INSTITUTO DE INFORMÁTICA
UFRGS

STUDIFY

Autores: Cristopher dos Santos Filho; Luís Filipe Moura; Milena Silva; Nickolas Xisto Machado; Pedro Schuck de Azevedo.

Disciplina: INF01120 - Técnicas de Construção de Programas

Professora: Karina Kohl

Propósito

- Qual o principal problema de um estudante universitário?

Propósito

- O que faz um estudante 

?

Propósito

- O que faz um estudante .infeliz ?

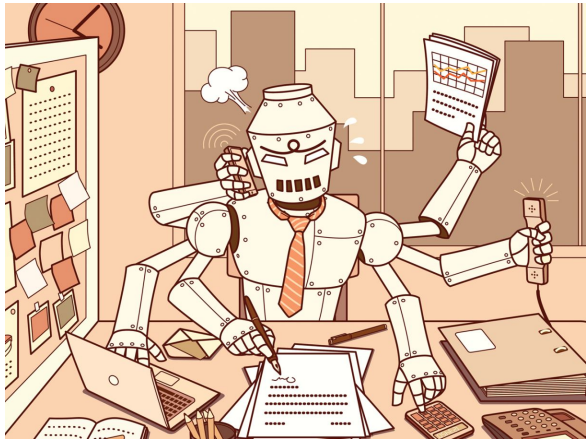
Propósito

- O que faz um estudante .infeliz ?

Propósito

- O que faz um estudante .infeliz ?

Propósito



- Cursar várias disciplinas, cada uma com múltiplas provas, trabalhos e exercícios;
  - Requer uma boa dose organização e disposição.
- Porém, este processo pode gastar
  - Muito tempo e energia.
- Assim, a aplicação Studify se propõem a criar
  - Uma rotina de estudos semestral
    - Baseada em prioridades e nas tarefas de cada cadeira.
- Para que o estudante possa focar no estudo em si.

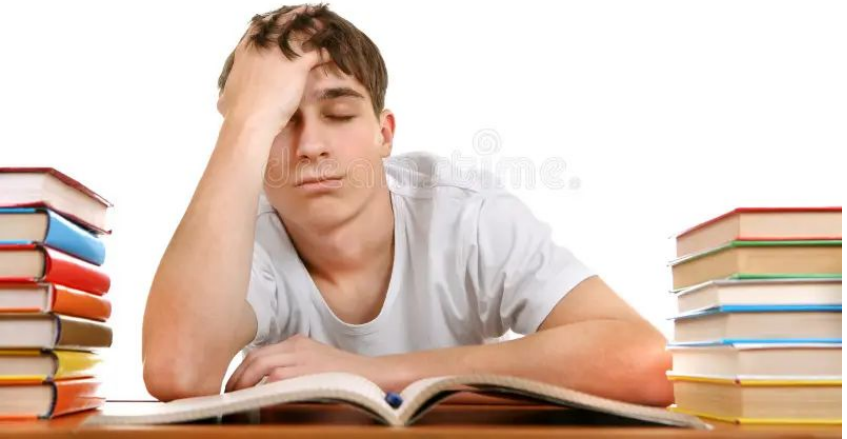


Diagrama de Classes

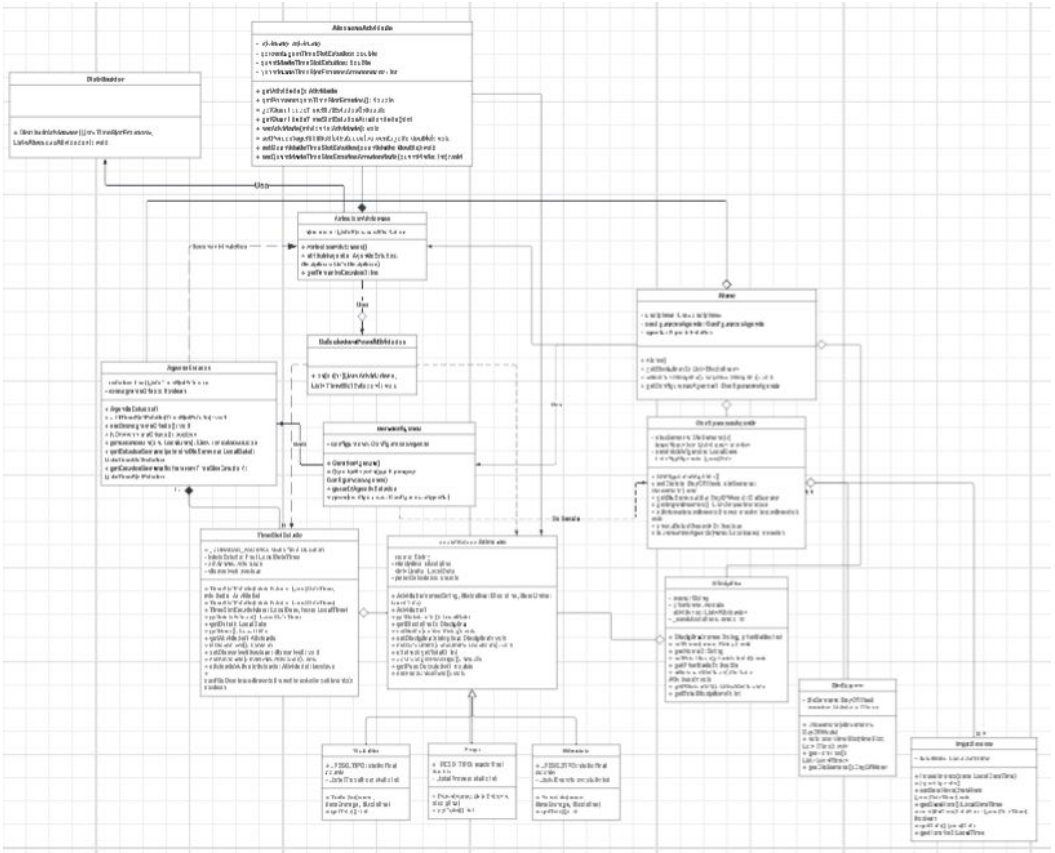
- 

Diagrama de Classes

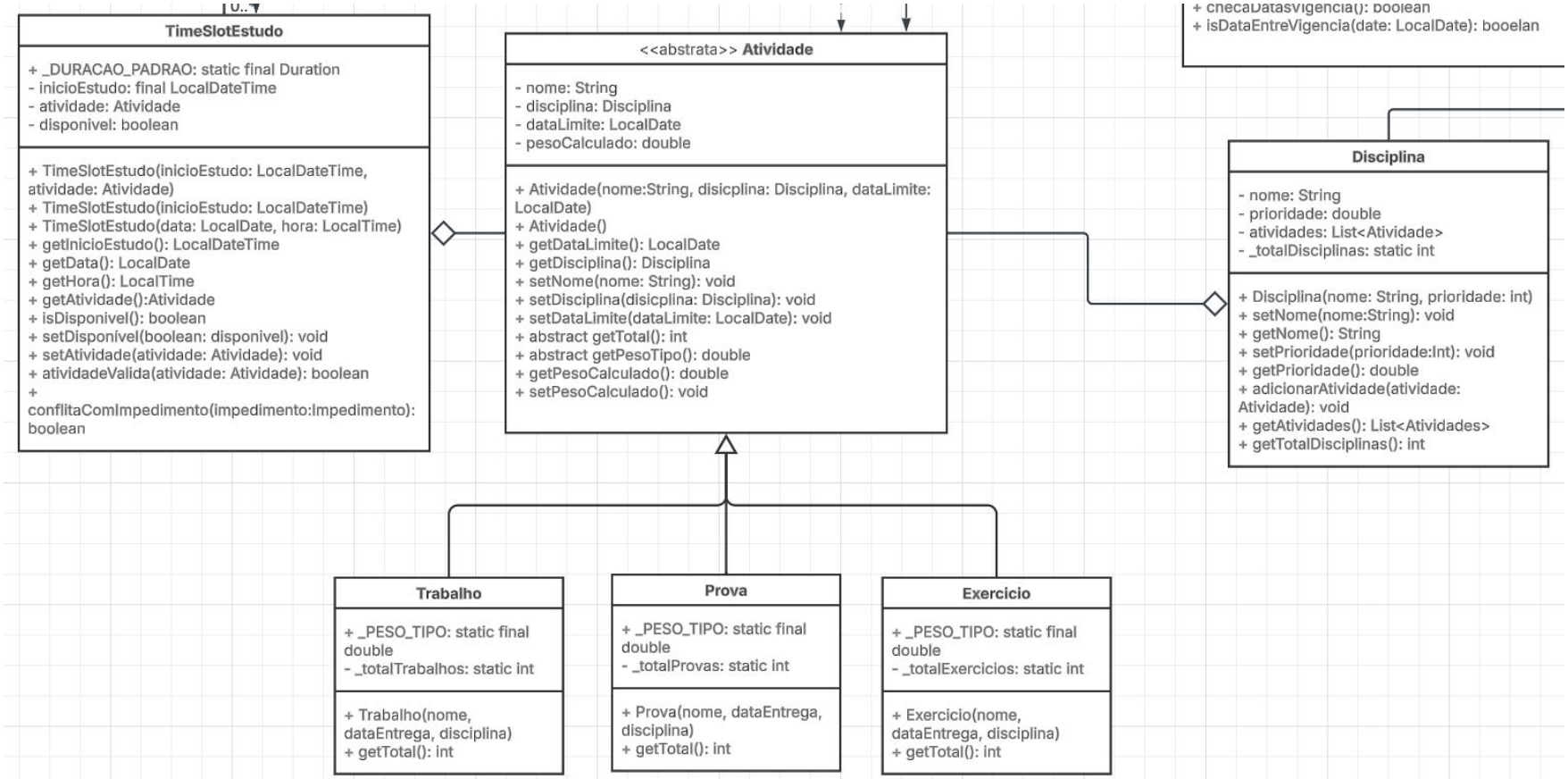


Diagrama de Classes

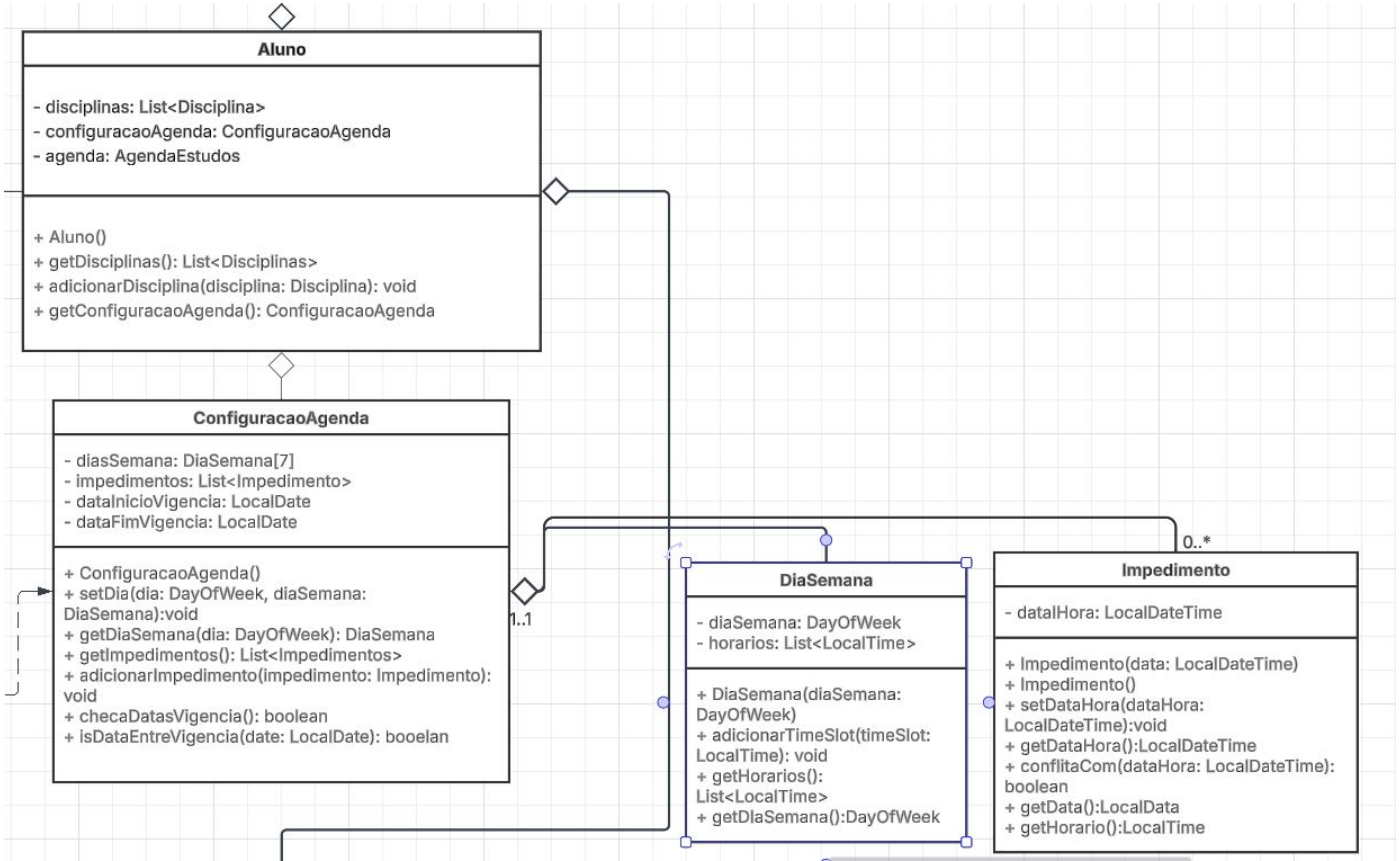
- 

Diagrama de Classes

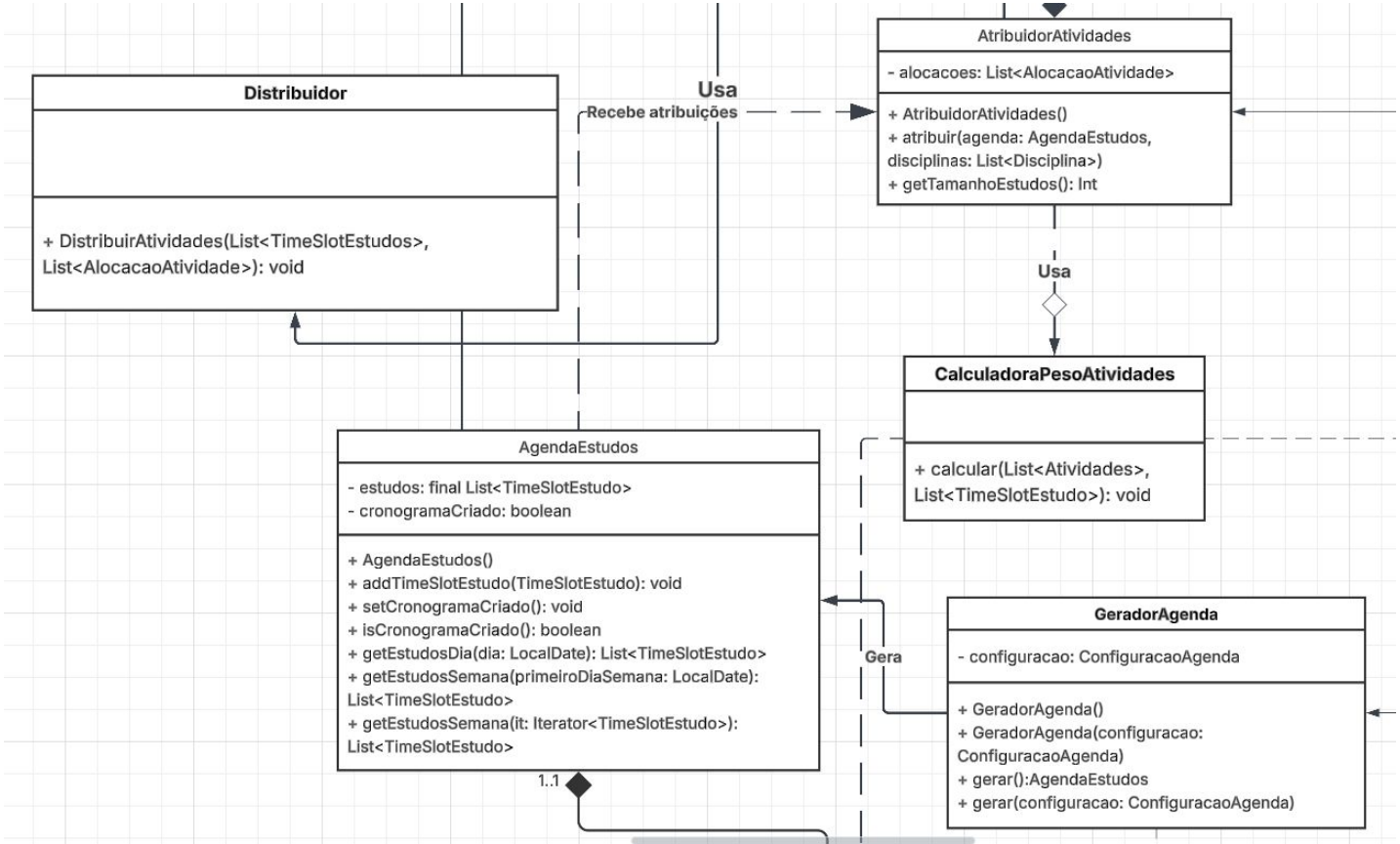
- 

Diagrama de Classes

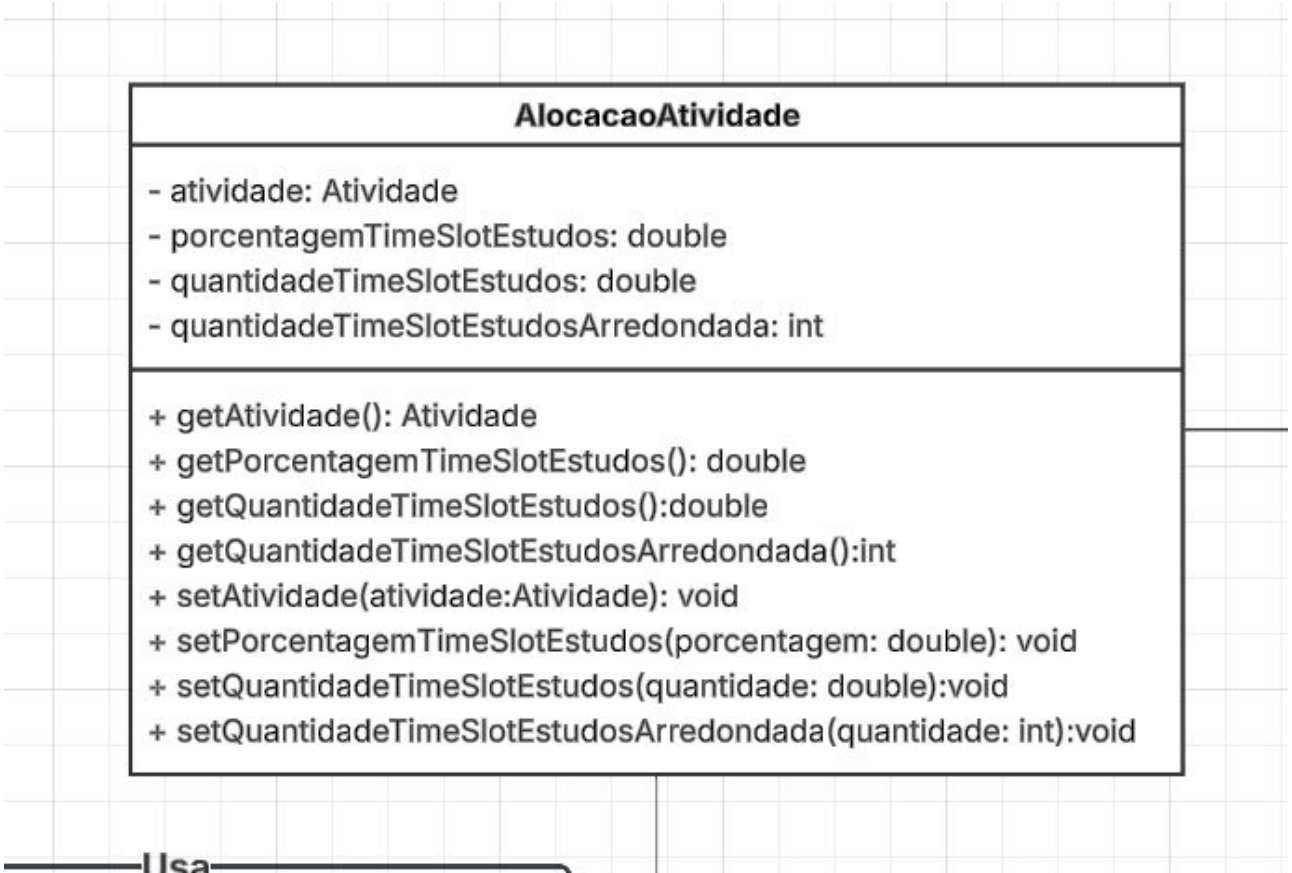
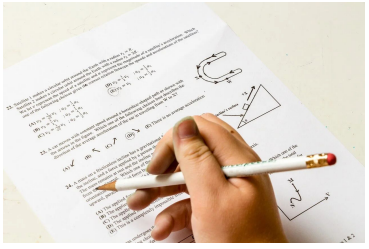
- 

Diagrama de Classes

Prova 1 - 08/10/2025

Prova 2 - 26/11/2025



- Classes modelo, utilizadas como base para armazenar dados e definir padrões:
- Disciplina;
- Impedimento;
- ConfiguracaoAgenda;
- Aluno;
- TimeSlotEstudo;
- AgendaEstudos;
- DiaSemana;
- AlocacaoAtividade;
- Atividade;
- Prova;
- Trabalho;
- Exercício.

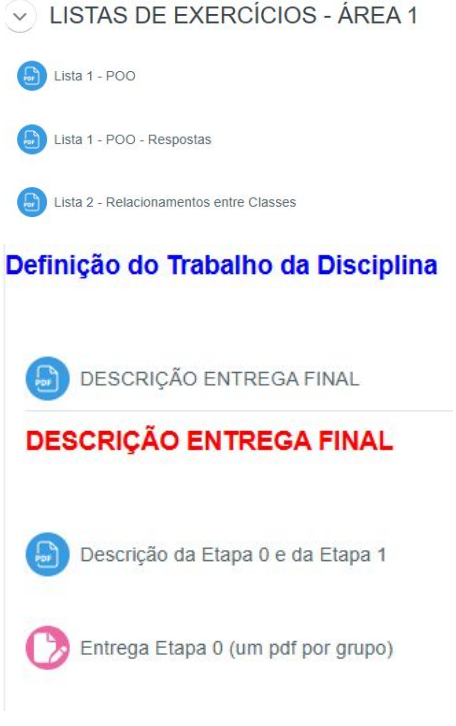
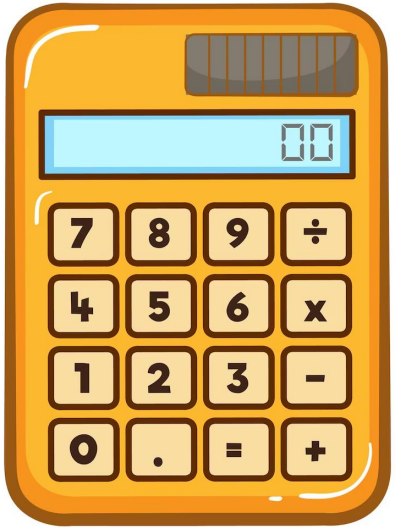
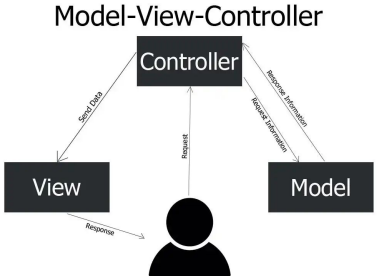


Diagrama de Classes

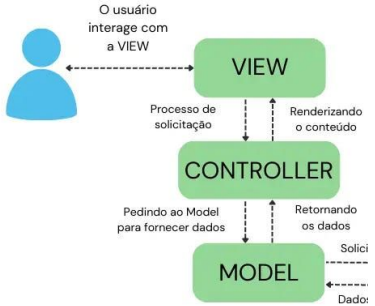
- Classes controladoras, empregadas para manipular e preencher as classes modelo:
- GeradorAgenda;
- AtribuidorAtividades;
- DistribuidorAtividades;
- CalculadoraPesoAtividades.

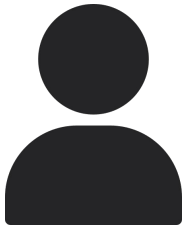


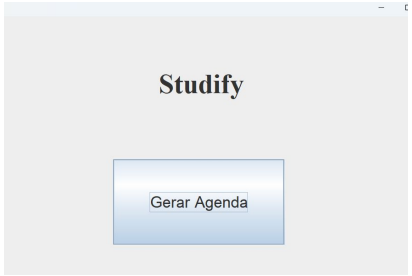
Código



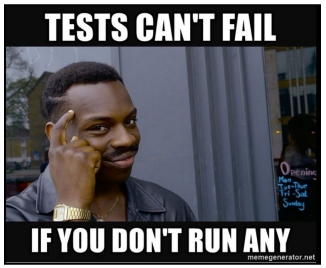
- Uma vez que os dados do usuário são registrados pela interface,
  - Eles passam por uma validação e conversão;
- Os dados convertidos são armazenados em classes modelo
- Classes controladoras utilizam os modelos e geram a agenda
- Informações da agenda são convertidas e exibidas na interface

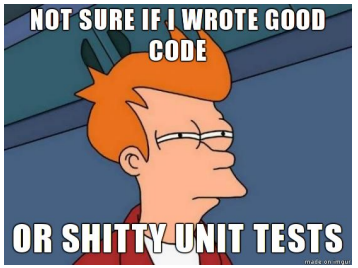




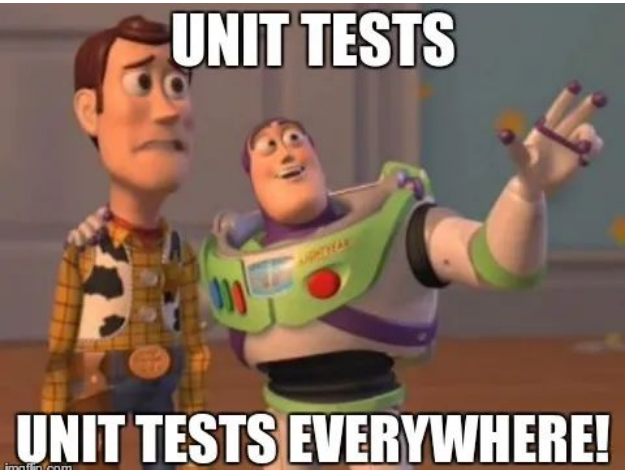


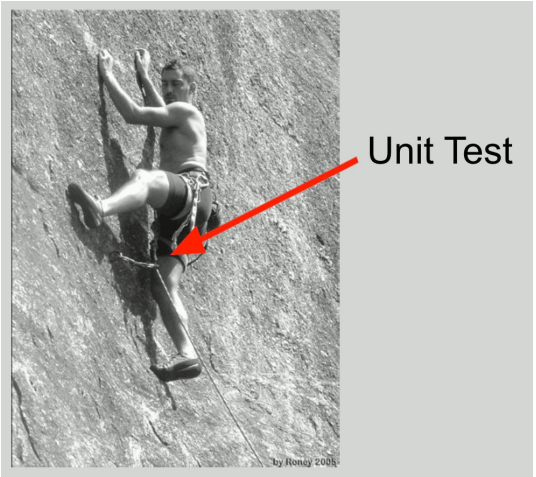
Testes unitários

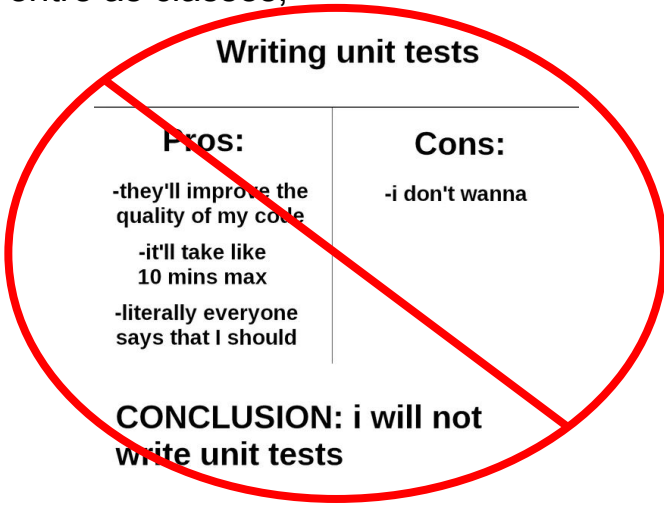




- Algumas das várias classes que foram testadas: ConfiguracaoAgenda, TimeSlotEstudo, Prova, Trabalho, Aluno, AgendaEstudos, AtribuidorAtividades e etc.
- Focou-se em testar métodos únicos de cada classe, bem como
  - Determinados fluxos de dados;
  - Exceções e casos incomuns;
- Mostraram-se de grande utilidade para comparar inconsistências entre as classes;







Testes unitários

```java
@Test(expected = IllegalArgumentException.class)
public void testSetDiaComDiaSemanaIncompatível() {
    ConfiguracaoAgenda config = new ConfiguracaoAgenda();

    // DiaSemana criado para TERÇA mas associado à chave de SEGUNDA
    DiaSemana diaTerça = new DiaSemana(DayOfWeek.TUESDAY);
    diaTerça.adicionarTimeSlot(LocalTime.of(8, 0));

    config.setDia(DayOfWeek.MONDAY, diaTerça);
}


@Test
public void testConflitaCom() {
    LocalDateTime dataHora = LocalDateTime.of(2023, 10, 1, 10, 0);
    Impedimento impedimento = new Impedimento(dataHora);
    assert(impedimento.conflitaCom(dataHora));
    assert(!impedimento.conflitaCom(LocalDateTime.of(2023, 10, 1, 12, 0)));
}
```

```java
@Test
public void fluxoCompletoUsuarioReal() {
    LocalDate inicio = LocalDate.of(2025, 1, 6); // segunda
    LocalDate fim = inicio.plusDays(5); // sabado
    ConfiguracaoAgenda config = new ConfiguracaoAgenda(inicio, fim);

    DiaSemana segunda = new DiaSemana(DayOfWeek.MONDAY);
    segunda.adicionarTimeSlot(LocalTime.of(8, 0));
    DiaSemana terca = new DiaSemana(DayOfWeek.TUESDAY);
    terca.adicionarTimeSlot(LocalTime.of(8, 0));
    DiaSemana quarta = new DiaSemana(DayOfWeek.WEDNESDAY);
    quarta.adicionarTimeSlot(LocalTime.of(8, 0));
    DiaSemana quinta = new DiaSemana(DayOfWeek.THURSDAY);
    quinta.adicionarTimeSlot(LocalTime.of(8, 0));
    DiaSemana sexta = new DiaSemana(DayOfWeek.FRIDAY);
    sexta.adicionarTimeSlot(LocalTime.of(8, 0));

    config.setDia(DayOfWeek.MONDAY, segunda);
    config.setDia(DayOfWeek.TUESDAY, terca);
    config.setDia(DayOfWeek.WEDNESDAY, quarta);
    config.setDia(DayOfWeek.THURSDAY, quinta);
    config.setDia(DayOfWeek.FRIDAY, sexta);

    GeradorAgenda gerador = new GeradorAgenda(config);
    AgendaEstudos agendaGerada = gerador.gerar();

    Disciplina disciplinaTCP = new Disciplina("TCP", 1.5);
    Disciplina disciplinaSD = new Disciplina("SD", 1.0);

    Prova provaTcp = new Prova("Prova TCP", inicio.plusDays(2), disciplinaTCP); // quarta
    Trabalho trabSd = new Trabalho("Trabalho SD", inicio.plusDays(5), disciplinaSD); // sexta

    disciplinaTCP.adicionarAtividade(provaTcp);
    disciplinaSD.adicionarAtividade(trabSd);
```