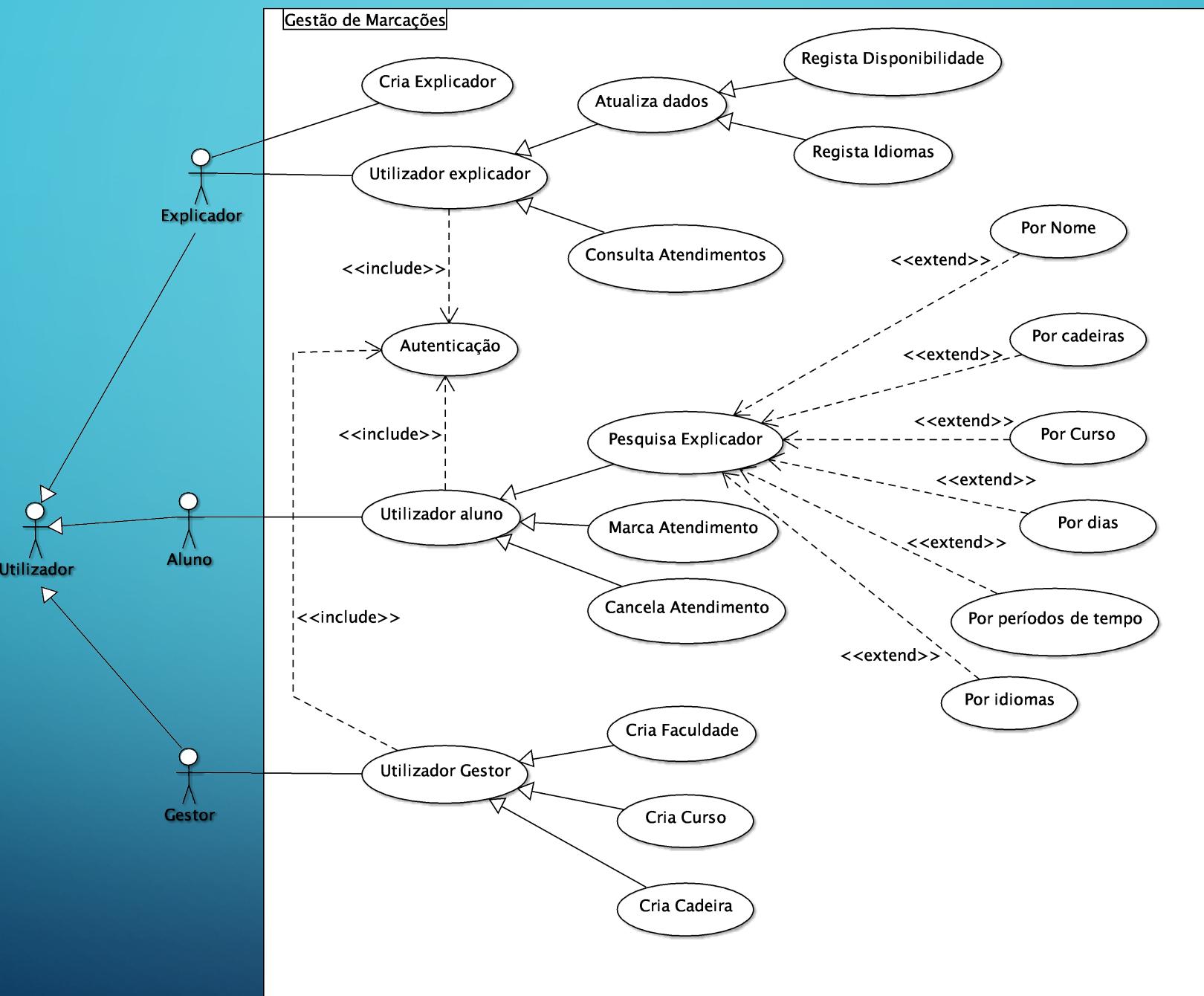
PROJETO PRÁTICO DE ESOF

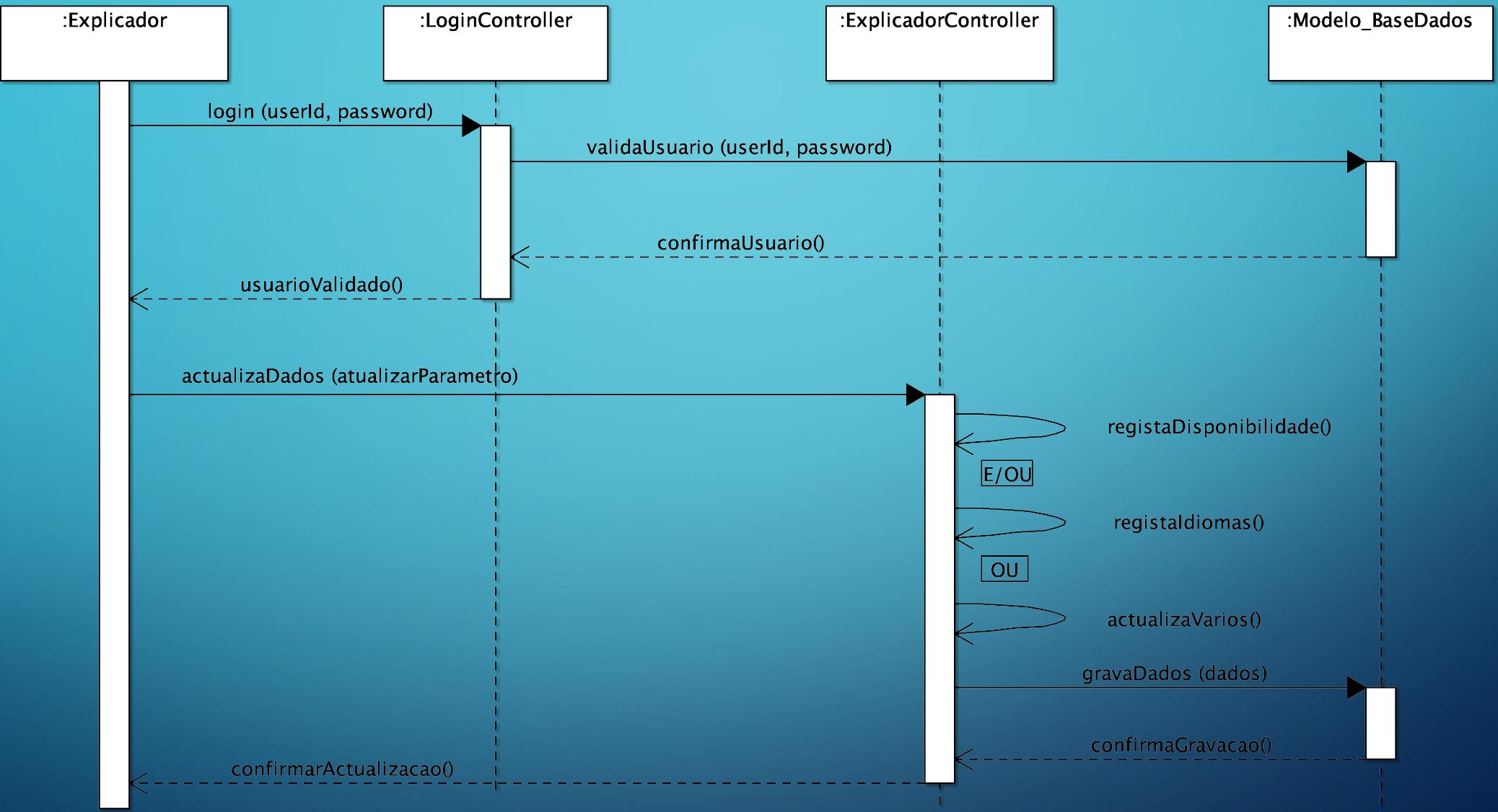
VALTER CARDOSO - 31062

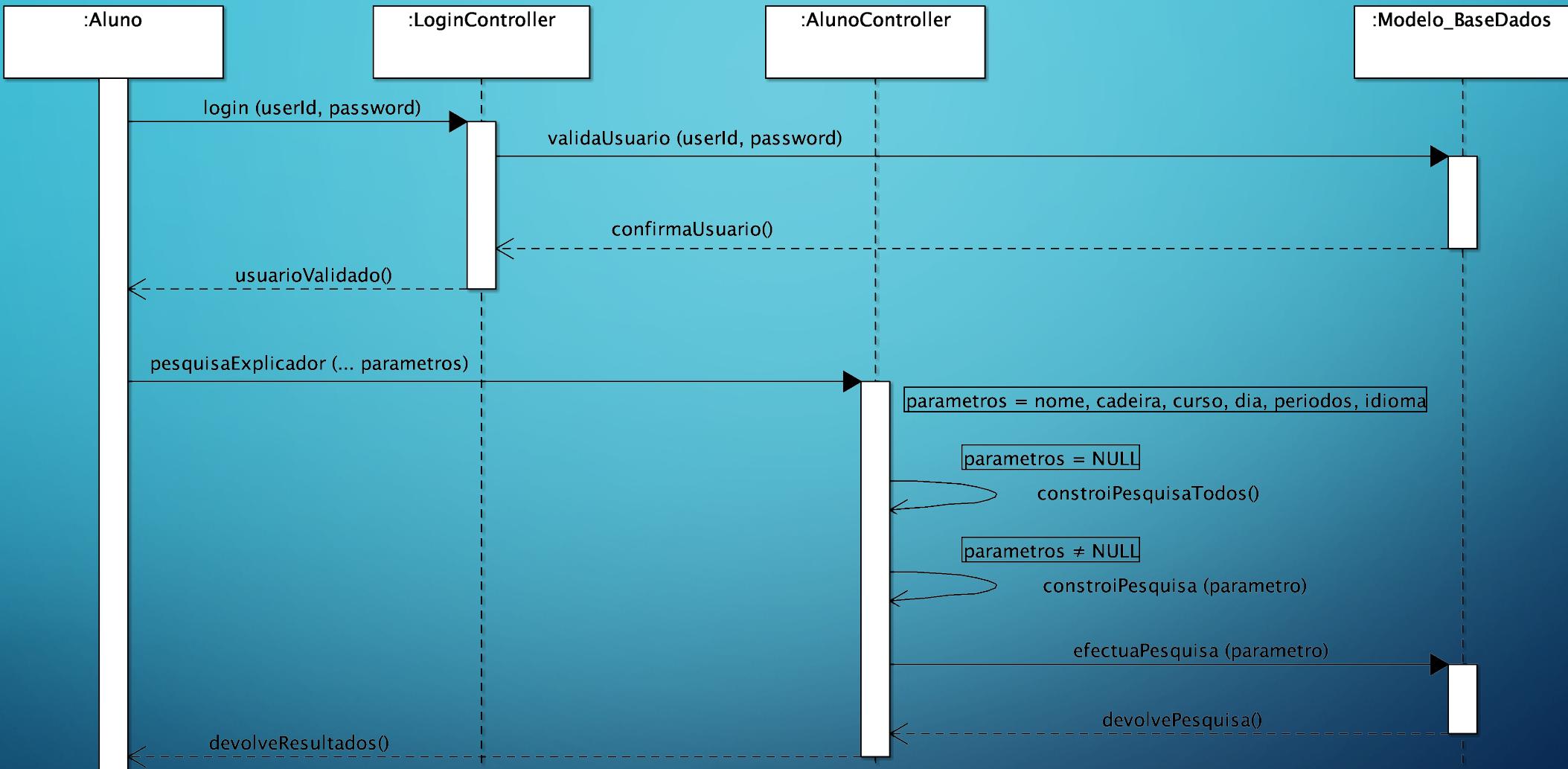
GUSTAVO TEIXEIRA - 21736

gestao_explicoes









ESOF_Projecto ~/Dropbox/exercícios/Eng_Software/Projeto/ESOF_Projecto

- .idea
- .mvn
- Diagramas_UML
- Documentos
- otherFiles
- ▼ src
 - main
 - java
 - com
 - projeto
 - gestao_expliacoes
 - controllers
 - AlunoController
 - exceptions
 - FalhaCriarException
 - FalhaPesquisaException
 - models
 - Aluno
 - Atendimento
 - Cadeira
 - Curso
 - Explicador
 - Faculdade
 - Horario
 - Idioma
 - repositories
 - AlunoRepo
 - AtendimentoRepo
 - CadeiraRepo
 - CursoRepo
 - ExplicadorRepo
 - FaculdadeRepo
 - HorarioRepo
 - IdiomaRepo
 - Bootstrap
 - GestaoDeExpliacoesApplication

```
@Component  
@Transactional  
public class Bootstrap implements ApplicationListener<ContextRefreshedEvent> {  
  
    @Autowired  
    private AlunoRepo alunoRepo; // enunciar os repositorios a utilizar  
  
    private Logger logger = LoggerFactory.getLogger(this.getClass());  
  
    @Override  
    public void onApplicationEvent(ContextRefreshedEvent contextRefreshedEvent) {  
        logger.info("Startup");  
  
        //exemploEntradas();  
        novosDados();  
    }  
  
    private void exemploEntradas(){  
        // Dados para introdução de Datas  
        System.out.println(LocalDate.now());  
        System.out.println(LocalDate.of( year: 2012, Month.SEPTEMBER, dayOfMonth: 21));  
        System.out.println(LocalTime.now());  
        System.out.println(LocalTime.of( hour: 20, minute: 36, second: 56));  
        System.out.println(LocalDateTime.now());  
        System.out.println(DayOfWeek.MONDAY);  
  
        // Para introdução de Idiomas (Idioma em UpperCase)  
        String idioma = "português";  
        System.out.println(idioma.toUpperCase());  
    }  
  
    private void novosDados(){  
  
        Faculdade faculdade1 = new Faculdade( nome: "Ciências e Tecnologia");  
        Faculdade faculdade2 = new Faculdade( nome: "Ciências Humanas e Sociais");  
        Faculdade faculdade3 = new Faculdade( nome: "Ciências da Saúde");  
    }  
}
```

```
@Data  
@Entity  
@NoArgsConstructor  
public class Explicador {  
  
    @Id  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
    private Long id;  
    private String nome;  
    private LocalDate dataNascimento;  
  
    @OneToMany  
    @JoinTable(name = "explicador_horario", joinColumns = @JoinColumn(name = "explicador_id"), inverseJoinColumns = @JoinColumn(name = "horario_id"))  
    private Set<Horario> horarios = new HashSet<>();  
  
    @OneToMany  
    @JoinTable(name = "explicador_idioma", joinColumns = @JoinColumn(name = "explicador_id"), inverseJoinColumns = @JoinColumn(name = "idioma_id"))  
    private Set<Idioma> idiomas = new HashSet<>();  
  
    @OneToMany(mappedBy = "explicador")  
    private Set<Atendimento> atendimentos = new HashSet<>();  
  
    @ManyToMany  
    @JoinTable(name = "explicador_cadeira", joinColumns = @JoinColumn(name = "explicador_id"), inverseJoinColumns = @JoinColumn(name = "cadeira_id"))  
    private Set<Cadeira> cadeiras = new HashSet<>(); // adicionado em "Cadeira"  
  
    // ***** METHODS *****  
  
    public Explicador(String nome, LocalDate dataNascimento) {  
        this.nome = nome;  
        this.dataNascimento = dataNascimento;  
    }  
  
    public void addHorario(Horario horario) { this.horarios.add(horario); }  
  
    public void addIdioma(Idioma idioma) { this.idiomas.add(idioma); }  
  
    public void addAtendimento(Atendimento atendimento){  
        this.atendimentos.add(atendimento);  
        atendimento.setExplicador(this);  
    }  
}
```

```
package com.projeto.gestao_explicacoes.models;

import lombok.Data;
import lombok.NoArgsConstructor;

import javax.persistence.*;
import java.time.LocalDate;
import java.util.HashSet;
import java.util.Set;

@Data
@Entity
@NoArgsConstructor
public class Aluno {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String nome;
    private Integer numero;
    private LocalDate dataNascimento;

    @ManyToOne
    private Curso curso; // adicionado em "Curso"

    @OneToMany(mappedBy = "aluno")
    private Set<Atendimento> atendimentos = new HashSet<>();

    // ***** METHODS *****

    public Aluno(String nome, Integer numero, LocalDate dataNascimento) {
        this.nome = nome;
        this.numero = numero;
        this.dataNascimento = dataNascimento;
    }

    public void addAtendimento(Atendimento atendimento){
        this.atendimentos.add(atendimento);
        atendimento.setAluno(this);
    }
}
```

```
package com.projeto.gestao_explicacoes.repositories;

import com.projeto.gestao_explicacoes.models.Aluno;
import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Repository;

import java.time.LocalDate;
import java.util.Optional;

@Repository
public interface AlunoRepo extends CrudRepository<Aluno, Long> {

    Optional<Aluno> findByNome(String nome);
    Optional<Aluno> findByNumero(Integer numero);
    Optional<Aluno> findByDataNascimento(LocalDate dataNascimento);
}
```

```
@Controller
@RequestMapping("/aluno")
public class AlunoController {

    private Logger logger= LoggerFactory.getLogger(this.getClass());

    @Autowired
    private AlunoRepo alunoRepo;

    //@RequestMapping(method = RequestMethod.GET, produces = MediaType.APPLICATION_JSON_VALUE)
    @GetMapping(value = "", produces = MediaType.APPLICATION_JSON_VALUE)
    public ResponseEntity<Iterable<Aluno>> getAllAlunos(){
        this.logger.info("Recebido um pedido GET");

        return ResponseEntity.ok(this.alunoRepo.findAll());
    }

    @GetMapping(value = "/{id}", produces = MediaType.APPLICATION_JSON_VALUE)
    public ResponseEntity<Aluno> getAlunoById(@PathVariable("id") Long id){
        this.logger.info("Recebido um pedido GET");

        Optional<Aluno> optionalAluno = this.alunoRepo.findById(id);
        if (optionalAluno.isPresent()){
            return ResponseEntity.ok(optionalAluno.get());
        }
        //return ResponseEntity.notFound().build();
        throw new NoAlunoException(id);
    }

    @PostMapping(value = "", produces = MediaType.APPLICATION_JSON_VALUE, consumes = MediaType.APPLICATION_JSON_VALUE)
    public ResponseEntity<Aluno> createAluno(@RequestBody Aluno aluno){
        this.logger.info("Recebido um pedido POST");

        // O mesmo que é feito no método "getAlunosById", mas mais compacto
        if (this.alunoRepo.findByNumero(aluno.getNumero()).isPresent()){
            //return ResponseEntity.badRequest().build();
            throw new AlunoAlreadyExistsException(aluno.getNome());
        }
        //System.out.println(aluno);
        //return ResponseEntity.ok().build();
        return ResponseEntity.ok(this.alunoRepo.save(aluno));
    }
}
```

O QUE FOI FEITO:

- Modelos
- Reppositórios
- Controlador do aluno
- Diagramas UML

DIFICULDADES / DESAFIOS:

- Diagramas UML
- Controladores
- Mapeamento de ManyToMany

PRÓXIMAS TAREFAS:

- Restantes controladores
- Serviços
- Filtros
- Testes