

**Projeto Final MC322 - Programação Orientada a Objetos**  
**“GreatGrades - Organizador da vida universitária”**

*GreatGrades*



**Participantes:**

- Caio Azevedo Dal Porto - RA 256709
- Lucas Cardoso - RA 246125
- Lucas Gugel Maciel - RA 260579
- Matheus Hencklein Ponte - RA 247277

## 1 - Sobre o projeto

O 'GreatGrades' é um aplicativo de estudos voltado para o estudante universitário. O aplicativo conta com uma interface amigável e auxilia o estudante a organizar sua rotina de estudos, mostrando informações úteis à sua graduação, tais como: disciplinas matriculadas, atividades, prazos, faltas cometidas e a situação da média. Disponível em: <<https://github.com/lcardosott/greatGrades>>.

## 2 - Processo

### 2.1 Esboço Inicial

Com o objetivo em mente, passamos para o aplicativo Figma e começamos a esboçar o que futuramente seria a cara do nosso aplicativo. O resultado está representado na Figura 1. Existem 4 telas principais: menu de login e cadastro, menu principal que contém o *overview* de cada matéria do estudante, espaço de visualização da matéria, contendo avaliações, prazos e notas, e uma tela para ver e adicionar avaliações.

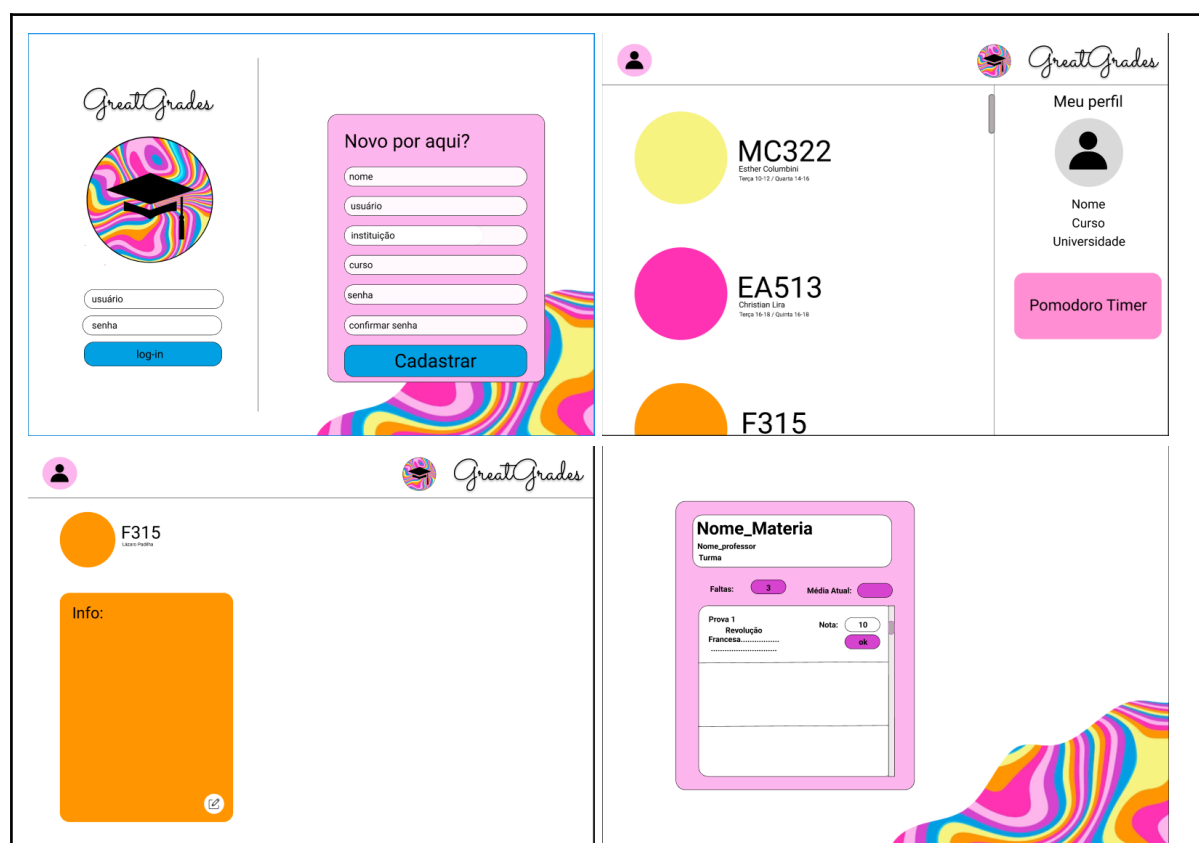


Figura 1 - Esboço Figma.

### 2.2 Definindo Classes

A partir do esboço inicial fomos capazes de realizar o Product Backlog (um conjunto de features e obrigatoriedades necessárias para contemplar todas as

funcionalidades do aplicativo) através das User Stories (histórias simples de se completar da forma “As a \_\_, I need \_\_ so that \_\_”).

Por se tratar de um processo muito importante para um projeto, diversas semanas foram depreendidas para essa etapa. O diagrama UML pode ser encontrado na Figura 2;

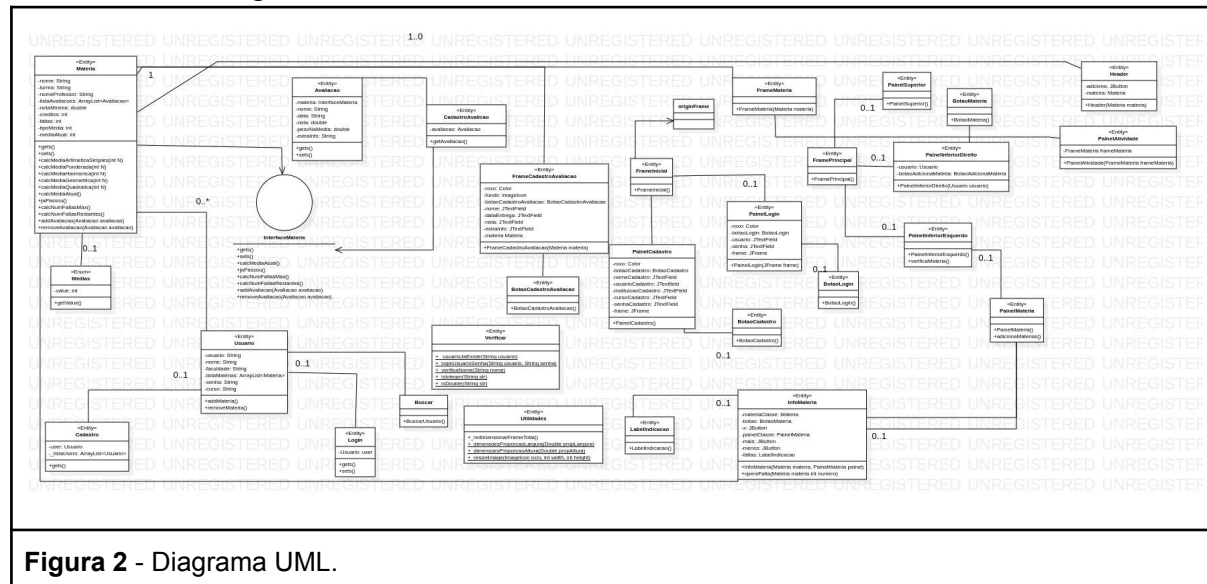


Figura 2 - Diagrama UML.

## 2.3 Programando

Todas as tarefas foram então divididas, com pessoas voltadas para o BackEnd e outras voltadas para o FrontEnd. No entanto, todos desempenham papéis complementares sempre que necessário.

## 3 - Especificações do Projeto

### 3.1 Ferramentas Utilizadas

O projeto foi desenvolvido inteiramente em Java e a biblioteca Swing foi a utilizada para a interface gráfica. O modelo de projeto optado foi o MVC (Model, View, Controller). Em Model, foram descritas as classes e seus respectivos atributos, bem como getters e setters. Em View, o enfoque foi a implementação da interface gráfica. Por fim, em Controller, existem todos os métodos que manipulam o banco de dados, instanciam objetos e os adicionam a listas, verificam validade de entradas, entre outras funções todas relacionadas ao back-end.

### 3.2 Classes e Métodos Definidos

Como pode-se observar na Figura 2 as principais classes criadas foram:

#### Model

**Usuário:** referente à pessoa que utiliza o programa. Possui dados básicos, como nome de usuário (username), nome da pessoa, da instituição de ensino, do curso e senha, bem como a lista de matérias que esse usuário cadastrou.

**Matéria:** cada matéria pertencente a um usuário, portanto, adicionada em sua lista de matérias. Possui atributos como o nome da matéria, a turma, o nome do

professor, a nota mínima para aprovação, os créditos que vale, o tipo de média (discutido melhor abaixo). Além disso, possui também o usuário que a cadastrou, a média atual, baseada nas avaliações já cadastradas, e uma lista de avaliações também cadastradas pelo usuário.

**Avaliação:** uma avaliação pertence a uma matéria que, por sua vez, pertence a um usuário. Possui os seguintes atributos: nome, data, nota, peso na média e informações extras. Também possui uma matéria, que permite associá-la à mesma.

**Médias:** as médias são separadas por tipos, variando com a matéria em questão. O tipo de média em uma matéria é dado no momento de sua criação. Todas herdam da classe abstrata *OriginMedia*, que possui a lista de avaliações. Cada vez que uma nova avaliação é adicionada, a média para aquela matéria é recalculada, o que permite que o usuário se atente à sua nota atual.

- **Média Aritmética:** Dada por  $\frac{1}{N} \sum_{i=0}^N (A_i)$ , em que N é a quantidade de avaliações e  $A_i$ , a i-ésima avaliação.
- **Média Ponderada:** Dada por  $\sum_{i=0}^N (P_i A_i)$ , em que N é a quantidade de avaliações,  $A_i$ , a nota da i-ésima avaliação e  $P_i$  o peso daquela avaliação,  $0 < P_i \leq 1$ .
- **Média Geométrica:** Dada por  $\sqrt[N]{A_1 \times A_2 \times \dots \times A_N}$  em que N é a quantidade de avaliações cadastradas e  $A_i$  a nota da i-ésima avaliação.
- **Média Harmônica:** Dada por  $N / (\sum_{i=1}^N \frac{1}{A_i})$  em que N é a quantidade de avaliações cadastradas e  $A_i$  é a nota da i-ésima avaliação. Note que  $A_i$  deve ser diferente de zero.
- **Média Quadrática:** Dada por  $\sqrt{(\sum_{i=1}^N A_i^2)/N}$  em que N é a quantidade de avaliações cadastradas e  $A_i$  é a nota da i-ésima avaliação.

### View

**Utilidades:** classe estática com funções que auxiliam no desenvolvimento front-end, com funções de obter medidas proporcionais a tela do usuário e redimensionar tamanho de figuras.

**OriginFrame:** classe estática que possui informações básicas sobre a construção de uma janela: ícone e título da janela e seu tamanho. Herda de *JFrame*, a classe para construção de janelas

**Demais classes:** herdam as especificações de *OriginFrame*, e correspondem às outras telas do front-end, a saber: tela de início (com log-in ou

cadastro), menu principal (com todas as matérias para aquele usuário), tela de cadastro de matéria, tela de visualização de matéria e tela de cadastro de avaliação. Também há classes de auxílio para cada tela, como botões, painéis e tarjas de texto.

### **Controller**

**App:** Ponto de partida para a execução do programa, chamando os métodos de leituras de arquivos, para a criação da lista de usuários. Também inicia a tela de início, que possui os campos de log-in ou cadastro.

**AddMateria:** Classe voltada à criação de uma nova matéria e adição à lista de matérias daquele usuário. Faz verificação de campos vazios no momento da criação e se os campos referentes a números (créditos, nota mínima) estão devidamente preenchidos.

**Buscar:** realiza a busca por um usuário na lista de usuários, busca de uma matéria em seu arquivo CSV e busca de uma avaliação em seu arquivo CSV.

**Cadastro:** possui métodos referentes ao cadastro de um novo usuário. Verifica se os campos de texto estão devidamente preenchidos e se o usuário já existe (esta última é feita pela classe Verificar). Caso passe, um novo usuário é instanciado, adicionado ao CSV (classe OriginFile) e à lista de usuários.

**CadastroAvaliacao:** instancia uma nova avaliação e a adiciona à lista de avaliações da matéria, como também ao CSV de avaliações. O nome de uma avaliação deve ser único, por isso, verifica se o nome já não existe na lista de avaliações daquela matéria.

**Deletar:** segundo ordem do usuário, realiza operações de deletar: remove uma matéria da lista de matérias do usuário e remove uma avaliação da lista de avaliações de uma matéria. Também faz a manipulação necessária no arquivo CSV de cada uma, pois esses dados precisam ser removidos.

**Inicializar:** classe que lê os arquivos CSV no início da execução do programa, obtendo os usuários cadastrados. Quando o log-in em um usuário válido é realizado, também instancia as matérias vinculadas à ele e suas respectivas avaliações, adicionando-as às listas corretas.

**Login:** verifica se ambos os campos do log-in estão preenchidos. Também, verifica se o usuário desejado já está cadastrado no sistema e se o nome de usuário e senha condizem com as informações dadas nos campos de texto. Essas verificações são feitas pelas classes Buscar e Verificar.

**OriginFile:** classe que faz adições de linhas aos respectivos arquivos CSV. novo usuário, matéria ou avaliação.

**Verificar:** realiza as verificações para outras classes: se o usuário já existe, se o nome de usuário dado condiz com a senha (para que seja feito o log-in), se a matéria já existe para aquele usuário e se a avaliação já existe para aquela matéria. Também faz verificações mais simples, como se uma dada String pode ser transformada para um inteiro ou double, e se uma String possui apenas letras.

## **3.3 Salvando em arquivos**

Algo pensado desde a concepção do projeto foi um modo de salvar as informações que um usuário insere, como suas matérias e avaliações de cada uma, sem que sejam perdidas após o encerramento de sua execução, possibilitando a sua continuidade.

Assim, a melhor solução encontrada pelo grupo foi, ao mesmo tempo, a mais simplista: organização em arquivos CSV com cabeçalho.

Quando um cadastro de usuário, matéria ou avaliação é realizado, esse é salvo em um respectivo arquivo CSV com informações necessárias para, quando o programa for novamente executado e o usuário realiza o log-in, suas matérias estejam vinculadas à sua conta e, por sua vez, as avaliações estejam vinculadas às suas matérias.

Conforme a **Figura 3**, abaixo, é notável a presença dos campos “usuario”, no arquivo Materias.csv, e os campos “usuario” e “matéria” no arquivo Avaliacoes.csv. Isso permite que, quando é feito um log-in, as matérias e avaliações sejam instanciadas corretamente e adicionadas às listas corretas, sem que haja ambiguidade entre usuários com mesmo nome de matérias ou com mesmo nome de avaliações. Users.csv é de importância para verificar se um usuário existe e já foi cadastrado no sistema. Também, evita que outro usuário de mesmo username seja criado.

← Users.csv					
	A	B	C	D	E
1	USER	NOME	INSTITUICAO	CURSO	SENHA
2	username	Nome Sobrenome	Unicamp	Eng. de Comp.	senhasupersecreta

← Materias.csv								Abrir com ▾
	A	B	C	D	E	F	G	H
1	USER	NOME_MATERIA	TURMA	NOME_PROFESSOR	NOTA_MINIMA	NUMERO_CREDITO	NUMERO_FALTAS	TIPO_MEDIA
2	username	Programacao	A	Professor	5	4	0	2

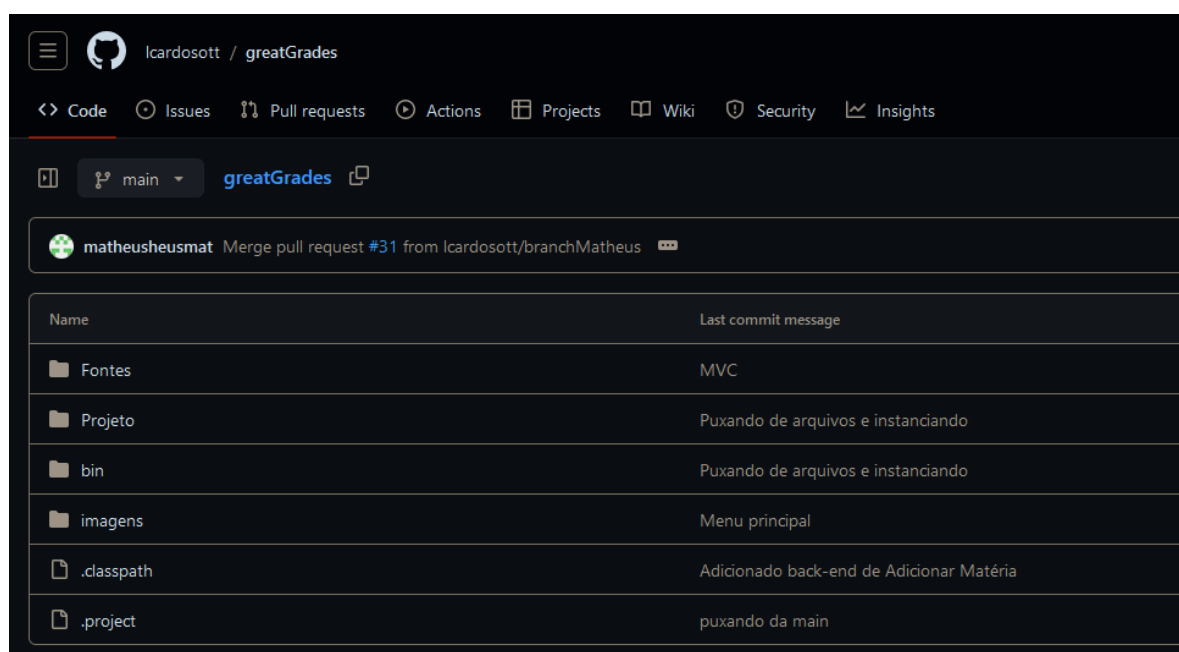
← Avaliacoes.csv								Abrir com ▾
	A	B	C	D	E	F	G	
1	USER	NOME_MATERIA	NOME_AVALIACAO	DATA_AVALIACAO	NOTA	PESO_NA_MEDIA	EXTRA_INFO	
2	username	Programacao	P1	29/07/2023	8	0.5	Super dificil	

**Figura 3** - Arquivos CSV como banco de dados do aplicativo.

## 4 - Controle de Versão

Para cooperação entre os membros do grupo, que trabalharam em conjunto para a construção do projeto, foi necessária uma ferramenta para compartilhamento de código e controle de versões. Utilizamos o GitHub para isso, e cada um trabalhou em sua *branch*. Toda vez que algum membro quisesse alterar o código ou adicionar mais funcionalidades, bastava realizar o comando *git pull* para copiar o que está na *branch* principal (*main*).

Feitas as atualizações, o programador realizava *git add* e *git commit* para salvá-las. Então, bastava realizar o comando *git push origin <nomeDaBranch>* para enviar ao repositório remoto. Após isso, para que esta versão torna-se a principal, foram feitos *merges* da *branch* do programador para a *main*.



The screenshot shows the GitHub interface for the repository 'greatGrades' by user 'Icardosott'. The 'main' branch is selected. A merge pull request #31 from 'Icardosott/branchMatheus' is shown. Below, a table lists the files and folders in the repository with their last commit messages.

Name	Last commit message
Fontes	MVC
Projeto	Puxando de arquivos e instanciando
bin	Puxando de arquivos e instanciando
imagens	Menu principal
.classpath	Adicionado back-end de Adicionar Matéria
.project	puxando da main

**Figura 4** - Organização do repositório do projeto no GitHub

## 5 - Considerações e conclusão

O projeto teve seus desafios de realização. Os principais problemas enfrentados pelo grupo dizem respeito ao controle de versão e utilização da interface gráfica.

Sobre o controle de versão, o uso da plataforma GitHub foi uma novidade para todos os membros e, em um primeiro momento, pareceu demasiadamente complicada. Porém, com a prática, o controle baseado em Git torna-se uma ferramenta extremamente útil para programadores que trabalham em conjunto. Durante a realização do projeto, nos deparamos com diversos casos complicados, principalmente quando duas pessoas trabalhavam ao mesmo tempo, o que levava a tempo perdido. Acreditamos que um aprofundamento nessa ferramenta,

conhecendo o seu potencial e suas funcionalidades, facilitará muito em futuros projetos.

A respeito da utilização da interface gráfica, sendo o Java Swing nossa escolha, aprendemos sua utilização básica, como a criação de frames, panels, labels e buttons. Porém, por se tratar de, novamente, algo novo para nós, a solução de problemas como a integração front-end e back-end, visibilidade de componentes na tela e atualização por ações do usuário demonstrou que, apesar de coerente, a biblioteca Swing é considerada, segundo um aforismo em inglês: *easy to learn, difficult to master* (fácil de aprender, difícil de dominar).

Sendo assim, em conclusão, levando em consideração o tempo destinado ao desenvolvimento do aplicativo (~3 semanas) e certa defasagem teórica para construí-lo, 'GreatGrades' funciona satisfatoriamente, e ainda tem potencial para ser, futuramente, mais desenvolvido, otimizado e ampliado.