Daniel Schreiber Guimarães

Sistema de recomendação de disciplinas para matrícula de um aluno da PUC-Rio

PROJETO FINAL

DEPARTAMENTO DE INFORMÁTICA

Programa de Graduação em Engenharia da Computação

Rio de Janeiro
Novembro de 2023

Pontifícia Universidade Católica
do Rio de Janeiro

Daniel Schreiber Guimarães

Sistema de recomendação de disciplinas para matrícula de um aluno da PUC-Rio

Relatório de Projeto Final I

Relatório de Projeto Final, apresentado ao Programa de Engenharia da Computação, do Departamento de Informática da PUC-Rio como requisito parcial para a obtenção do titulo de Bacharel em Engenharia da Computação.

Orientador: Prof. Marcos Vianna Villas

Rio de Janeiro
Novembro de 2023

Daniel Schreiber Guimarães

Graduando em Engenharia da Computação na PUC - Rio

Ficha Catalográfica

Guimarães, Daniel Schreiber

Sistema de recomendação de disciplinas para matrícula de um aluno da PUC-Rio / Daniel Schreiber Guimarães; orientador: Marcos Vianna Villas. – 2023.

61 f: il. color. ; 30 cm

Projeto Final - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2023.

Inclui bibliografia

1. Informática – Teses. 2. Recomendação. 3. Disciplinas. I. Villas, Marcos Vianna. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Para os meus pais, por
todo o seu suporte e encorajamento.

Abstract

Guimarães, Daniel Schreiber; Villas, Marcos Vianna (Advisor). **Sistema de recomendação de disciplinas para matrícula de um aluno da PUC-Rio**. Rio de Janeiro, 2023. 61p. Projeto Final – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

This abstract will be filled.

Keywords

Recommendation; Courses.

Resumo

Guimarães, Daniel Schreiber; Villas, Marcos Vianna. **Sistema de recomendação de disciplinas para matrícula de um aluno da PUC-Rio**. Rio de Janeiro, 2023. 61p. Projeto Final – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Esse resumo ainda será preenchido.

Palavras-chave

Recomendação; Disciplinas.

Sumário

Lista de figuras

Lista de tabelas

1
Introdução

A cada semestre, um aluno que está fazendo graduação no Departamento de Informática da PUC-Rio precisa fazer a sua matrícula em disciplinas oferecidas pela universidade para os próximos seis meses. Apesar de algumas restrições como pré-requisitos de disciplinas, o aluno tem ampla liberdade de escolher as disciplinas que melhor se encaixam na sua grade horária. Essa liberdade é uma vantagem devido à disponibilização de uma grade horária flexível para o aluno, porém precisa de um maior esforço de pesquisa e organização deste aluno para que a sua matrícula no próximo período seja feita de forma eficaz segundo critérios do próprio aluno.

A motivação desse projeto é o estudo, planejamento e desenvolvimento de um sistema de recomendação de disciplinas para o próximo período que auxilie o aluno em sua matrícula ao sugerir interativamente disciplinas com base em dados fornecidos pela universidade e por avaliações informadas por alunos.

2
Situação Atual

2.1
Serviços disponibilizados

O microhorario[1] é um serviço disponibilizado pela universidade para consultar informações das disciplinas do semestre atual. Este possui uma interface simples que pode ser visualizada na figura 2.1. Essa interface permite filtrar disciplinas por seus atributos como nome, código, professor e departamento. Porém, essa interface não é personalizada com relação ao aluno, ou seja, não exibe as disciplinas que o aluno em específico ainda não cursou ou que o aluno não pode cursar devido à pré-requisitos.

Figura 2.1: Interface do *microhorario*

Para consultar a grade recomendada e os pré-requisitos, o aluno precisa acessar um documento[2] presente na página do departamento ou acessar o currículo disponibilizado na página da universidade. A grade recomendada possui os nomes das disciplinas e seus pré-requisitos, mas não contém a disponibilidade do próximo período.

Além disso, as disciplinas eletivas que o departamento oferece durante o semestre são anunciadas em diferentes veículos de comunicação, como e-mail, página da coordenação do curso, ou folhetos em corredores do departamento. Não há uma distribuição centralizada das ofertas de disciplinas eletivas,

[1] https://www.puc-rio.br/microhorario

[2] http://www.inf.puc-rio.br/wordpress/wp-content/uploads/2023/01/Grade_Eng_comp_2023.pdf

portanto o aluno pode não saber aonde procurar essas ofertas, e perder boas oportunidades por desconhecer o anúncio da disciplina eletiva.

2.2 Processo de matrícula

A matrícula na PUC-Rio é um processo que dura vários dias. A universidade divulga o agendamento de matrícula, em que cada aluno recebe uma data e hora para realizar sua matrícula. As datas possíveis abrangem um período de cinco dias. Durante o período de matrícula, o microhorario atualiza periodicamente para indicar quais turmas ainda possuem vagas disponíveis, e se houve o cadastro ou cancelamento de alguma outra disciplina durante este período.

Portanto, é comum que um aluno crie formas de planejamento próprias para conseguir organizar todo o fluxo de dependências, anotar as disciplinas que estão sendo oferecidas, assim como suas turmas e professores. Esses dados precisam estar em constante atualização durante o período de matrícula, conforme as vagas vão sendo preenchidas e a disponibilização de novas turmas ou disciplinas são anunciadas. Este é um processo que maximiza a qualidade da sua grade horária, mas demanda muito esforço e tempo.

A universidade disponibiliza um simulador de matrícula por um curto período antes do processo de matrícula em si. A interface do simulador é a mesma da matrícula, como pode ser observado na figura 2.2, afim de o aluno poder simular a criação da sua grade de disciplinas para o próximo semestre. O simulador apresenta os dados das disciplinas atualizados conforme o microhorario, e os disponibiliza de três formas diferentes: buscas por disciplinas que faltam cursar no currículo do aluno, buscar por nomes de disciplinas e buscar por horários das turmas das disciplinas.[1]

2.3 Estudos relacionados

A escolha de disciplinas é um problema comum em universidades que possuem algum grau de flexibilidade na grade horária. Ng & Linn[2] desenvolveram um sistema de recomendação de disciplinas para a sua universidade chamado CrsRecs que utiliza análise de sentimento, pontuações de professores e disciplinas, e preferências que o aluno escolhe fornecer. A vantagem do CrsRecs é não depender de dados privados da universidade como notas e avaliação dos alunos nas disciplinas, utilizando somente informações fornecidas diretamente pelo usuário. Mas há também um esforço na decodificação das informações for-

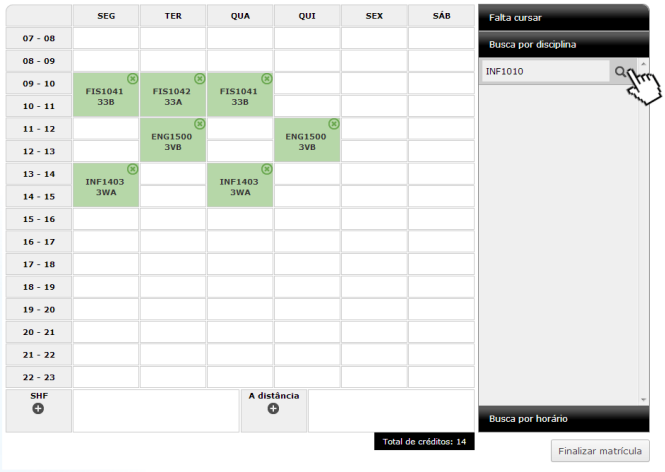

Figura 2.2: Interface do simulador

necidas, o que pode ocasionar recomendações incorretas provenientes da má interpretação do usuário.

Há estudos que utilizam históricos escolares dos alunos para gerar modelos de previsão de notas.[3, 4, 5] Nesse caso, o objetivo do modelo é tentar prever quais disciplinas o aluno tem a maior chance de obter uma boa nota, mas não necessariamente prever quais delas mais combina com o aluno.[3, 4]

A escolha de disciplinas eletivas também é uma preocupação presente nas universidades. Na PUC-Rio e em outras universidades, os currículos de graduação solicitam créditos de disciplinas eletivas, seja dentro do departamento da graduação do aluno ou não. O sistema do estudo de Adak et al.[5] recomenda disciplinas dentro de um departamento que parecem se relacionar com o histórico escolar do aluno, mas excluem disciplinas fora do departamento. Já o sistema de Xu et al.[6] recomenda uma sequência de disciplinas eletivas que maximem a sua nota e não aumentem o tempo de conclusão da graduação do aluno, levando em conta dados históricos do aluno e a disponibilidade atual das disciplinas. Por último, o estudo de Adak & Ercan[7] utiliza dois algoritmos de inteligência artificial de aprendizado supervisionado, support vector machine (SVM) e árvores de decisão, para recomendar eletivas que mais se assemelham ao aluno, utilizando dados históricos do aluno tanto para o treinamento dos modelos de algoritmo como para a recomendação.

O comum nos estudos é a tentativa de maximizar a nota média do aluno utilizando os dados históricos de alunos da universidade para treinar modelos de inteligência artificial.[3, 4, 5, 7] Porém, o foco específico nos resultados pode ocultar oportunidades de disciplinas que o aluno poderia se interessar.

3
Objetivo

O sistema desenvolvido nesse trabalho consiste em uma interface de planejamento de matrícula semelhante ao simulador de matrícula integrado com um algoritmo de recomendação. O sistema disponibiliza as disciplinas do próximo período conforme os dados mais atuais do microhorario. Para que haja uma personalização na recomendação das disciplinas, o sistema permite que o aluno carregue o histórico escolar na universidade afim de fornecer o histórico das disciplinas e seus graus para o algoritmo de recomendação.

O algoritmo de recomendação do sistema recebe as disciplinas oferecidas no próximo período e seus pré-requisitos, o modelo de grade recomendada pelo departamento e o histórico do aluno, e então responde com disciplinas selecionadas. O algoritmo também pode ser personalizado com informações de preferência do usuário.

O sistema armazena o histórico fornecido pelo aluno sem seus dados pessoais, afim de montar uma base de dados contendo exemplos de grades de alunos e suas notas, para que o algoritmo a utilize para obter resultados mais satisfatórios para o aluno usuário do sistema.

O sistema e o algoritmo foi desenvolvido especificamente para alunos do departamento de informática da PUC-Rio devido a dificuldade de abranger todas os currículos nos diferentes departamentos da universidade. Essa restrição também permitiu restrigir o escopo do projeto afim de tentar obter melhores resultados.

4
Plano de Ação

Para modelar o sistema, foi realizada uma validação do problema. Alunos do departamento de informática da universidade foram entrevistados afim de descobrir quais dificuldades enfrentam durante o processo de matrícula e quais são as suas preferências ao montar uma grade disciplinar. O objetivo era identificar quais características de uma grade disciplinar contribuem para a satisfação do aluno e adicionar as funcionalidades necessárias no sistema e no algoritmo.

Com o problema validado, então o projeto passou por uma etapa de pesquisa. Foram estudados diferentes algoritmos de recomendação em contextos semelhantes ao sistema a ser desenvolvido, afim de se projetar o algoritmo que mais se adequa às necessidades do problema validado. Nessa etapa, também foram analisadas as fontes de informações disponibilizadas pela universidade e como cada fonte pode ser integrada no sistema.

Após a etapa de pesquisa, o algoritmo começou a ser desenvolvido. Depois de obter um algoritmo minimamente viável, foi desenvolvida a interface de planejamento para mostrar o funcionamento do algoritmo. Tanto o algoritmo com a interface foram desenvolvidas utilizando um modelo de desenvolvimento incremental, para que o sistema passe por etapas de desenvolvimento, teste e validação com os alunos.

A figura 4.1 exemplifica o fluxo das etapas efetuadas, desde os estudos preliminares até os testes e trabalhos futuros.

4.1
Cronograma original

O cronograma inicial pode ser visualizado na figura 4.2.

4.2
Cronograma revisado

==== A FAZER A FAZER A FAZER ====

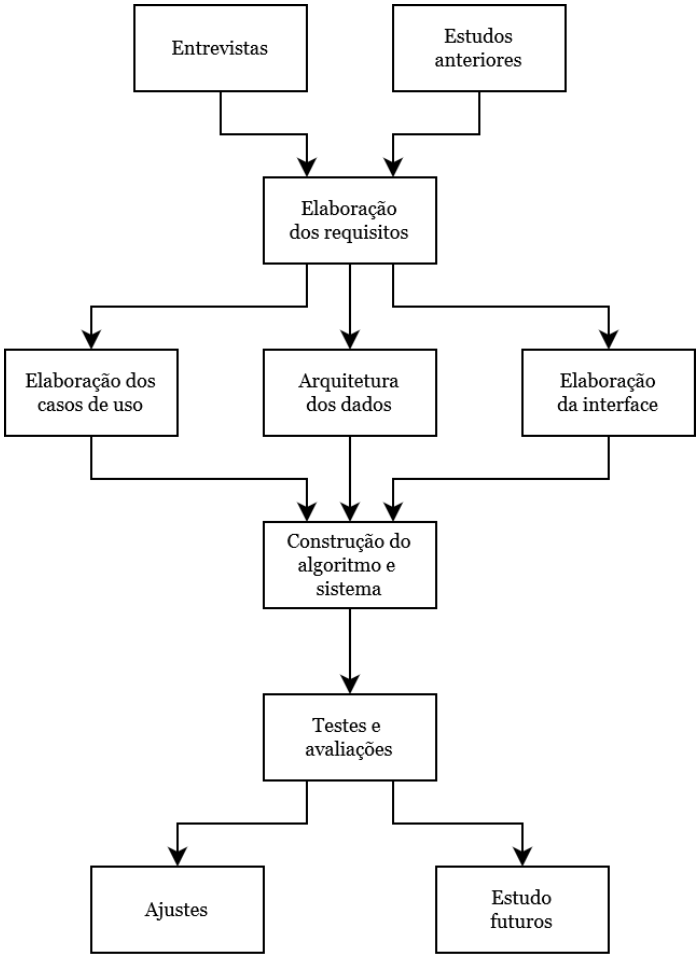

Figura 4.1: Fluxo das etapas realizadas

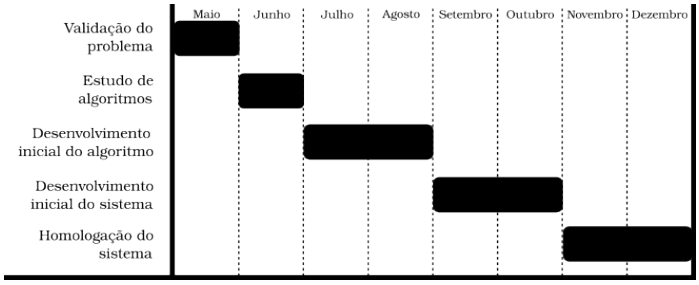

Figura 4.2: Cronograma original do projeto

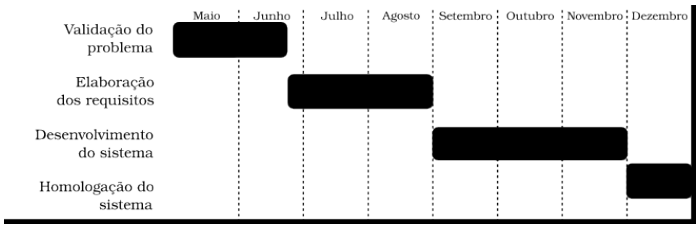

Figura 4.3: Cronograma atualizado do projeto

5
Validação do problema

A validação do problema foi efetuada por meio de entrevistas com alunos de graduação do departamento de informática. As entrevistas tinham como objetivo obter as opiniões de diferentes alunos em diferentes períodos de suas graduações quanto as suas preferências ao realizar as suas matrículas. As opiniões são essenciais para planejar um algoritmo de recomendação que abrange as preferênciasdo máximo de estudantes.

5.1
Método das entrevistas

Os entrevistados são alunos de graduação dos cursos do departamento de informática da PUC-Rio, cursando os cursos de engenharia da computação ou de ciência da computação. Foram selecionados vários alunos em diferentes estágios da graduação, do segundo ao sexto ano de graduação. Como algumas perguntas se referem às escolhas de disciplinas eletivas, e essas só começam a ser escolhidas no meio do período de graduação, não foram escolhidos estudantes no primeiro ano de graduação.

5.2
Perguntas realizadas

A seguir estão as seis perguntas realizadas para cada entrevistado do departamento de informática, assim como uma explicação do motivo da pergunta ser realizada.

1. **Qual é o seu curso e período atual?** O período atual do entrevistado é utilizado para entender o contexto de suas respostas, pois é necessário observar se o período em que o graduando está altera sua opinião ao realizar a sua matrícula. Como a quantidade de períodos propostos para a conclusão do curso variam de curso para curso no departamento de informática, também foi solicitado ao entrevistado seu curso para entender seu progresso de estudo na universidade e entender ainda melhor o contexto das suas próximas respostas.

2. **Em média quantos créditos você faz num semestre?** A quantidade de créditos por semestre é relevante para que o algoritmo de recomendação tenha um conhecimento da média de disciplinas selecionadas por período. Essa média pode ser usada pelo algoritmo para saber quantas

disciplinas o estudante ainda pretende fazer naquele semestre e recomendar de acordo.

3. **Você se prepara com antecedência para sua matrícula? Se sim, com quanta antecedência? Se não, por quê?** Essa pergunta é fundamental para planejar uma expectativa de utilização da ferramenta de recomendação de disciplinas, pois a ferramenta com o algoritmo estaria disponibilizada em um sistema cujo obtivo é planejar a sua matrícula.

4. **Que critérios você utiliza para escolher uma disciplina? Dados os critérios fornecidos, coloque-os em uma ordem de mais relevante para menos relevante.** Essa é a pergunta mais fundamental para o planejamento do algoritmo. Afinal, o objetivo do algoritmo de recomendação é recomendar disciplinas relevantes para um aluno. Por isso, é necessário saber o que torna uma disciplina relevante para ser escolhida.

5. **Como você procura disciplinas eletivas?** O objetivo dessa pergunta é descobrir como cada estudante procura as disciplinas eletivas, para saber quais são as principais fontes de informações utilizadas.

6. **Você, pessoalmente, prefere disciplinas mais fáceis, mas que talvez não sejam muito úteis para a sua carreira, ou disciplinas mais relevantes mas que possam ser mais difíceis?** Essa pergunta pretende entender qual das duas opções é mais frequente. Caso a maioria dos estudantes preferem disciplinas mais fáceis, o algoritmo irá depender mais das notas dos alunos, e não tanto do conteúdo da disciplina em si. Caso a maioria dos estudantes preferem disciplinas mais relevantes, por exemplo.

Ao final da entrevista, também foi oferecida a oportunidade de fornecer alguma opinião sobre o sistema de recomendação a ser desenvolvido.

5.3 Resultados das entrevistas

5.3.1
Dados básicos dos estudantes (Perguntas 1 e 2)

Dos onze entrevistados, cinco eram alunos de ciência de computação e os outros seis eram de engenharia de computação. 5 dos 6 alunos entrevistados de engenharia de computação já haviam cursado mais da metade do curso (já cursou 5 dos 10 semestres recomendados), e 3 dos 5 alunos entrevistados de ciência de computação já haviam cursado mais da metade do curso (já cursou 4 dos 8 semestres recomendados). Além disso, dos onze alunos, nove eram homens e dois eram mulheres.

A distribuição dos períodos atuais dos entrevistados pode ser visualizada na figura 5.1.

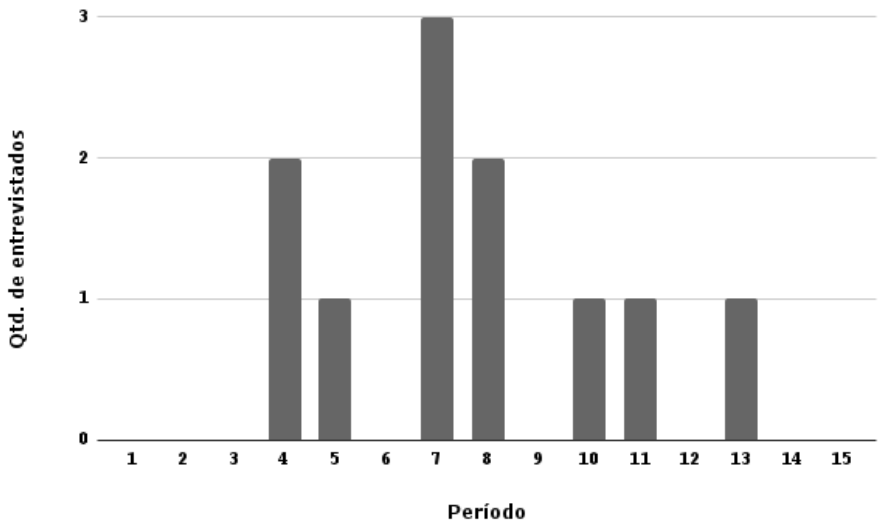

Figura 5.1: Distribuição da quantidade de entrevistados pelos seus períodos atuais.

A quantidade média de créditos por semestre dos alunos entrevistados é de 24 créditos, semelhante à quantidade média de créditos por semestre no currículo recomendado de ambos os cursos.

5.3.2
Escolha de disciplinas (Perguntas 3 e 4)

Dez dos onze entrevistados se preparam com antecedência para a matrícula. Três desses utilizam como base o repositório de disciplinas microhorário para buscar as informações das disciplinas oferecidas para o próximo período, e também utilizam o microhorário para descobrir quais são as disciplinas eletivas disponibilizadas no período. Todos os dez entrevistados utilizam o simulador de matrícula para confirmar suas escolhas. Apenas um dos entrevistados não se

prepara com antecedência, fazendo suas pesquisas e escolhas durante o período da própria matrícula.

Os critérios preferidos de cada aluno variam bastante. Por exemplo, um entrevisto citou que alguns critérios são *"a matéria em si, se ela é mais difícil de entender ou mais fácil, e também o método de avaliação, ou seja, se a nota final é composta de somente duas provas, ou se são vários trabalhos pequenos espalhados pelo semestre. Além disso, eu diria que o horário também é um limitador, porque não consigo ter aula sete da manhã todos os dias. Mas eu acho que o mais importante para mim é o professor: Se eu sei quem é o professor, sei que ele explica bem, vale a pena mesmo que a matéria seja mais difícil. Eu diria que a ordem dos critérios seria o professor, depois a matéria em si, depois o método de avaliação e por último o horário da disciplina."*.

Como pode ser observado na resposta fornecida acima, as respostas podem variar bastante de aluno para aluno. Por isso, os critérios de escolha de disciplinas ditos nas entrevistas foram agrupados em cinco grupos: (1) Conteúdo da disciplina; (2) Professor; (3) Método de avaliação; (4) Horário da disciplina e (5) Opinião de amigos. A tabela 5.1 com as respostas dos onze entrevistados e as suas respostas. O número indica em qual posição ficou aquele critério na ordem de preferência pessoal do aluno.

	Conteúdo	Professor	Avaliação	Horário	Opinião
Entrevistado 1	3	-	-	1	2
Entrevistado 2	1	-	-	3	2
Entrevistado 3	2	3	-	-	1
Entrevistado 4	3	1	-	2	-
Entrevistado 5	2	-	-	1	3
Entrevistado 6	1	-	-	-	-
Entrevistado 7	1	-	-	-	-
Entrevistado 8	2	3	1	4	-
Entrevistado 9	3	2	-	-	1
Entrevistado 10	1	-	3	2	-
Entrevistado 11	-	2	-	1	-

Tabela 5.1: Graus dos critérios de escolhas de disciplinas escolhidos pelos entrevistados

Para gerar uma comparação dos critérios, foi dado um peso para cada grau de importância de acordo com a tabela 5.2.

Grau	1	2	3	4	5	-
Peso	5	4	3	2	1	0

Tabela 5.2: Relação de peso para grau dos critérios de escolhas

Ao substituir os pesos da Tabela 5.2 na Tabela 5.1, é possível somar os pesos para cada critério de escolha e obter uma relação entre eles, conforme a tabela 5.3.

	Conteúdo	Professor	Avaliação	Horário	Opinião
Entrevistado 1	3	0	0	5	4
Entrevistado 2	5	0	0	3	4
Entrevistado 3	4	3	0	0	5
Entrevistado 4	3	5	0	4	0
Entrevistado 5	4	0	0	5	3
Entrevistado 6	5	0	0	0	0
Entrevistado 7	5	0	0	0	0
Entrevistado 8	4	3	5	2	0
Entrevistado 9	3	4	0	0	5
Entrevistado 10	5	0	3	4	0
Entrevistado 11	0	4	0	5	0
Soma	41	19	8	28	21

Tabela 5.3: Pesos dos critérios de escolhas de disciplinas escolhidos pelos entrevistados.

Observando a Tabela 5.3, as somas dos pesos indicam que a ordem dos critérios mais relevantes na escolha de uma disciplina, do mais relevante para o menos relevante: *Conteúdo da disciplina*, *Horário da disciplina*, *Opinião de amigos*, *Professor* e *Método de avaliação*.

5.3.3 Disciplinas eletivas (Perguntas 5 e 6)

Dos onze entrevistados, nove citaram a utilização do microhorário para coletar as disciplinas eletivas do período. Os outros dois dependem mais de recomendação de amigos. Três entrevistados procuram por uma listagem oficial

de disciplinas eletivas oferecidas pelo departamento para o próximo período, mas nem sempre encontram essa listagem oficial.

Por último, quatro entrevistados preferem disciplinas eletivas mais fáceis, enquanto três entrevistados preferem disciplinas eletivas que sejam mais relevantes para a sua formação. Os quatro restantes preferem um equilíbrio de facilidade e relevância. Dois entrevistados disseram que inicialmente escolhem disciplinas mais relevantes, mas que ao decorrer dos períodos optam por eletivas mais fáceis. Uma das respostas foi que *"[Eu] prefiro [disciplinas] eletivas mais fáceis, porque eu as vejo como uma oportunidade para facilitar a minha vida na faculdade, pois já tenho outras disciplinas mais difíceis para me preocupar. Eu encaro as [disciplinas] eletivas como um escape para poder respirar."*

6
Requisitos

Os requisitos estão divididos em requisitos relacionados ao sistema e requisitos relacionados ao algoritmo. Os requisitos do sistema dizem respeito ao sistema que disponibiliza o serviço de montagem de grade horária do próximo período, assim como um serviço de avaliar disciplinas e professores, que existe para satisfazer as necessidades observadas nos capítulos anteriores. Os requisitos do algoritmo dizem respeito às entradas e saídas do algoritmo afim de se produzir uma recomendação adequada de acordo as necessidades observadas no capítulo 5.

6.1
Do algoritmo

Os requisitos do algoritmo de recomendação de disciplinas precisam satisfazer os critérios de escolha de disciplinas mais relevantes, discutidos no capítulo 5.3.2, ou seja, recomendar de acordo com o conteúdo da disciplina, horários disponíveis, opiniões de amigos, o professor, e por último o método de avaliação. Além disso, para satisfazer a preferência de disciplinas eletivas discutidas no capítulo 5.3.3, os requisitos precisam satisfazer a preferência de recomendar disciplinas com base na sua facilidade.

A lista de requisitos do algoritmo está disponível na tabela 6.1.

6.2
Do sistema

O sistema precisa satisfazer as necessidades do usuário e também as necessidades do algoritmo, pois o sistema hospeda o algoritmo de recomendação. Os requisitos do sistema estão disponíveis na tabela 6.2.

RF01	O algoritmo deve receber as disciplinas e turmas oferecidas no próximo semestre conforme o microhorário.
RF02	O algoritmo deve receber opcionalmente a grade curricular do curso que o aluno está cursando.
RF03	O algoritmo deve receber as avaliações das disciplinas e professores fornecidas pelos usuários do sistema.
RF04	O algoritmo deve receber opcionalmente o histórico escolar do aluno.
RF05	O algoritmo deve receber disciplinas já selecionadas pelo aluno.
RF06	O algoritmo deve retornar recomendações de disciplinas com base nas entradas fornecidas.
RNF1	O algoritmo deve retornar as recomendações em menos de 3 segundos.
RNF2	O algoritmo deve ser determinístico, ou seja, retorna as mesmas recomendações para as mesmas entradas.

Tabela 6.1: Requisitos do algoritmo

RF07	O sistema deve permitir que o usuário crie uma grade horária para o próximo período.
RF08	O sistema deve permitir que o usuário submita seu histórico escolar.
RF09	O sistema deve permitir que o usuário selecione turmas das disciplinas para compor sua grade horária.
RF10	O sistema deve permitir que o usuário armazene a sua grade horária finalizada.
RF11	O sistema deve permitir que o usuário compartilhe a sua grade horária finalizada.
RF12	O sistema deve permitir que o usuário recupere uma grade horária montada a partir de um link compartilhado.
RF13	O sistema deve permitir que o usuário inicie uma sessão autenticada utilizando sua matrícula.
RF14	O sistema deve permitir que o usuário finalize uma sessão autenticada.
RF15	O sistema deve permitir que o usuário avalie uma disciplina.
RF16	O sistema deve permitir que o usuário avalie um professor de uma disciplina.
RF17	O sistema deve permitir que o usuário altere uma avaliação feita anteriormente.
RNF3	O sistema deve suportar pelo menos 40 usuários simultâneos.

Tabela 6.2: Requisitos do sistema

7
Casos de Uso

Os casos de uso do sistema servem para descrever como o sistema pode ser utilizado afim de se satisfazer as necessidades do usuário. Os casos de uso referenciam os requisitos do sistema descritos no capítulo 6. A figura 7.1 contém o diagrama de casos de uso do sistema.

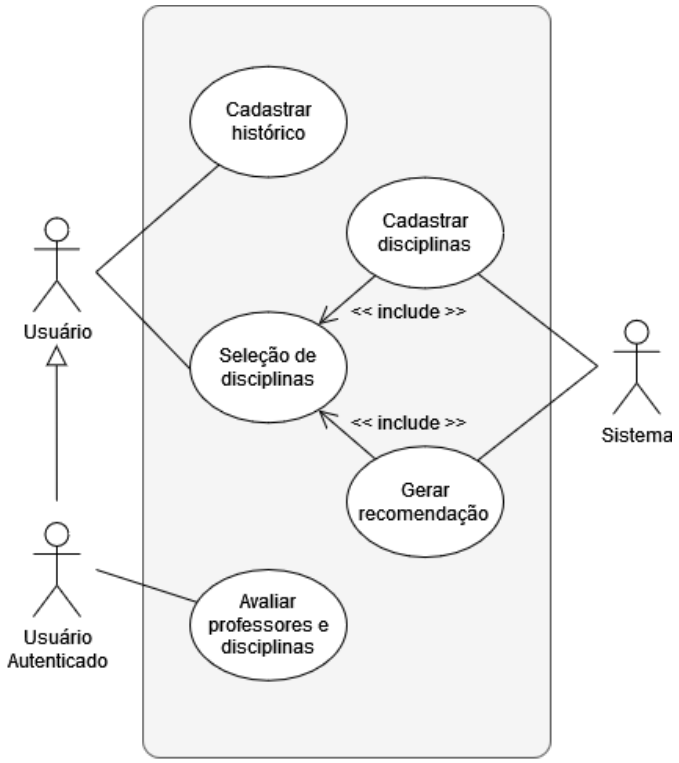

Figura 7.1: Diagrama dos casos de uso

A seguir estão as descrições dos casos de uso presentes na figura 7.1.

Caso de Uso **UC01** - Cadastrar disciplinas	
Objetivo	Permitir que o sistema atualize as disciplinas no seu banco de dados, incluindo as eletivas.
Requisitos	RF01 e RF09
Atores	Sistema
Pré condições	Não se aplica

Caso de Uso **UC01** (continuação)	
Fluxo principal	[1] O sistema acessa o microhorário, coletando informações de todas as disciplinas.
	[2] O sistema trata e converte as informações para o modelo do banco.[**A1**]
	[3] O sistema remove as informações anteriores do banco.
	[4] O sistema armazena as novas informações no banco.
	[5] O sistema atualiza a data e hora da última atualização do microhorário que aparece na interface.
	[6] O caso de uso é encerrado.

Tabela 7.1: Caso de uso UC01

Caso de Uso **UC02** - Cadastrar histórico	
Objetivo	Permitir que o usuário cadastre seu histórico escolar no sistema para personalizar as recomendações.
Requisitos	RF02, RF04, RF08
Atores	Usuário
Pré condições	O usuário seleciona a opção de cadastrar histórico.
Fluxo principal	[1] O sistema exibe uma tela sobreposta solicitando o histórico escolar do aluno.
	[2] O usuário submete o histórico escolar. [**A1**]
	[3] O sistema armazena o histórico e o currículo do aluno.
	[4] O sistema exibe um texto de histórico cadastrado com sucesso. [**A2**]
	[5] O caso de uso é encerrado.

Caso de Uso **UC02** (continuação)	
Fluxos Alternativos	**[A1] O usuário pressiona fora da tela de cadastro de histórico**
	[1] O caso de uso é encerrado.
	[A2] O sistema não consegue encontrar o currículo ou as disciplinas cursadas no currículo
	[1] O sistema exibe um texto de erro.
	[2] O sistema volta para o passo 1 do fluxo principal.

Tabela 7.2: Caso de uso UC02

Caso de Uso **UC03** - Gerar recomendações	
Objetivo	Permitir que o sistema recomende disciplinas para o usuário.
Requisitos	RF01-06, RNF01-02
Atores	Sistema
Pré condições	O usuário modificou a grade horária.
Fluxo principal	[1] O sistema coleta as disciplinas e turmas já adicionadas no grade horária do usuário.
	[2] O sistema recupera o histórico escolar do usuário do banco de dados, caso o usuário esteja autenticado e o tenha submetido.[**A1**]
	[3] O sistema utiliza o algoritmo para gerar recomendações de disciplinas para o usuário.
	[4] O sistema exibe as recomendações na interface do usuário
	[5] O caso de uso é encerrado.

Tabela 7.3: Caso de uso UC03

Caso de Uso **UC04** - Selecionar disciplina	
Objetivo	Permitir que o usuário adicione e remova disciplinas da sua grade.
Requisitos	RF1
Atores	Usuário
Pré condições	O usuário está na área de criação da grade horária.
Fluxo principal	[1] O sistema exibe a grade horária do usuário, uma lista de disciplinas disponíveis, uma lista de disciplinas recomendadas e um campo de texto para pesquisa.
	[2] O usuário seleciona uma disciplina da lista de disciplinas disponíveis ou recomendadas. [**A1**] [**A2**] .
	[3] O sistema exibe as turmas disponíveis para a disciplina selecionada e um botão de voltar.
	[4] O usuário seleciona uma das turmas exibidas. [**A3**]
	[5] O sistema acrescenta a turma selecionada na grade horária, e recalcula as disciplinas recomendadas.
	[6] O caso de uso é encerrado.
Fluxos Alternativos	[**A1**] **O sistema seleciona uma das disciplinas na sua grade horária**
	[1] O sistema exibe um botão de informação e um botão de excluir.
	[2] O usuário seleciona o botão de excluir.[**A3**]
	[3] O sistema remove a turma e disciplina da grade horária do usuário, e recalcula as disciplinas recomendadas.
	[4] O sistema volta para o passo 1 do fluxo principal.
	[**A2**] **O usuário preenche o campo de texto para pesquisa**

Caso de Uso **UC04** (continuação)	
	[1] O sistema altera a lista de disciplinas, filtrando de acordo com o texto do campo de pesquisa.
	[2] O sistema volta para o passo 1 do fluxo principal.
	[A3] O usuário pressiona o botão de informação
	[1] O sistema volta para o passo 3 do fluxo principal.

Tabela 7.4: Caso de uso UC04

Caso de Uso **UC05** - Avaliar disciplinas e professores	
Objetivo	Permitir que o usuário avalie uma disciplina ou um professor.
Requisitos	RF03, RF15 e RF16
Atores	Usuário
Pré condições	Não se aplica.
Fluxo principal	[1] O sistema exibe uma lista de disciplinas e professores e um campo de texto para pesquisa.
	[2] O usuário seleciona uma disciplina. [**A1**] [**A2**]
	[3] O sistema exibe uma tela para avaliar a disciplina, um botão de salvar e um botão de voltar.
	[4] O usuário avalia a disciplina e seleciona o botão de salvar. [**A3**]
	[5] O sistema armazena a avaliação do usuário.
	[6] O caso de uso é encerrado.
Fluxos Alternativos	**[A1] O usuário seleciona um professor**
	[1] O sistema exibe uma tela para avaliar o professor, um botão de salvar e um botão de voltar.

Caso de Uso **UC05** (continuação)	
	[2] O usuário avalia o professor e seleciona o botão de salvar. [**A3**]
	[3] O sistema armazena a avaliação do usuário.
	[4] O caso de uso é encerrado.
	[A2] O usuário preenche o campo de texto para pesquisa
	[1] O sistema exibe disciplinas e professores utilizando como filtro o texto do usuário.
	[2] O sistema volta para o passo 1 do fluxo principal.
	[A2] O usuário seleciona o botão de voltar
	[2] O sistema volta para o passo 1 do fluxo principal.

Tabela 7.5: Caso de uso UC05

Para melhor descrever a interação do usuário com o sistema com a interface da criação de grade horária e com o algoritmo de recomendação, a figura 7.2 apresenta um fluxograma com as possíveis ações do usuário na interface. Neste diagrama, os retângulos representam componentes da interface passíveis de interação do usuário, os hexágonos representam operações efetuadas pelo sistema, e o losango representa uma tomada de decisão pelo sistema. Além disso, as setas contínuas representam operações do usuário, e as setas tracejadas representam transições automáticas entre os componentes e operações.

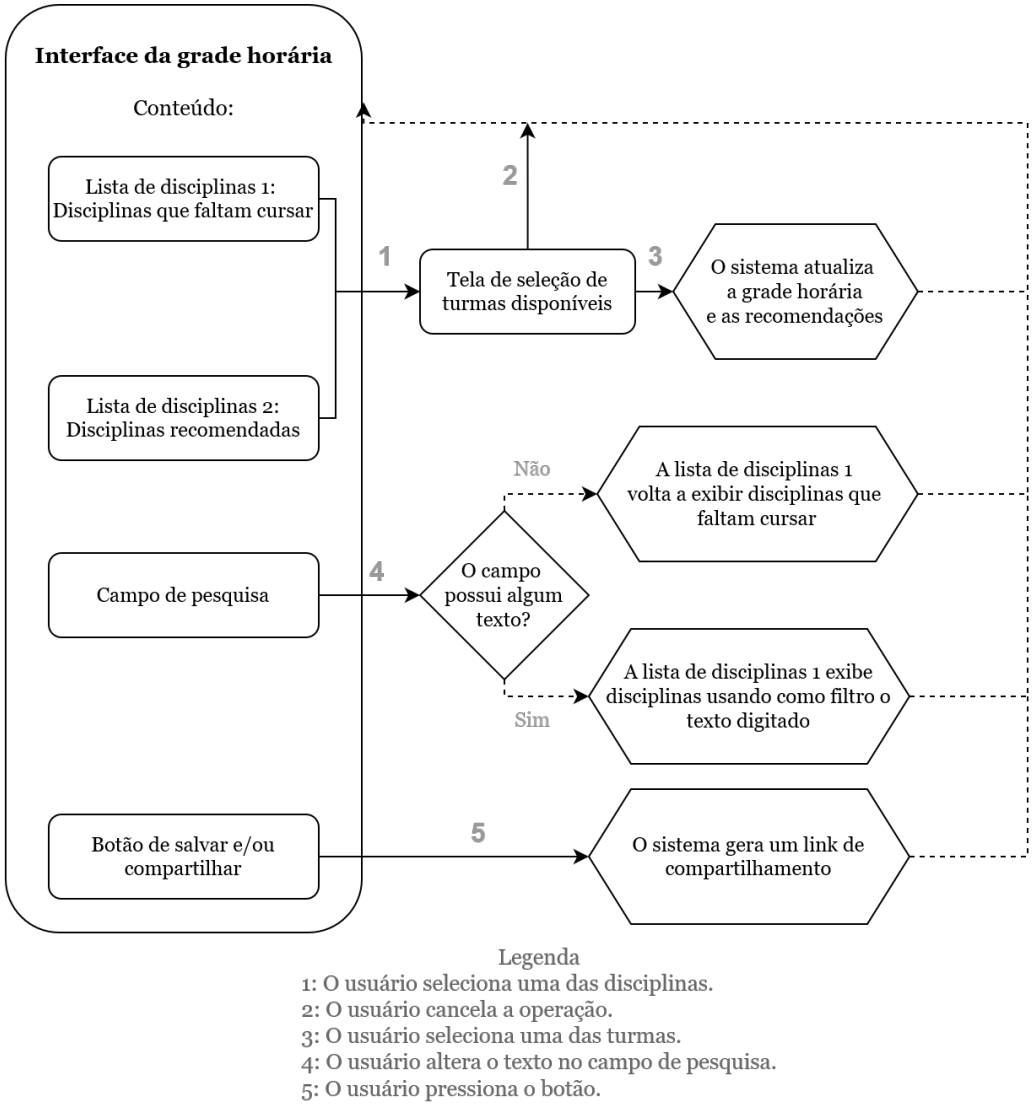

Figura 7.2: Fluxograma da interface de criação de grade horária

8
Wireframe

Um *wireframe* é um protótipo da interface do sistema, que serve para ilustrar o funcionamento e interação do usuário com o sistema. A seguir estão três telas do sistema: a página inicial, a página de criação de grade horária, e a página de avaliação de disciplinas e professores. Essas são as principais páginas do sistema.

O design da página inicial (*Landing Page*), apresentado na figura 8.1, busca ser simples, com uma rápida descrição do sistema e das suas funcionalidades, e com duas opções grandes para enviar o usuário para a página de criação de grade ou de avaliação.

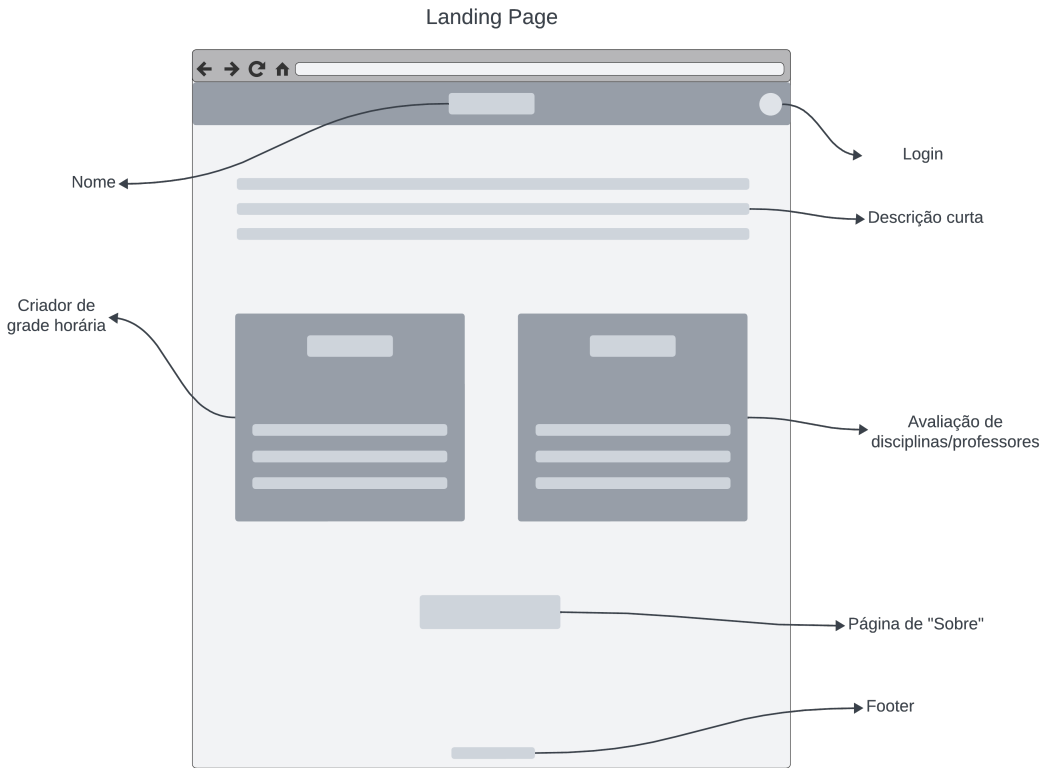

Figura 8.1: Wireframe da página inicial (*Landing Page*)

A página de criação de grade horária, apresentado na figura 8.2, pretende ser semelhante à interface de matrícula real da faculdade, como na figura 2.2, com a adição de mais informações que auxiliem o usuário a fazer suas escolhas, além da barra de disciplinas recomendadas.

Por fim, a página de avaliação de disciplinas e professores, apresentado na figura 8.3, procura ser uma interface simples, com uma lista das disciplinas

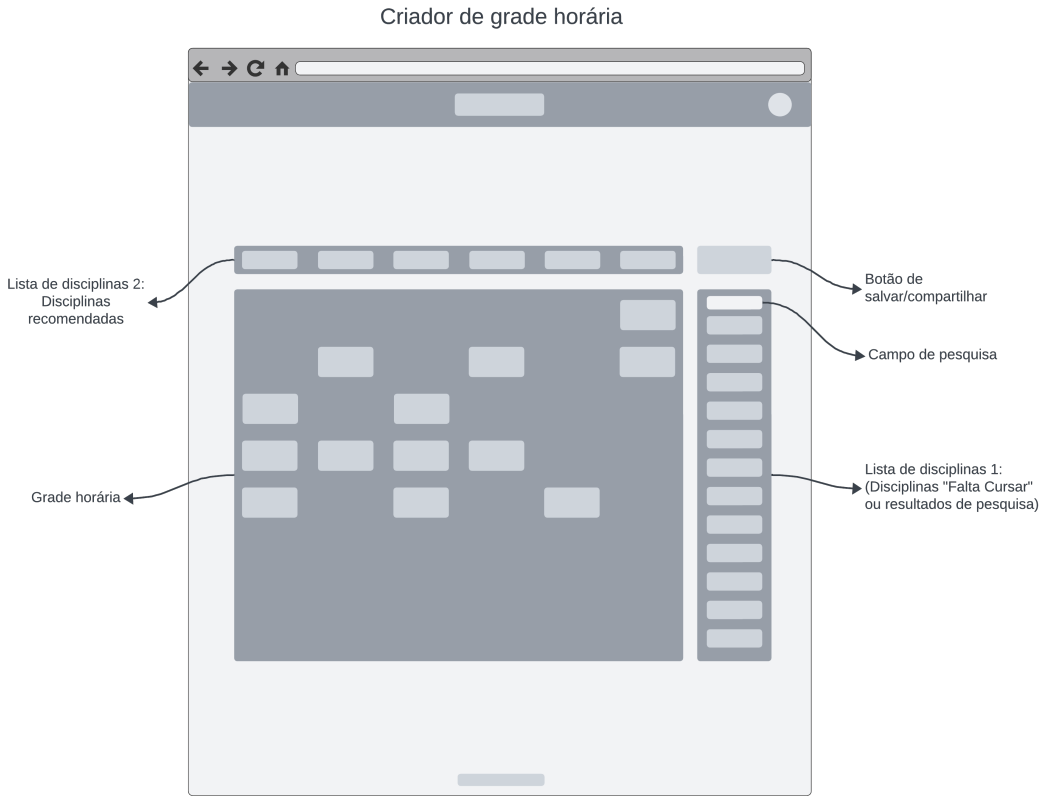

Figura 8.2: Wireframe da página de criação de grade horária

cursadas ou professores, e com a opção de avaliar no formato de estrelas, sendo a pior avaliação uma estrela, e cinco estrelas a melhor.

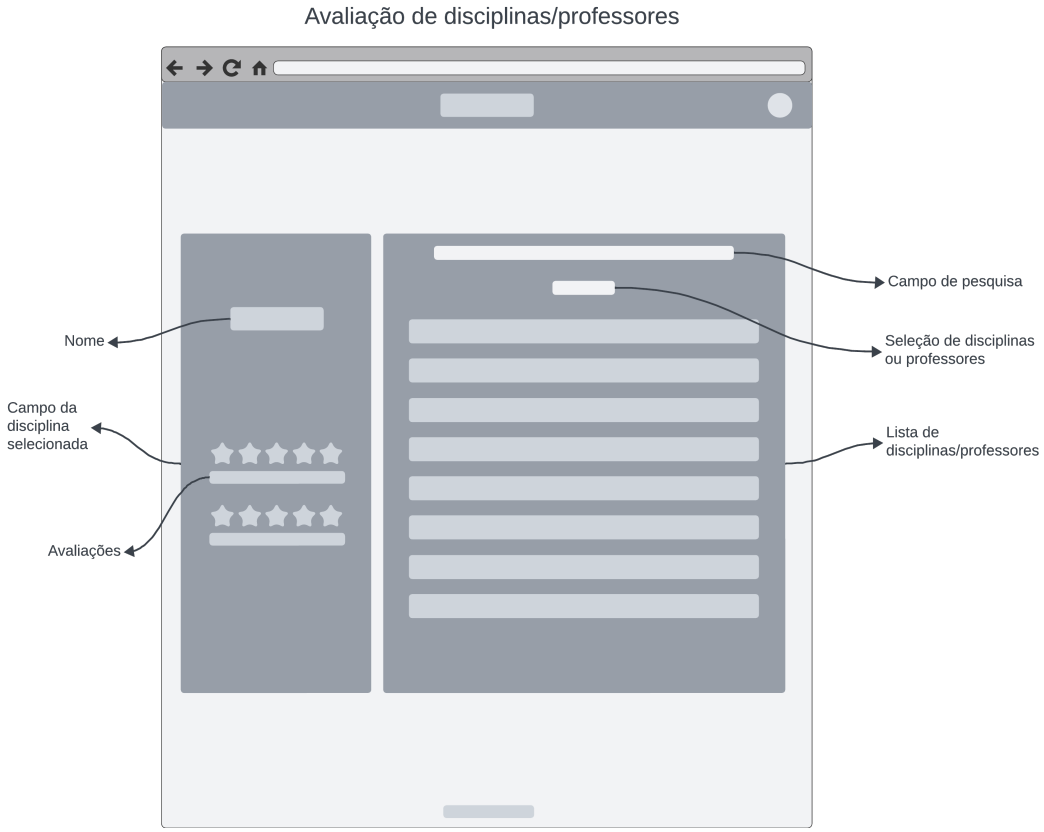

Figura 8.3: Wireframe da página de avaliação de disciplinas e professores

9
Dados do sistema

Para satisfazer os requisitos e os casos de uso, foi necessário modelar os dados disponíveis ao sistema e ao algoritmo.

9.1
Diagrama de entidades-relacionamento

A imagem 9.1 exibe o diagrama de entidades-relacionamento (ER), que descreve os dados no modelos de entidades (coisas de eresse) e seus relacionamentos. Este modelo segue a notação do Peter Chen [8].

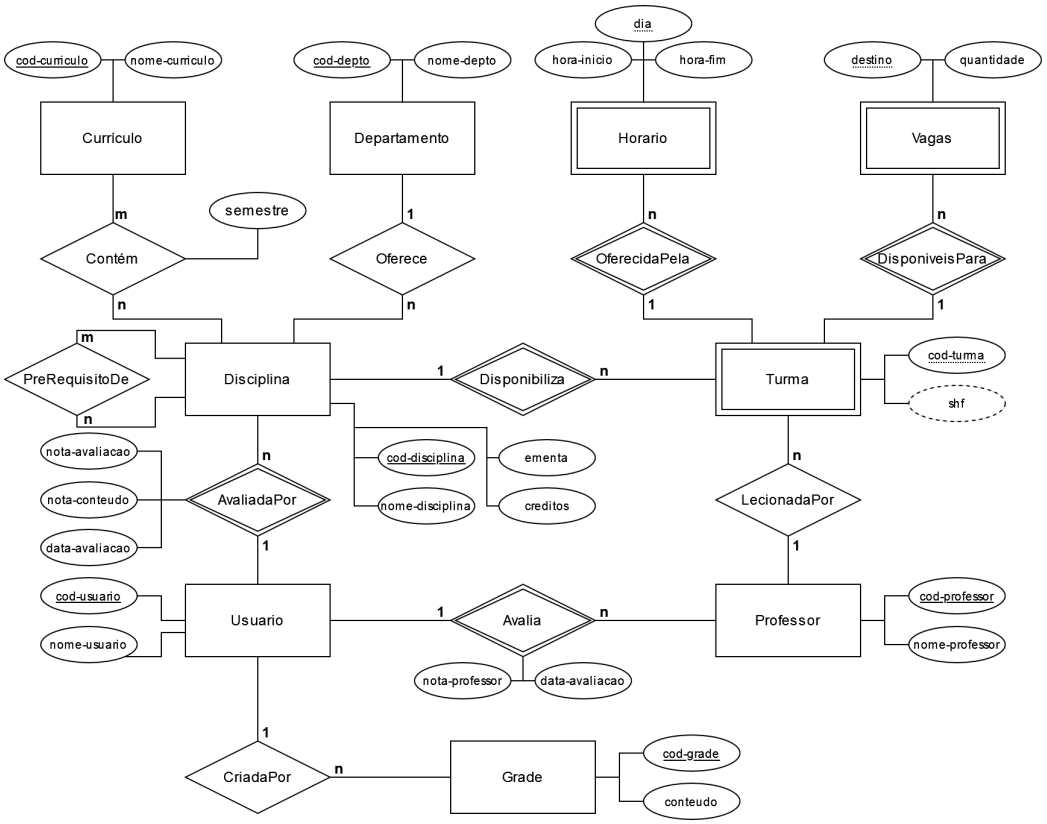

Figura 9.1: Diagrama do modelo entidade-relacionamento

9.2
Modelo lógico

A imagem 9.2 exibe o diagrama lógico dos dados com base no modelo Entidade-Relacionamento. Ele representa a estrutura implementada no banco de dados, com suas tabelas e chaves primárias (PK) e chaves estrangeiras (FK).

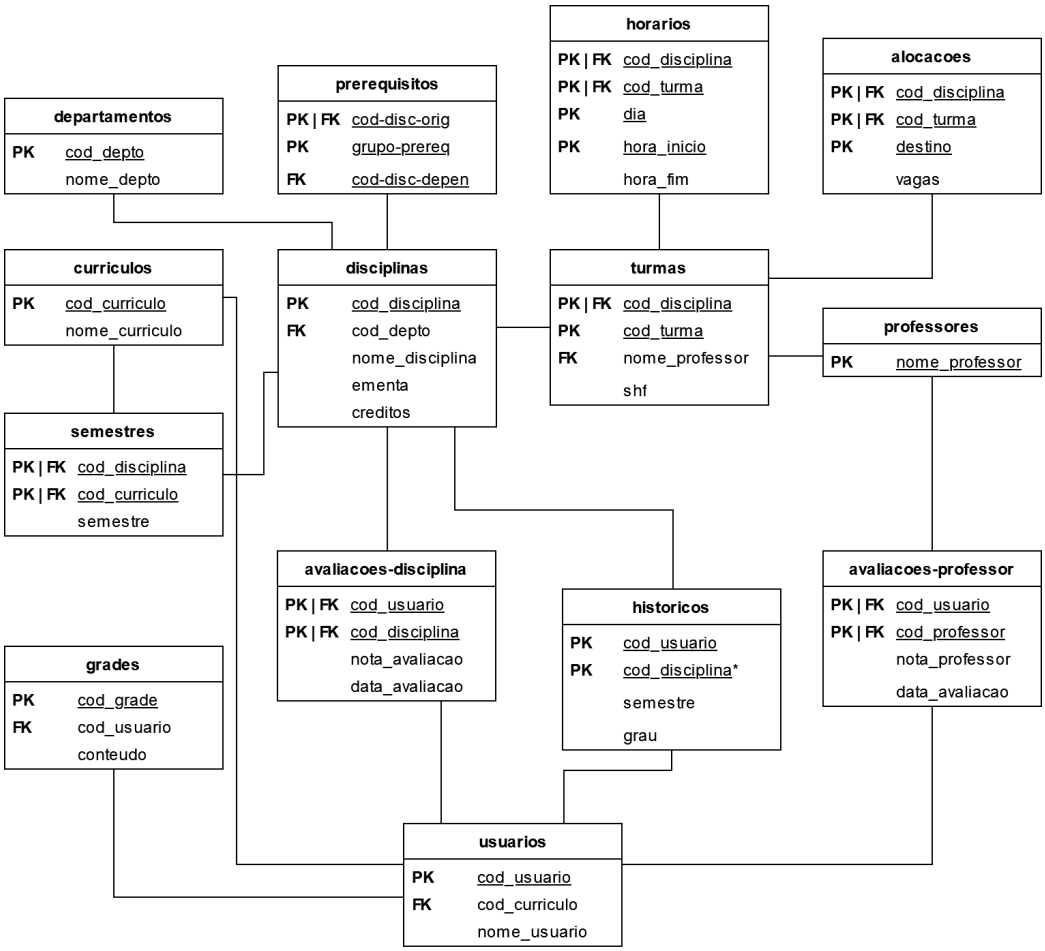

Figura 9.2: Diagrama do modelo lógico dos dados

9.3 Dicionário de dados

O dicionário de dados possui a função de categorizar os dados. Ele é uma coleção de metadados do modelo físico. O dicionario de dados do sistema está disponível na tabela 9.1. Nele está representado todas as colunas observadas no diagrama lógico da figura 9.2. O tipo do dado é unico para o nome da coluna, o que significa que o mesmo nome da coluna em duas tabelas diferentes representa o mesmo tipo de dado. Por exemplo, o tipo de dado na coluna *cod-usuario* é o mesmo na tabela *grades*, *usuarios*, *avaliacoes-disciplina* e *avaliacoes-professor*.

Coluna	**Descrição**	**Tipo**	**Domínio**
cod_curriculo	Identificador do currículo. Exemplo: "CEGBCO20180"	string	Sem restrição.

Coluna	**Descrição**	**Tipo**	**Domínio**
cod_depto	Abreviação de três letras do departamento, conforme disponibilizado no microhorario. Exemplo: "ENG".	string	Três letras maiúsculas.
cod_disciplina	Código da disciplina. Exemplo: "INF1011".	string	Três letras maiúsculas (não necessariamente um departamento) seguidas de 4 números.
cod_disc_depen	Códigos da disciplinas que fazem parte de um grupo de pré-requisitos de outra disciplina.	Array de strings	Mesmo do cod-disciplina.
cod_disc_orig	Código da disciplina que possui algum pré-requisito.	string	Mesmo do cod-disciplina.
cod_grade	Código da grade, gerado ao salvar uma nova grade.	string	8 caracteres em formato base64.
cod_turma	Código da turma.	string	3 caracteres.
cod_usuario	Código do usuário, representado por sua matrícula	string	7 números.
conteudo	Conteúdo codificado da grade horária, que é decodificado pela interface ao exibir.	string	Sem restrição.
creditos	Quantidade de créditos da disciplina.	int	Valor entre 0 a 30.
data_avaliacao	Data em que a avaliação foi efetuada pelo aluno.	date	Sem restrição. Pode ser nulo.

Coluna	**Descrição**	**Tipo**	**Domínio**
dia	Dia da semana que a turma é oferecida	string	"SEG", "TER", "QUA", "QUI", "SEX", "SAB"ou "xxx"(sem dia definido).
destino	Destino das vagas disponíveis. Exemplo: "QQC"(Qualquer curso), "BCO"(Bacharelado em Engenharia de Computação), entre outros.	string	Três letras maiúsculas. Pode ser nulo.
ementa	Ementa da disciplina.	string	Sem restrição. Pode ser nulo.
grau	Grau do aluno na disciplina.	int	Entre 0 a 100. Pode ser nulo.
grupo_prereq	Um identificador do grupo do pré-requisito. Um aluno só pode cursar a disciplina se todas as disciplinas do mesmo grupo foram cursadas.	int	Sem restrição.
hora_fim	Hora do fim da aula da turma	int	Valor entre 9 e 23, deve ser maior que hora-inicio. Pode ser nulo (sem hora de fim).
hora_inicio	Hora do inicio da aula da turma	int	Valor entre 7 e 21, ou 0 (sem hora de início).

Coluna	**Descrição**	**Tipo**	**Domínio**
nome_curriculo	Nome curto do currículo. Exemplo: "Engenharia | 19.0".	string	Sem restrição.
nome_depto	Nome do departamento, conforme disponibilizado no microhorario. Exemplo: "Engenharia".	string	Sem restrição.
nome_disciplina	Nome da disciplina. Exemplo: "Semântica de Linguagens".	string	Sem restrição.
nome_professor	Nome do professor, conforme disponibilizado no microhorário	string	Sem restrição.
nome_usuario	Nome do usuário	string	Sem restrição.
nota_avaliacao	Nota da disciplina ou professor avaliado.	int	Valor entre 1 e 5.
semestre	Semestre recomendado para cursar a disciplina de acordo com o currículo.	int	Valor entre 1 e 10. Pode ser nulo.
shf	Quantidade de horas "Sem Horário Fixo"de uma disciplina.	int	Valor maior ou igual a zero. Pode ser nulo.
vagas	Quantidade de vagas disponíveis.	int	Valor maior que 0.

Tabela 9.1: Dicionário de dados

10
Construção

10.1
Tecnologias utilizadas

Para o armazenamento e consulta dos dados, foi utilizado o banco de dados relacional PostgreSQL [9]. O PostgreSQL foi escolhido por ser robusto e eficiente, mas fácil e rápido de configurar.

O sistema foi separado em interface e API (*Application programming interface*, ou interface de programação da aplicação). A coleta dos dados provenientes do Microhorario, a autenticação com o sistema acadêmico universitário e carga/processamento dos históricos foi desenvolvida em serviços menores uma única funcionalidade respecitvamente (microserviços).

A figura 10.1 exibe um diagrama da arquitetura do sistema.

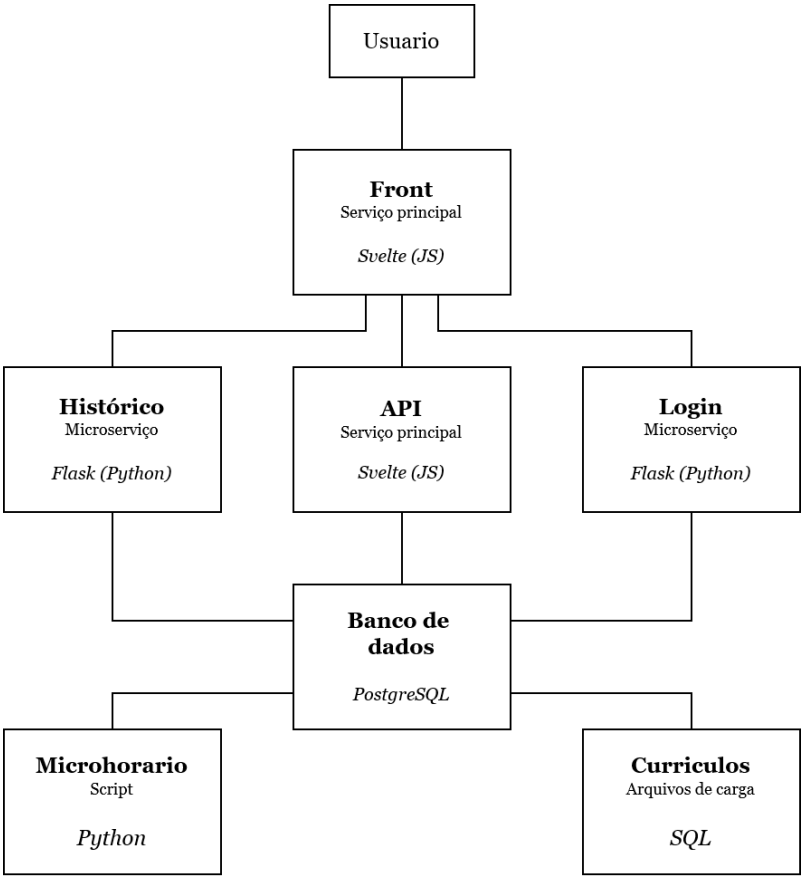

Figura 10.1: Diagrama da arquitetura do sistema

10.2 Coleta de dados

O o modelo lógico representado na figura 9.2 define a construção de catorze tabelas no banco de dados. O trecho de código 1 contém o modelo físico do banco de dados, ou seja o código SQL que define a modelagem das tabelas.

Código 1: Modelo físico

```
1  CREATE TABLE IF NOT EXISTS departamentos (
2      cod_depto CHAR(3) PRIMARY KEY,
3      nome_depto TEXT NOT NULL
4  );
5
6  CREATE TABLE IF NOT EXISTS disciplinas (
7      cod_disciplina CHAR(7) PRIMARY KEY,
8      cod_depto CHAR(3) NOT NULL,
9      nome_disciplina TEXT NOT NULL,
10     ementa TEXT,
11     creditos SMALLINT NOT NULL,
12     FOREIGN KEY (cod_depto) REFERENCES departamentos(cod_depto)
13 );
14
15 CREATE TABLE IF NOT EXISTS prerequisitos (
16     cod_disc_orig CHAR(7) NOT NULL,
17     grupo_prereq SMALLINT NOT NULL,
18     cod_disc_depen CHAR(7) NOT NULL,
19     PRIMARY KEY (cod_disc_orig, grupo_prereq, cod_disc_depen),
20     FOREIGN KEY (cod_disc_orig) REFERENCES disciplinas(
           cod_disciplina)
21 );
22
23 CREATE TABLE IF NOT EXISTS professores (
24     nome_professor TEXT PRIMARY KEY
25 );
26
27 CREATE TABLE IF NOT EXISTS turmas (
28     cod_turma CHAR(3) NOT NULL,
29     cod_disciplina CHAR(7) NOT NULL,
30     nome_professor TEXT NOT NULL,
31     shf SMALLINT NOT NULL,
32     PRIMARY KEY (cod_turma, cod_disciplina),
33     FOREIGN KEY (cod_disciplina) REFERENCES disciplinas(
           cod_disciplina),
34     FOREIGN KEY (nome_professor) REFERENCES professores(
           nome_professor)
35 );
36
37 CREATE TABLE IF NOT EXISTS horarios (
38     cod_disciplina CHAR(7) NOT NULL,
```

```
39      cod_turma CHAR(3) NOT NULL,
40      dia CHAR(3) NOT NULL,
41      hora_inicio SMALLINT NOT NULL,
42      hora_fim SMALLINT,
43      PRIMARY KEY (cod_disciplina, cod_turma, dia, hora_inicio),
44      FOREIGN KEY (cod_disciplina, cod_turma) REFERENCES turmas(
            cod_disciplina, cod_turma)
45  );
46
47  CREATE TABLE IF NOT EXISTS alocacoes (
48      cod_disciplina CHAR(7) NOT NULL,
49      cod_turma CHAR(3) NOT NULL,
50      destino VARCHAR(80) NOT NULL,
51      vagas SMALLINT NOT NULL,
52      PRIMARY KEY (cod_disciplina, cod_turma, destino),
53      FOREIGN KEY (cod_disciplina, cod_turma) REFERENCES turmas(
            cod_disciplina, cod_turma)
54  );
55
56  CREATE TABLE IF NOT EXISTS curriculos (
57      cod_curriculo VARCHAR(11) PRIMARY KEY,
58      nome_curriculo TEXT NOT NULL
59  );
60
61  CREATE TABLE IF NOT EXISTS semestres (
62      cod_disciplina CHAR(7) NOT NULL,
63      cod_curriculo VARCHAR(11) NOT NULL,
64      semestre SMALLINT NOT NULL,
65      PRIMARY KEY (cod_disciplina, cod_curriculo),
66      FOREIGN KEY (cod_disciplina) REFERENCES disciplinas(
            cod_disciplina),
67      FOREIGN KEY (cod_curriculo) REFERENCES curriculos(cod_curriculo)
68  );
69
70  CREATE TABLE IF NOT EXISTS usuarios (
71      cod_usuario CHAR(7) PRIMARY KEY,
72      nome_usuario TEXT NOT NULL,
73      cod_curriculo VARCHAR(11),
74      FOREIGN KEY (cod_curriculo) REFERENCES curriculos(cod_curriculo)
75  );
76
77  CREATE TABLE IF NOT EXISTS grades (
78      cod_grade CHAR(8) PRIMARY KEY,
79      cod_usuario CHAR(7) NOT NULL,
80      conteudo TEXT NOT NULL,
81      FOREIGN KEY (cod_usuario) REFERENCES usuarios(cod_usuario)
82  );
83
84  CREATE TABLE IF NOT EXISTS historicos (
85      cod_usuario CHAR(7) NOT NULL,
86      cod_disciplina CHAR(7) NOT NULL,
```

```
87      semestre SMALLINT NOT NULL,
88      grau SMALLINT,  -- a disciplina pode nao ter nota
89      PRIMARY KEY (cod_usuario, cod_disciplina, semestre),
90      FOREIGN KEY (cod_usuario) REFERENCES usuarios(cod_usuario),
91      FOREIGN KEY (cod_disciplina) REFERENCES disciplinas(cod_usuario)
92  );
93
94  CREATE TABLE IF NOT EXISTS avaliacoes_disciplinas (
95      cod_usuario CHAR(7) NOT NULL,
96      cod_disciplina CHAR(7) NOT NULL,
97      nota_avaliacao SMALLINT NOT NULL,
98      data_avaliacao DATE NOT NULL,
99      PRIMARY KEY (cod_usuario, cod_disciplina),
100     FOREIGN KEY (cod_usuario) REFERENCES usuarios(cod_usuario)
101     FOREIGN KEY (cod_disciplina) REFERENCES disciplinas(
            cod_disciplina)
102 );
103
104
105 CREATE TABLE IF NOT EXISTS avaliacoes_professores (
106     cod_usuario CHAR(7) NOT NULL,
107     nome_professor TEXT NOT NULL,
108     nota_avaliacao SMALLINT NOT NULL,
109     data_avaliacao DATE NOT NULL,
110     PRIMARY KEY (cod_usuario, nome_professor),
111     FOREIGN KEY (cod_usuario) REFERENCES usuarios(cod_usuario)
112     FOREIGN KEY (nome_professor) REFERENCES professores(
            nome_professor)
113 );
114
115
116 CREATE TABLE IF NOT EXISTS modificacao (
117     data_ementa timestamptz NOT NULL,
118     data_geral  timestamptz NOT NULL,
119     modo_fallback boolean NOT NULL,
120     PRIMARY KEY (data_ementa, data_geral, modo_fallback)
121 )
```

Observando o modelo físico, é possível observar que é possível observar que as avaliações, históricos e currículos devem ser sempre associados às disciplinas, através das chaves estrangeiras. Portanto, as disciplinas não podem ser excluídas, mesmo que elas não possam não estar sendo oferecidas no semestre atual. Para diferenciar uma disciplina que está sendo oferecida de uma disciplina que não está disponível, basta verificar se ela existe na tabela `turmas`, que é constantemente atualizada conforme as disponibilidades.

No modelo físico, existe a criação de uma tabela `modificacao`, que não estava prevista no modelo lógico. Essa tabela não contém dados relacionais, mas apenas três valores armazenados em uma só linha da tabela. Esses valores

representam as datas de coleta das informações das disciplinas e turmas, e se os dados estão em modo de `fallback`, que será explicado depois.

Além dos dados gerados pela interação do usuário, o sistema precisa de dados provenientes da faculdade, como as disciplinas, turmas, professores e currículos. Esses dados são coletados ocasionalmente, exceto os currículos, que foram manualmente costruídos a partir de informações disponíveis nas plataformas da universidade. Como o algoritmo e o sistema foi planejado para alunos do departamento de informática, a quantidade de currículos a ser inserida é pequena. No total, foram inseridos seis currículos: três de engenharia de computação (Currículos 2023, 2018.0 e 2018.1) e três de ciência de computação, disponíveis nas respectivas páginas [1] [2] dos cursos.

Os dados do currículo foram transformados manualmente em um arquivo no formato SQL que insere o código do currículo e constrói uma `PROCEDURE`, um procedimento responsável por inserir as disciplinas referentes ao currículo. As disciplinas estão dentro de uma verb|procedure| pois esta deve ser executada toda vez que uma disciplina nova possa ter sido inserida no banco. Isso é necessário pois a `procedure` armazena somente o código de disciplinas que existem no banco, devido a restrição de chave estrangeira da tabela. Portanto, cada vez que uma disciplina nova é inserida, essa operação precisa ser refeita. O trecho de código 2 contém um exemplo de um dos seis arquivos criados, um para cada currículo.

Código 2: Exemplo do arquivo de carga de currículos

```
1  INSERT INTO curriculos (cod_curriculo, nome_curriculo)
2  SELECT * FROM (
3      VALUES ('CCPBCC20181', 'Ciência da Computação (Bacharelado) -
           Currículo 18.1')
4  ) AS i (cod_curriculo, nome_curriculo)
5  WHERE NOT EXISTS (
6      SELECT cod_curriculo
7      FROM curriculos
8      WHERE cod_curriculo = i.cod_curriculo
9  );
10
11 CREATE OR REPLACE PROCEDURE inserir_curriculo_cie_18_1()
12 LANGUAGE SQL
13 BEGIN ATOMIC
14     DELETE FROM semestres WHERE cod_curriculo = 'CCPBCC20181';
15
16     INSERT INTO semestres (cod_curriculo, cod_disciplina, semestre)
17     SELECT * FROM (VALUES
18         ('CCPBCC20181', 'INF1025', 1),
```

[1] <https://www.puc-rio.br/ensinopesq/ccg/eng_computacao.html>
[2] <https://www.puc-rio.br/ensinopesq/ccg/ciencia_computacao.html>

```
19          ('CCPBCC20181', 'INF1012', 1),
20          ('CCPBCC20181', 'INF1009', 1),
21          ('CCPBCC20181', 'INF1031', 1),
22          -- ....
23          -- ....
24          ('CCPBCC20181', 'INF0381', 8),
25          ('CCPBCC20181', 'INF1951', 8)
26      ) AS i (cod_curriculo, cod_disciplina, semestre)
27      WHERE EXISTS (
28          SELECT cod_disciplina
29          FROM disciplinas
30          WHERE cod_disciplina = i.cod_disciplina
31      );
32  END
```

Os dados provenientes das disciplinas e turmas são coletados diretamente através do microhorário. Existe uma biblioteca [3] Python [10] que permite baixar todos as informações disponíveis no microhorário, e opcionalmente coletar as ementas e pré-requisitos que estão disponíveis em página. A biblioteca baixa os dados das disciplinas e turmas em menos de três segundos, mas pode executar por até 30 minutos para baixar as ementas e pré-requisitos que estão em outro serviço da universidade. Portanto, foi criado um módulo, chamado `microhorario`, que permite atualizar o banco com as informações básicas das disciplinas e turmas, mas que pode ser alterado para atualizar também as ementas e pré-requisitos. Esse módulo foi configurado de forma a ser executado na sua forma mais simples (somente atualiza os dados referentes às turmas) uma vez a cada hora, e executado na forma completa (coleta inclusive as ementas e pré-requisitos) uma vez por semana.

No final de cada período, devido a preparação das disciplinas e turmas para o próximo semestre, o microhorário não disponibiliza dados como quantidade de vagas e quantidade de créditos. Por isso, foi necessário que o sistema e o algoritmo acomodasse essas necessidades temporárias. Nesse caso, a variável de `fallback`, presente na tabela de `modificacao` estará no seu valor verdadeiro. Se o microhorário estiver disponível, essa variável está com seu valor falso.

10.3 Definição e implementação do algoritmo

O algoritmo busca satisfazer as necessidades apresentadas na tabela 5.3. As entrevistas indicaram a importância de cinco categorias (Conteúdo,

[3] Disponível em: <https://pypi.org/project/microhorario-dl/>

Professor, Avaliação, Horário e Opinião). Por isso, o algoritmo foi dividido nessas cinco categorias.

O algoritmo se baseia num valor $V[d, u]$ atribuído a cada disciplina, que indica a relevância da disciplina d para o usuário u. O valor $V[d, u] \in [0, 1]$, sendo 1 o maior grau de relevância da disciplina, e 0 o menor. O cálculo de V é uma média ponderada, conforme a equação 10-1.

$$V[d,u] = P_c V_c[d,u] + P_p V_p[d] + P_o V_o[d] + P_h V_h[d,u] + P_a V_a[d] \quad (10\text{-}1)$$

Em que d é uma disciplina, u é um usuário, P_x é o valor de uma das cinco categorias para a disciplina d e o usuário u, e P_x é o peso de uma das cinco categorias.

O valor $V_c[d, u]$ indica o quão relevante é a disciplina para o usuário de acordo com o seu conteúdo. Para isso, o valor é calculado de acordo com o histórico de outros alunos e com o currículo do aluno, conforme a equação 10-2.

$$V_c[d,u] = \begin{cases} 1.0 & \text{caso } d \in Historico[u] \\ \dfrac{|\, A_{cursou}[d,u] \,|}{|\, A_{curriculo}[u] \,|} & \text{caso } A_{curriculo}[u] > 0 \\ 0 & \text{caso contrário} \end{cases} \quad (10\text{-}2)$$

Em que:

$$A_{curriculo}[u] = \forall a \in Alunos \,|\, Curriculo[a] = Curriculo[u]$$
$$A_{cursou}[d,u] = \forall a \in A_{curriculo}[u] \,|\, d \in Cursadas[a]$$

Em que d é uma disciplina, u é um usuário, *Alunos* é o conjunto de todos os usuários cadastrados no sistema, $Curriculo[u]$ é o currículo do usuário u, e $Cursadas[u]$ é o conjunto de disciplinas que o aluno u cursou. Em resumo, o valor $V_c[d, u]$ é o valor máximo caso a disciplina deve ser cursada pelo usuário, ou sendo uma eletiva é a proporção de alunos do mesmo currículo que fizeram esta disciplina. Essa proporção indica se o conteúdo é relevante para alunos semelhantes ao usuário.

O valor $V_p[d]$ indica o quão relevante é a disciplina para o usuário de acordo com o professor. Para isso, o valor é calculado de acordo com as

avaliações dos professores das turmas das disciplinas, conforme a equação 10-3.

$$V_p[d] = \frac{\sum_{p \in P[d]} \frac{\sum_{a \in Avs[p]} a}{|Avs[p]|}}{|P[d]|} \,/\, 5 \tag{10-3}$$

Em que d é uma disciplina, $P[d]$ é o conjunto dos professores que lecionam a disciplina d, e $Avs[p]$ representa o conjunto das avaliações dos usuários do professor p. Em resumo, o valor $V_p[d, u]$ é a média das avaliações de todos os professores que estão lecionando a disciplina, com o valor entre 0 e 1.

O valor $V_o[d]$ indica o quão relevante é a disciplina para o usuário de acordo com a opinião dos alunos. A equação é semelhante ao cálculo apresentado na equação 10-3. Porém, nesse caso, é levado em conta a média das avaliações das próprias disciplinas, conforme a equação 10-4.

$$V_p[d] = \frac{\sum_{a \in Op[d]} a}{|Op[d]|} \,/\, 5 \tag{10-4}$$

Em que d é uma disciplina, e $Op[d]$ é o conjunto das avaliações dos usuários da disciplina d.

O valor $V_h[d, u]$ indica o quão relevante é a disciplina para o aluno de acordo com o horário. Para isso, o valor é calculado de acordo com os horários ocupados na grade do usuário e os horários das disciplinas, conforme a equação 10-5

$$V_h[d, u] = \begin{cases} \frac{|\, T_{possiveis}[d, u] \,|}{|\, T[d] \,|} & \text{caso } T[d] > 0 \\ 0 & \text{caso contrário} \end{cases} \tag{10-5}$$

Em que d é uma disciplina, u é um usuário, $T_{possiveis}[d, u]$ é o conjunto de turmas da disciplina d que se encaixam na grade do usuário u, e $T[d]$ é a quantidade de turmas da disciplina d. Em resumo, $V_h[d, u]$ é a porcentagem de turmas da disciplina que se encaixam na grade do usuário.

Por último, o valor $V_a[d]$ indica o quão relevante é a disciplina para o aluno de acordo com os graus (a nota final, considerando as provas do aluno durante o semestre) da disciplina. O cálculo é semelhante às equações 10-3 e 10-4, conforme a equação 10-6.

$$V_p[d] = \frac{\sum_{g \in Graus[d]} g}{|Graus[d]|} \,/\, 100 \tag{10-6}$$

Em que d é uma disciplina e $Graus[d]$ é o conjunto das notas da disciplina d.

Para obter todos os valores dos conjuntos citados nas equações anteriores, foi criada uma consulta SQL que obtém todos os valores de uma só vez. O trecho de código 3 contém a consulta por completo. Antes de executar, são

substituidos os valores `@usuario` pelo código do usuário, e o valor `@escolhas` por um trecho de código SQL referente às escolhas de turmas de disciplinas na grade atual do aluno.

Código 3: Consulta dos dados para o algoritmo

```
WITH rec_c1 AS (
    SELECT cod_disciplina
    FROM semestres
    WHERE cod_curriculo =
          (SELECT cod_curriculo FROM usuarios where cod_usuario =
              @usuario)
), rec_c2_1 AS (
    SELECT s.cod_disciplina, count(u.cod_usuario) as qtd
    FROM semestres s, usuarios u
    WHERE s.cod_curriculo = u.cod_curriculo
      AND u.cod_curriculo =
          (SELECT cod_curriculo FROM usuarios where cod_usuario =
              @usuario)
    GROUP BY s.cod_disciplina
), rec_c2_2 AS (
    SELECT count(cod_usuario)
    FROM usuarios
    WHERE cod_curriculo =
          (SELECT cod_curriculo FROM usuarios where cod_usuario =
              @usuario)
), rec_h AS (
    SELECT h1.cod_disciplina, count(DISTINCT h1.cod_turma) as
        possiveis, todas
  FROM horarios h1
    LEFT JOIN (
      SELECT cod_disciplina, count(cod_turma) as todas
      FROM turmas
      GROUP BY cod_disciplina
  ) AS h2 ON h1.cod_disciplina = h2.cod_disciplina
  WHERE (dia, hora_inicio, hora_fim) NOT IN (
    SELECT dia, hora_inicio, hora_fim
    FROM horarios
    WHERE @escolhas
  )
  GROUP BY h1.cod_disciplina, todas
), rec_o AS (
    SELECT cod_disciplina, avg(nota_avaliacao) as media
    FROM avaliacoes_disciplinas
    GROUP BY cod_disciplina
), rec_p AS (
    SELECT d.cod_disciplina, avg(a.nota_avaliacao) as media
    FROM disciplinas d, avaliacoes_professores a, turmas t
    WHERE d.cod_disciplina = t.cod_disciplina
    AND t.nome_professor = a.nome_professor
    GROUP BY d.cod_disciplina, d.nome_disciplina
```

```
42 ), rec_a AS (
43     SELECT cod_disciplina, avg(grau) as media
44     FROM historicos
45     GROUP BY cod_disciplina
46 ), filtro AS (
47     SELECT d.cod_disciplina
48     FROM disciplinas d
49     WHERE cod_disciplina NOT IN (
50         SELECT h.cod_disciplina
51         FROM historicos h
52         WHERE cod_usuario = @usuario
53     )
54 )
55
56 SELECT f.cod_disciplina,
57     c1.cod_disciplina IS NOT NULL conteudo1,
58     c2_1.qtd conteudo21,
59     c2_2.count conteudo22,
60     h.cod_disciplina IS NOT NULL horario,
61     o.media opiniao,
62     p.media professor,
63     a.media avaliacao
64 FROM filtro f
65     LEFT JOIN rec_c1 c1 ON f.cod_disciplina = c1.cod_disciplina
66     LEFT JOIN rec_c2_1 c2_1 ON f.cod_disciplina = c2_1.
           cod_disciplina
67     LEFT JOIN rec_h h ON f.cod_disciplina = h.cod_disciplina
68     LEFT JOIN rec_o o ON f.cod_disciplina = o.cod_disciplina
69     LEFT JOIN rec_p p ON f.cod_disciplina = p.cod_disciplina
70     LEFT JOIN rec_a a ON f.cod_disciplina = a.cod_disciplina,
71     rec_c2_2 c2_2
```

Por fim, os pesos P_x da média ponderada foram baseados na tabela 5.3. Cada peso é a soma dos pesos dos critérios, de acordo com as equações a seguir.

$$P_c = \frac{41}{41 + 19 + 8 + 28 + 21} \tag{10-7}$$

$$P_p = \frac{28}{41 + 19 + 8 + 28 + 21} \tag{10-8}$$

$$P_o = \frac{8}{41 + 19 + 8 + 28 + 21} \tag{10-9}$$

$$P_h = \frac{19}{41 + 19 + 8 + 28 + 21} \tag{10-10}$$

$$P_a = \frac{21}{41 + 19 + 8 + 28 + 21} \tag{10-11}$$

10.4 Implementação da API

A API é responsável por servir disponibilizar à interface os dados de forma organizada e eficaz. Foi estudada a implementação da API em quatro possíveis frameworks em três diferentes linguagens: *Django* [11], *Flask* [12], *Gin* [13] e *Rocket* [14]. Cada um dos frameworks possui suas vantagens e desvantagens.

O framework *Django* é um *Web Framework* completo, que possui múltiplas funcionalidades pré-configuradas, possui uma interface de comunicação com banco de dados bem robusta. O framework *Flask* por sua vez não possui muitas funcionalidades imbutidas, e depende da instalação de pacotes externos para ampliar suas funcionalidades. Ambos os frameworks são desenvolvidos na linguagem Python, o que torna o desenvolvimento mais fácil, mas reduz performance do funcionamento, por ser uma linguagem interpretada.

O framework *Rocket* é desenvolvido na linguagem Rust [15], conhecida por ser rápida e segura, por possuir um abordagem de manipulação de memória diferente de outras linguagens. Porém, Rust possui uma alta curva de aprendizado, dificultando o desenvolvimento do código.

Por fim, o framework *Gin* é desenvolvido na linguagem Go [16], conhecida por bem eficiente, útil para o desenvolvimento de APIs pelo sua capacidade de multiprocessamento, e fácil de usar. Por isso, esse framework foi escolhido para a API do sistema.

A API disponibiliza a documentação completa de todas as suas rotas, com as respectivas entradas e saídas, conforme a figura 10.2.

10.5 Implementação dos microserviços

Os serviços de autenticação, e de carga e processamento dos históricos dos alunos são processos separados da API. Ambos foram desenvolvidos na linguagem Python, por sua facilidade de processamento dos dados, e por não haver a necessidade uma performance ótima. Esses utilizam o framework Flask para disponibilizar os serviços, por serem serviços bem simples.

O serviço de autenticação se comunica com o sistema acadêmico universitário (SAU). Existe uma API para autenticação dos alunos da universidade, mas é restrita para os serviços interno da mesma. Por isso, o serviço de autenticação implementado simula um aluno autenticando-se no portal [4] do SAU, e verifica se a autenticação foi efetuada com sucesso.

[4]Disponível em: <https://www.puc-rio.br/ensinopesq/academicas/>, Sistemas Acadêmicos - SAU

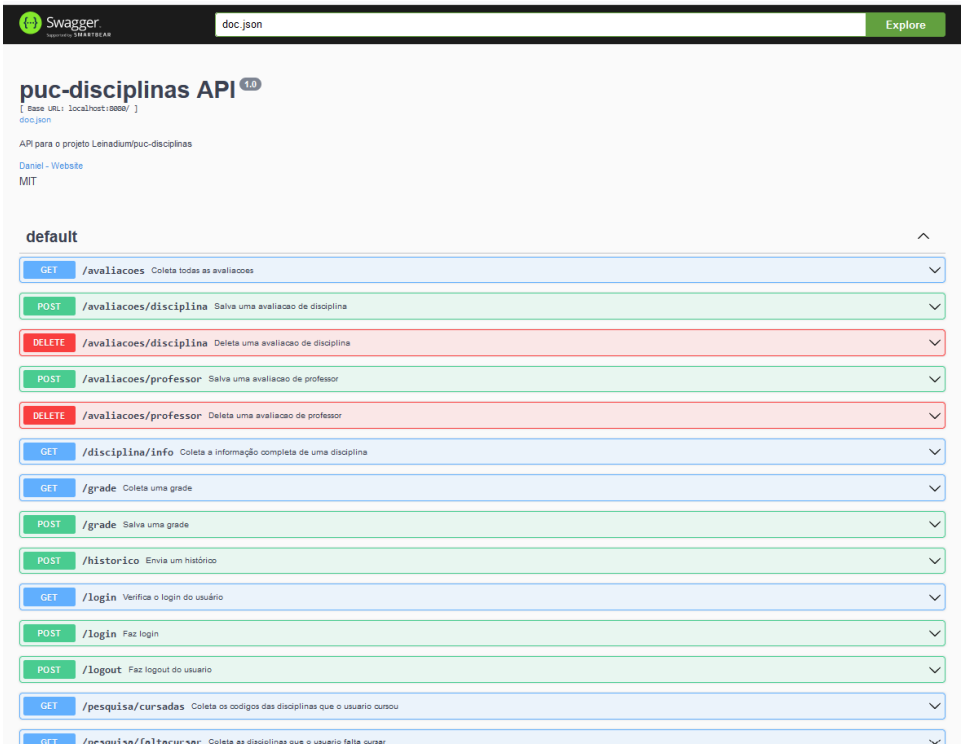

Figura 10.2: Documentação da API

O serviço de carga dos históricos precisa receber como entrada a página do histórico do usuário. É possível coletar o histórico de forma automática, acessando o portal do SAU também simulando um usuário. Porém, como essa interação com o SAU não seria transparente com o usuário, sendo executado de forma oculta, não era a melhor opção. Por isso, optou-se pelo usuário salvando uma cópia da página do seu histórico, e submtendo-o através da interface, que se comunica com o serviço de carga e processamento dos históricos.

10.6 Implementação da interface

A interface foi implementada utilizando o framework *Svelte*, que permite desenvolver páginas interativas baseadas em componentes. A programação é feita utilizando um misto de Javascript e HTML, que é compilada em pacotes Javascript pequenos que são interpretados pelo navegador de internet.

Como foi necessário desenvolver pelo menos três páginas diferentes, conforme o wireframe do capítulo 8, foi utilizado o framework *SvelteKit* [17]. Este permite disponibilizar páginas desenvolvidas em Svelte em diferentes rotas, além de outras possibilidades, como SSR (*Server-Side Rendering*, ou renderização no servidor), que permite reduzir o esforço do navegador do usuário ao construir inicialmente a página no servidor.

A seguir estão algumas imagens da interface implementada do sistema. A figura 10.3 mostra a tela inicial, com um menu de seleção para a página de grade e de avaliações. As figuras 10.4 e 10.5 exibem a interface de criação de grade horária. Na figura 10.4 é possível observar a mensagem de aviso, caso os dados completos do microhorário estejam indisponíveis.

Figura 10.3: Implementação da tela inicial da interface

As disciplinas recomendadas podem possuir informações extras. A figura 10.7 mostra uma legenda disponível no sistema explicando as informações exibidas. A figura explica que uma disciplina pode estar sendo recomendada devido a 5 categorias, que estão associadas aos pesos explicados na seção 10.3.

Figura 10.4: Implementação da tela de criação de grade da interface

Figura 10.5: Detalhes de uma disciplina na interface, exibido ao selecionar uma disciplina

Figura 10.6: Implementação da tela de avaliação da interface

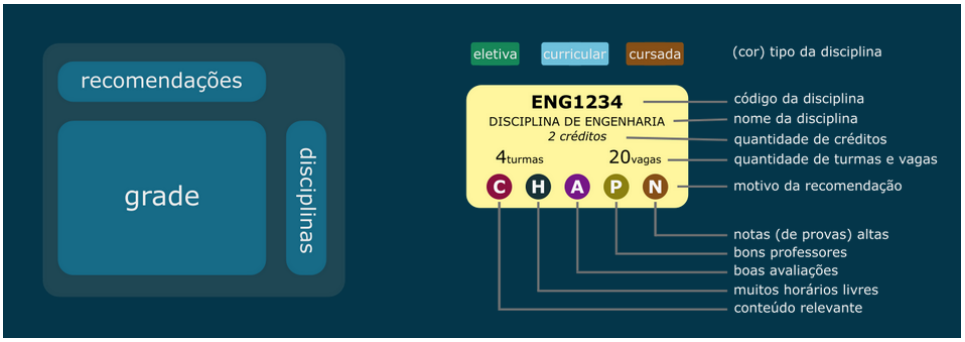

Figura 10.7: Legenda da interface de criação de grade horária

11
Testes

Afim de satisfazer todos os requisitos descritos no capítulo 6, foram efetuados testes do sistema e algoritmo durante toda a fase de desenvolvimento. Esses testes eram efetuados de maneira não estruturada, apenas para testar todos os possíveis casos de erro. Foi considerada a metodologia de *Desenvolvimento orientado a testes*, que planeja o desenvolvimento primeiro de testes unitários e de integração, para depois o desenvolvimento da aplicação em si. Porém, a implantação dessa metodologia costuma aumentar consideravelmente o tempo de desenvolvimento de aplicações, por isso ela não foi implementada.

Por não ter testes unitários ou de integração, o desenvolvimento precisou ser meticuloso. Por isso, optou-se por utilizar Typescript na interface, um superconjunto de Javascript, o que significa que suporta a sintaxe de Javascript e suas funcionalidades, mas introduz outras funcionalidades, que no caso são os tipos. Esses tipos permitem analisar o código em tempo de compilação, exibindo-os durante o processo de desenvolvimento, tornando o código mais resistente a falhas. Do mesmo modo, nos microserviços implementados em Python foram usados a sintaxe de tipos, pela mesma vantagem. Python é uma linguagem dinâmica em que os tipos não são necessários, mas foram utilizados para trazer mais segurança. Por último, a API foi implementada em Go, uma linguagem compilada *fortemente tipada*, o que significa que todas as variáveis possuem um tipo específico.

O trecho de código 3 exibe a consulta ao banco para coletar os parâmetros de entrada do algoritmo. É possível observar que a consulta procura através das tabelas `disciplinas` e `turmas`, procura na tabela `semestres` (que contém as disciplinas dos currículos), procura nas tabelas de avaliações, procura na tabela `historicos` e utiliza-se de um filtro com as disciplinas e turmas já escolhidas pelo aluno, satisfazendo os requisitos RF01, RF02, RF03, RF04 e RF05. A fórmula matemática satisfaz o requisito RF06. Portanto, todos os requisitos funcionais do algoritmo foram satisfeitos.

Como a implementação do algoritmo é um modelo matemático, pode-se observar que o requisito RNF2 (que determina que o algoritmo deve retornas os mesmos resultados para as mesmas entradas) é satisfeito.

Foram efetuados testes de performance para verificar se os requisitos RNF1 e RNF3 foram satisfeitos. A figura 11.1 representa um teste de performance, com 40 usuários simultâneos autenticados solicitando recomendações diferentes a cada 1-5 segundos. É possível observar que o tempo médio de

resposta das recomendações foi 20 milisegundos, e não houve nenhuma falha no sistema ou algoritmo. O teste de performance foi desenvolvido usando o framework *Locust* [18], utilizando a linguagem Python.

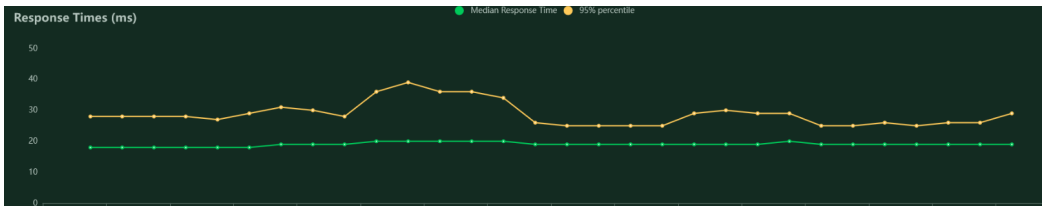

Figura 11.1: Teste de performance da recomendação

Para satisfazer os requisitos do sistema, a interface e a API precisam agir de forma integrada. A lista abaixo contém os requisitos do sistema, o(s) componente(s) que os satisfazem, e a rota da API que é utilizada para alimentar a interface.

- **RF07:** O sistema deve permitir que o usuário crie uma grade horária para o próximo período.
  - **Interface:** `grade/+page.svelte`
  - **API:** `/pesquisa/{info,podecursar,faltacursar,cursadas}`
- **RF08:** O sistema deve permitir que o usuário submita seu histórico escolar.
  - **Interface:** `HistoricoPopup.svelte`
  - **API:** `/historico`
- **RF09:** O sistema deve permitir que o usuário selecione turmas das disciplinas para compor sua grade horária.
  - **Interface:** Componentes em `components/grade/turma`
  - **API:** `/disciplinas/info`
- **RF10:** O sistema deve permitir que o usuário armazene a sua grade horária finalizada.
  - **Interface:** `JanelaSalvar.svelte` e `GrupoBotoes.svelte`
  - **API:** `/grade`
- **RF11:** O sistema deve permitir que o usuário compartilhe a sua grade horária finalizada.
  - **Interface:** `JanelaSalvar.svelte`

  - **API:** `/grade`

- **RF12:** O sistema deve permitir que o usuário recupere uma grade horária montada a partir de um link compartilhado.
  - **Interface:** `grade/+page.svelte`
  - **API:** `/grade`

- **RF13:** O sistema deve permitir que o usuário inicie uma sessão autenticada utilizando sua matrícula.
  - **Interface:** `CheckLogin.svelte` e `LoginPopup.svelte`
  - **API:** `/login`

- **RF14:** O sistema deve permitir que o usuário finalize uma sessão autenticada.
  - **Interface:** `LoginHeader.svelte`
  - **API:** `/logout`

- **RF15:** O sistema deve permitir que o usuário avalie uma disciplina.
  - **Interface:** `avaliacao/+page.svelte`
  - **API:** `/avaliacoes/`

- **RF16:** O sistema deve permitir que o usuário avalie um professor de uma disciplina.
  - **Interface:** `CampoAvaliacao.svelte`
  - **API:** `/avaliacoes/professor`

- **RF17:** O sistema deve permitir que o usuário altere uma avaliação feita anteriormente.
  - **Interface:** `CampoAvaliacao.svelte`
  - **API:** `/avaliacoes/disciplina`

12
Uso real

Ao ter-se uma versão com todas os requisitos atendidos, o processo de testes com os usuários deu-se inicío. A metodologia dos testes foi:

1. Apresentar a ideia do sistema e do algoritmo de recomendação;
2. Apresentar todas as possíveis funcionalidades do sistema;
3. Criar uma grade horária como exemplo, utilizando-se de todas as funcionalidades do sistema;
4. Permitir que o usuário faça sua autenticação e crie um grade horária;
5. Ouvir o feedback do usuário.

Os testes com os usuários não foram triviais. A interface do sistema ainda estava em desenvolvimento. Todas as funcionalidades estavam implementadas, mas não claramente especificadas e explicadas. Por isso, em um primeiro momento os usuários precisavam ser guiados cuidadosamente através da interface para interagir com as funcionalidades

13
Conclusão

14
Referências bibliográficas

1 PUC-RIO, S. A. U. *Nova matrícula de graduação PUC-Rio*. 2018. Disponível em: <https://www.cbctc.puc-rio.br/Publicacao/Paginas/Files/20181/Apresenta%C3%A7%C3%A3oNovaMatr%C3%ADculaPUC-Rio_09jul18.pdf>. Acesso em: 25 abr. 2023.

2 NG, Y.-K.; LINN, J. Crsrecs: A personalized course recommendation system for college students. In: *2017 8th International Conference on Information, Intelligence, Systems & Applications (IISA)*. [S.l.: s.n.], 2017. p. 1–6.

3 RANI, L. P. J. et al. Course recommendation for students using machine learning. In: *2020 International Conference on Electronics and Sustainable Communication Systems (ICESC)*. [S.l.: s.n.], 2020. p. 381–384.

4 NGUYEN, V. A. et al. A course recommendation model for students based on learning outcome. In: *2021 Education and Information Technologies*. [S.l.: s.n.], 2021. (26), p. 5389–5415.

5 ADAK, M. F.; YUMUSAK, N.; TASKIN, H. An elective course suggestion system developed in computer engineering department using fuzzy logic. In: *2016 International Conference on Industrial Informatics and Computer Systems (CIICS)*. [S.l.: s.n.], 2016. p. 1–5.

6 XU, J.; XING, T.; SCHAAR, M. van der. Personalized course sequence recommendations. *IEEE Transactions on Signal Processing*, v. 64, n. 20, p. 5340–5352, 2016.

7 ADAK, M. F.; ERCAN, S. Support vector machine and decision tree-based elective course suggestion system: A case study. In: *2021 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*. [S.l.: s.n.], 2021. p. 552–556.

8 CHEN, P. P.-S. The entity-relationship model—toward a unified view of data. *ACM Trans. Database Syst.*, Association for Computing Machinery, New York, NY, USA, v. 1, n. 1, p. 9–36, mar 1976. ISSN 0362-5915. Disponível em: <https://doi.org/10.1145/320434.320440>.

9 The PostgreSQL Global Development Group. *PostgreSQL: The World's Most Advanced Open Source Relational Database*. 2023. Disponível em: <https://www.postgresql.org/>. Acesso em: 02 nov. 2023.

10 Python Software Foundation. *Welcome to Python.org*. 2023. Disponível em: <https://www.python.org>. Acesso em: 02 nov. 2023.

11 Django Software Foundation. *The web framework for perfectionists with deadlines | Django*. 2023. Disponível em: <https://www.djangoproject>. Acesso em: 02 nov. 2023.

12 Pallets. *Welcome to Flask*. 2023. Disponível em: <https://flask.palletsprojects.com>. Acesso em: 02 nov. 2023.

13 Gin Team. *Gin Web Framework*. 2023. Disponível em: <https://gin-gonic.com>. Acesso em: 02 nov. 2023.

14 Sergio Benitez. *Rocket - Simple, Fast, Type-Safe Web Framework for Rust*. 2023. Disponível em: <https://www.rust-lang.org>. Acesso em: 03 nov. 2023.

15 The Rust Foundation. *Rust Programming Language*. 2023. Disponível em: <https://www.rust-lang.org>. Acesso em: 03 nov. 2023.

16 Google. *The Go Programming Language*. 2023. Disponível em: <https://go.dev>. Acesso em: 02 nov. 2023.

17 Svelte Contributers. *SvelteKit - Web development, streamlined*. 2023. Disponível em: <https://kit.svelte.dev>. Acesso em: 03 nov. 2023.

18 C. Bystrom, J. Heyman, J. Hamrém, H. Heyman e L. Holmberg. *Locust - A modern load testing framework*. 2023. Disponível em: <https://locust.io>. Acesso em: 04 nov. 2023.