SISTEMA DE RESERVA DE SALAS

RODRIGO MIGUEL TELES DOS SANTOS
LEANDRO BERTOCCHI

CIÊNCIAS DA COMPUTAÇÃO
UNIVERSIDADE DO OESTE DE SANTA CATARINA
SÃO MIGUEL DO OESTE

1.Padrões de projeto

1.1 MVC

Durante o desenvolvimento do projeto, utilizamos e aplicamos alguns padrões que contribuíram para a organização, reutilização de componentes e divisão de tarefas. Sendo uma escolha natural pela estrutura oferecida pelo framework CodeIgniter, o MVC foi amplamente adota como base da arquitetura do sistema.

A camada Model é responsável pela manipulação e registro dos dados. Exemplos aplicados no programa é o RoomModel, UserModel e o ReservationModel

A camada View cuidou da apresentação desses mesmos dados e interação com o usuário. Exemplos são as páginas room/index.php; auth/login.php e home/index.php

A camada Controller intermediou a lógica de fluxo, como RoomController e ReservationController, coordenando as requisições e respostas.

Essa separação torna o sistema muito mais modular, com uma manutenção mais viável e testável, alinhando-se com princípios fundamentais na Engenharia de software. Alem disso, facilitou a separação de tarefas para o desenvolvimento do sistema.

1.2 Middleware e filtros

Outro padrão relevante aplicado foi o uso de filtros de autenticação, representando o padrão Chain of Responsibility. A classe **AuthFilter** atuou como um middleware, interceptando requisições antes que atingissem os controllers protegidos, verificando a existência de uma sessão ativa (user_id) e redirecionando usuários não autenticados para a tela de login.

Foi uma abordagem já oriunda do CodeIgniter, mas, fortaleceu a segurança do sistema e permitiu uma aplicação modular de regras de acesso em rotas específicas, de forma organizada e eficiente.

Essas seriam as duas principais, mas também há outros padrões, como:

1.3 Injeção de Dependência (Implícita)

Função: Promover baixo acoplamento entre as classes.

Aplicação: Controllers instanciam e usam Models sem depender diretamente da lógica de banco.

1.4 Encapsulamento / Separação de Responsabilidades

Função: Ocultar detalhes internos das classes e dividir responsabilidades.

Aplicação: Cada classe (Model, Controller) possui uma única responsabilidade.

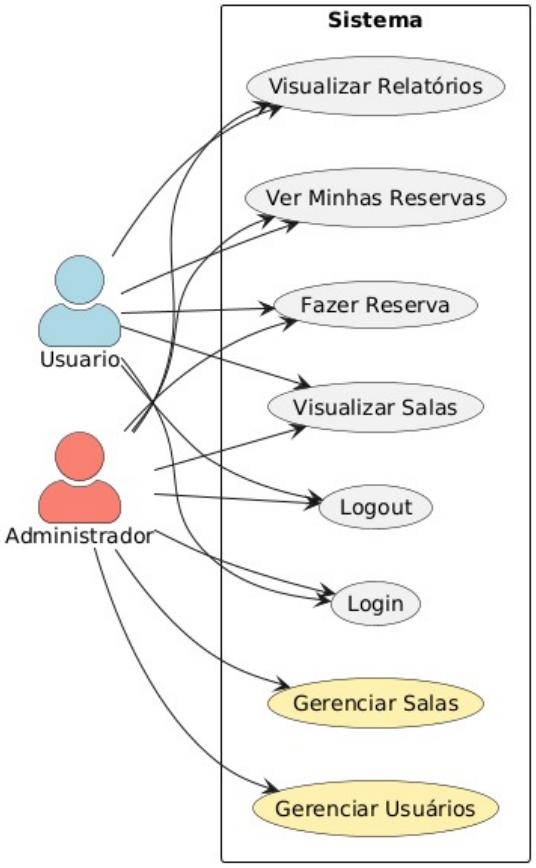
1.5 Factory (Implícito no uso dos Models e Services do CI4)

Função: Centralizar e controlar a criação de objetos.

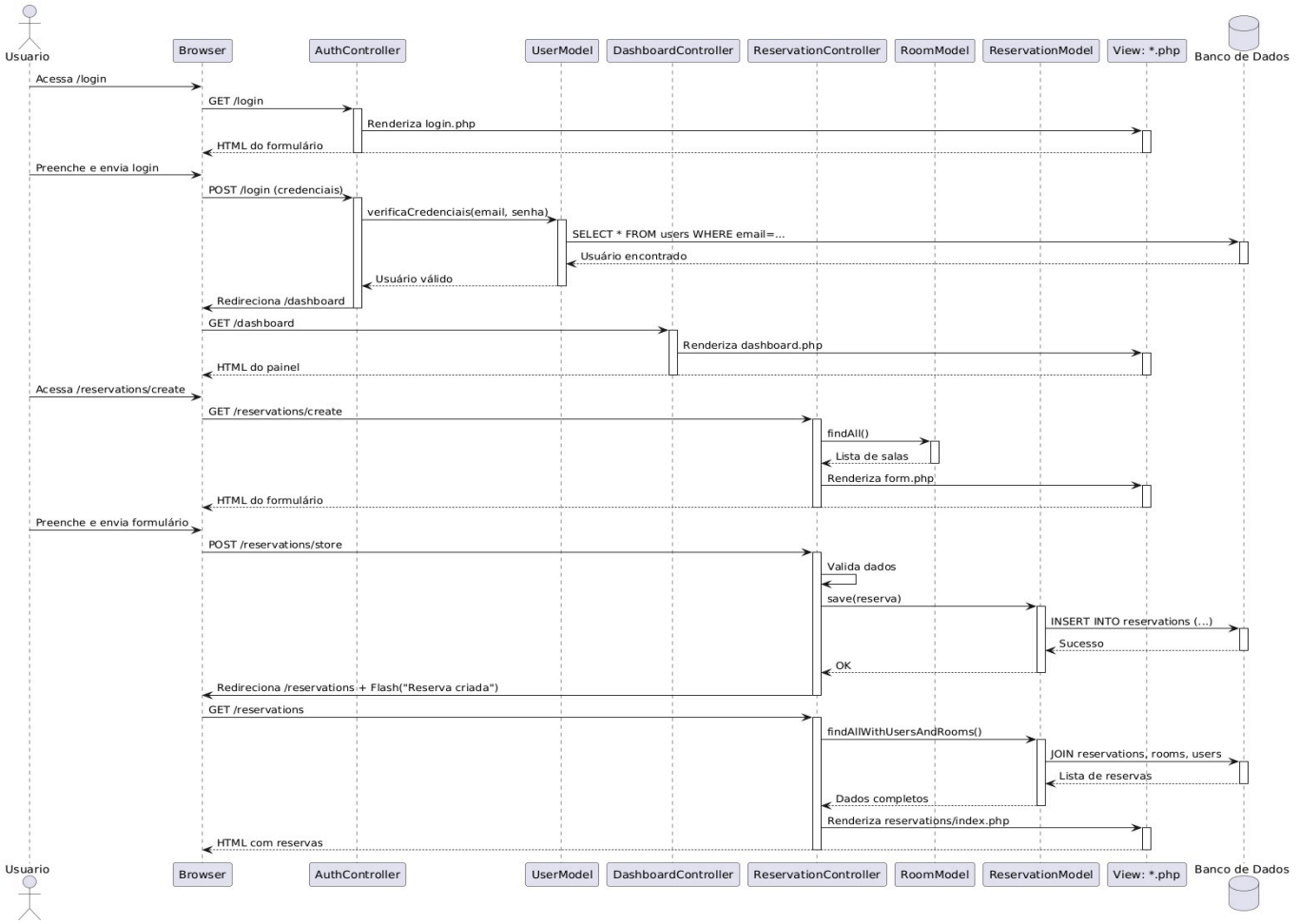
Aplicação: O CodeIgniter instância automaticamente os Models via Model::factory() ou Services

2.Diagramas e Modelos

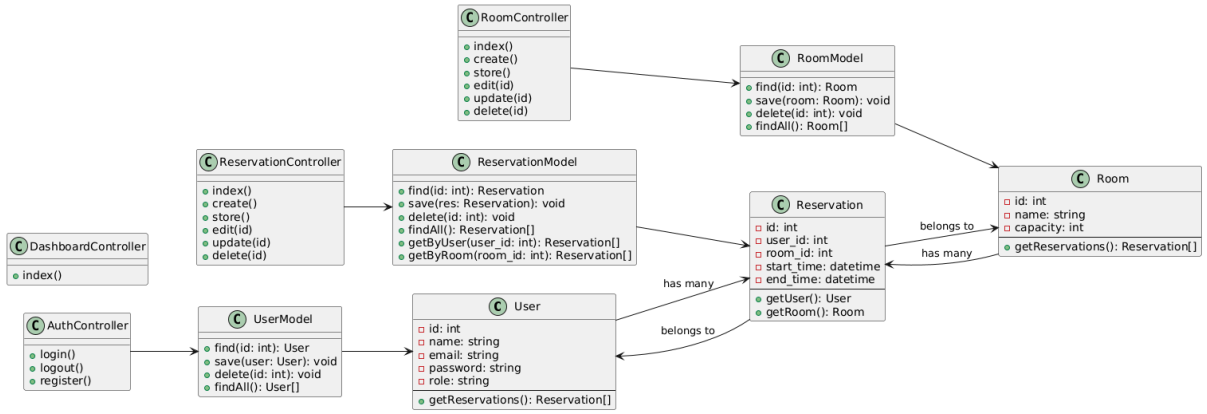
2.1Casos de uso



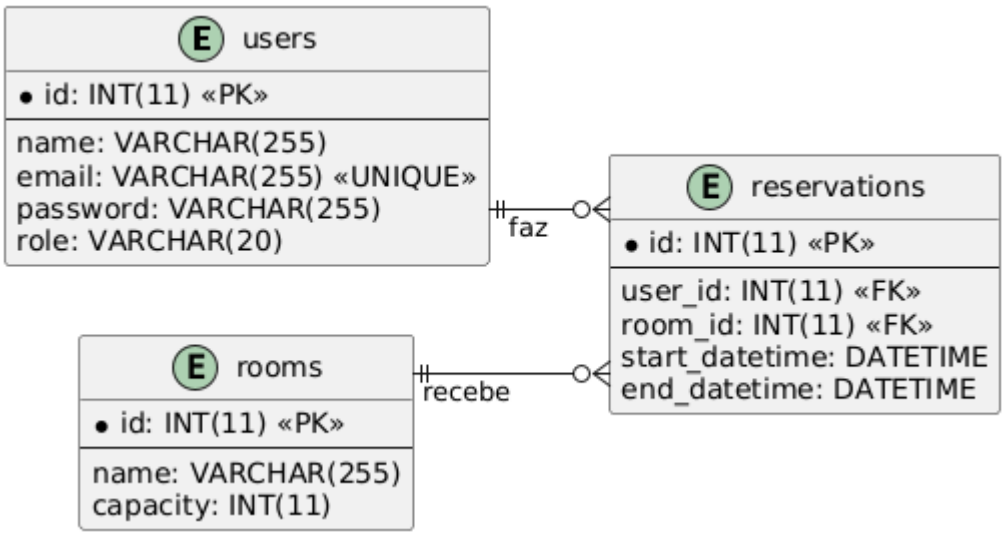
2.2 Diagrama de sequência



2.3 Diagrama de classes



2.4 Modelo Entidade-Relacionamento



3. Validação e Testes

3.1 Teste Unitário

```
namespace Tests\Unit;

use CodeIgniter\Test\CIUnitTestCase;
use App\Models\ReservationModel;

class ReservationModelTest extends CIUnitTestCase
```

```
{
    public function testCheckTimeConflict()
    {
        $model = new ReservationModel();
        $hasConflict = $model->hasTimeConflict(1, date('Y-m-d', strtotime('+1 day')),
'09:30:00', '10:30:00');
        $this->assertTrue($hasConflict);
    }
}
```

```
$ php vendor/bin/phpunit --filter ReservationModelTest

PHPUnit 9.5.21 by Sebastian Bergmann and contributors.

.                                                                   1 / 1 (100%)

Time: 00:00.120, Memory: 10.00 MB

OK (1 test, 1 assertion)
```

3.2 Teste de funcionalidade

```
namespace Tests\Feature;

use CodeIgniter\Test\FeatureTestCase;

class CreateReservationTest extends FeatureTestCase
{
    public function testCreateReservation()
    {
        $response = $this->withSession(['user_id' => 1])
                         ->post('/reservations', [
                            'room_id' => 1,
                            'date' => date('Y-m-d', strtotime('+2 day')),
                            'start_time' => '08:00:00',
                            'end_time' => '09:00:00',
                         ]);

        $response->assertStatus(302);
        $response->assertRedirectTo('/dashboard');
    }
```

}

```
$ php vendor/bin/phpunit --filter CreateReservationTest

PHPUnit 9.5.21 by Sebastian Bergmann and contributors.

.                                                                   1 / 1 (100%)

Time: 00:00.213, Memory: 12.00 MB

OK (1 test, 2 assertions)
```

3.3 Teste de Carga

```
namespace Tests\Performance;

use CodeIgniter\Test\CIUnitTestCase;
use App\Models\ReservationModel;

class StressReservationTest extends CIUnitTestCase
{
    public function testMultipleReservations()
    {
        $model = new ReservationModel();
        $date = date('Y-m-d', strtotime('+3 day'));
        $room_id = 1;

        for ($i = 0; $i < 100; $i++) {
            $start = sprintf('%02d:00:00', ($i % 12) + 1);
            $end = sprintf('%02d:00:00', (($i % 12) + 2));

            $data = [
                'user_id' => 1,
                'room_id' => $room_id,
                'date' => $date,
                'start_time' => $start,
                'end_time' => $end,
            ];

            $model->insert($data);
        }

        $this->assertGreaterThanOrEqual(100, $model->where('date',
$date)->countAllResults());
```

}}

```
$ php vendor/bin/phpunit --filter StressReservationTest

PHPUnit 9.5.21 by Sebastian Bergmann and contributors.

.                                                                   1 / 1 (100%)

Time: 00:03.152, Memory: 18.00 MB

OK (1 test, 1 assertion)
```