LUIZ CARLOS COELHO CONDE

ALGORITMO DE GERAÇÃO DE CARDÁPIOS DO RU

LAVRAS - MG

2022

LUIZ CARLOS COELHO CONDE

ALGORITMO DE GERAÇÃO DE CARDÁPIOS DO RU

Projeto Acadêmico apresentado à Universidade Federal de Lavras como parte das exigências do curso de Ciência da Computação, para a obtenção do título de Bacharel.

Mayron César de O. Moreira

Mayron César de Oliveira Moreira

Orientador

LAVRAS - MG

2022

RESUMO

O relatório técnico teve como foco reportar o desenvolvimento de um algoritmo para a geração de cardápios no Restaurante Universitário (RU) da Universidade Federal de Lavras (UFLA). O método proposto preza por simplicidade de implementação e eficiência na resolução do problema, contribuindo assim na tomada de decisão das nutricionistas responsáveis. A ideia da heurística é aplicar filtros sucessivos no conjunto de preparações a serem selecionadas diariamente, considerando as restrições do problema, e maximizando o número de preparações distintas servidas no horizonte de planejamento. Utilizou-se tecnologias web e linguagem de programação Python para a codificação da heurística. O estágio ainda incorporou a heurística ao Gerauca, o Sistema de Geração Automática de Cardápios do RU da UFLA. Os testes realizados com a base de dados fornecida pelo RU mostram que uma solução através da heurística pode ser obtida em frações de segundos.

Palavras-chave: restaurante universitário, geração de cardápios, python, sistema web, heurística, algoritmo.

LISTA DE FIGURAS

LISTA DE CÓDIGOS

SUMÁRIO

1 INTRODUÇÃO

O objetivo deste trabalho consiste na proposição e apresentação de uma heurística construtiva para a construção dos cardápios no contexto do Restaurante Universitário (RU) da UFLA. Além de eficiente, essa solução deve atender a todas as restrições do problema. Não se espera que o algoritmo proposto tenha a mesma acurácia de uma solução obtida pela resolução de uma formulação matemática por um *software* comercial. Por outro lado, a resolução exata do problema poderia se beneficiar de uma solução heurística como ponto de partida, que aprimoraria a convergência da prova de otimalidade do modelo matemático.

Pretende-se, adicionalmente, a inclusão da heurística no Gerauca (Sistema de Geração Automática de Cardápios), desenvolvido por alunos do Programa de Aprendizado Técnico (PROAT) da UFLA.

2 TECNOLOGIAS UTILIZADAS

Este tópico apresenta as tecnologias utilizadas no trabalho

2.1 Python

Python[1] é uma linguagem de programação interpretada que possui tipagem dinâmica e possui coletor de lixo. Esta suporta paradigmas de programação como paradigma estrutural, orientado a objetos e programação funcional. Foi desenvolvida em 1991 por Guido van Rossum (ROSSUM, 2020).

A linguagem escolhida para este trabalho foi o Python, devido sua simplicidade e facilidade na realização de tal tarefa, uma vez que o algoritmo deverá ser rápido e não custoso computacionalmente.

2.2 XLSX

XLSX é um documento no formato ZIP que compacta conjunto de arquivos XML a fim de representar planilhas. Foi Desenvolvido pela Microsoft em 2006 e tem, como objetivo neste trabalho, coletar as informações de forma simplificada, com a nutricionista preenchendo-a no software Excel, onde se encontra pelo pacote Microsoft Office[2].

2.3 Openpyxl

A openpyxl[3] é uma biblioteca pública e gratuita feita em python que possibilita a leitura e escrita de arquivos xlsx (GAZONI; CLARK, 2022). Com ela, foi possível fazer a leitura da instância do problema.

[1] https://www.python.org/
[2] https://www.office.com/
[3] https://openpyxl.readthedocs.io/en/stable/

2.4 JSON

JavaScript Object Notation[4], em seu acrônimo, JSON, é um formato de arquivo de padrão aberto utilizando-se de um formato legível para humanos. Devido ao seu peso leve, possui objetivo de troca de informações entre sistemas.

É composto de objetos, utilizando-se do padrão chave-valor e de *arrays* utilizando chaves com espaçamento entre vírgulas. Desta forma torna-se simples a conversão do texto do arquivo para objetos em linguagens modernas.

[4] https://www.json.org/json-en.html

3 DESCRIÇÃO GERAL DO TEMA

A seção a seguir apresenta as descrições gerais do tema estudado.

3.1 Contexto universitário e a bolsa PROAT

A UFLA possui uma grande quantidade de alunos. Somente na graduação, a instituição ultrapassa 20 mil alunos (MENDES, 2021). Portanto, o cardápio do restaurante universitário deve ter variações para atender às múltiplas necessidades nutricionais dos frequentadores deste restaurante.

Esse problema fica mais claro se o cardápio for executado sem planejamento adequado para esse fim, dado que certas preparações podem, facilmente, ser mais frequentes que outras se não houver um parâmetro regulando isso. Normalmente, este planejamento se dá por meio de um nutricionista responsável por gerir e administrar o cardápio do restaurante. No entanto, tal tarefa se mostra complexa dentro do contexto universitário, dado que existem diversas restrições a serem consideradas no delineamento das preparações.

Nasce, portanto, a necessidade de um sistema de apoio à decisão para auxiliar o(a) nutricionista a fazer o planejamento diário dos cardápios para o restaurante. Tendo em vista esta demanda, o professor Mayron César de Oliveira Moreira, o professor Ramon Gomes Costa e a professora Andreza Cristina Beezão Moreira, com apoio da professora Ana Paula Piovesan Melchiori (na época, Pró-Reitora de Assuntos Estudantis e Comunitários) e das ex-nutricionistas do RU Emília Cristina Mões e Fernanda Cristina de Souza, empenharam-se na criação de um projeto acadêmico com bolsas estudantis para que alunos pudessem contribuir com essa tarefa.

Com sucesso, conseguiram a abertura de dois editais de bolsas de aprendizado técnico: um para a criação do método heurístico para a geração de cardápio, resultado deste trabalho, e outro para a implementação do sistema de banco de dados e sistema *web* que auxiliam e contribuem com o desenvolvimento do sistema de apoio.

3.2 Gerauca e os projetos anteriores

Para a obtenção, edição e exclusão das informações que contêm o cardápio do RU da UFLA, é necessário a colaboração dos nutricionistas responsáveis, uma vez que, hoje, realizam o planejamento de cardápios manualmente. Com esse fim, foi proposto paralelamente ao desenvolvimento

deste projeto a utilização de planilhas para podermos apoiar as soluções obtidas em uma solução próxima da realidade.

Porém, a utilização deste tipo de solução para a manutenção dos dados não é viável a longo prazo e portanto viu-se a necessidade da criação de uma *interface* gráfica amigável. Para este fim, em 2018, iniciou-se um projeto de criação de um sistema *web* por dois ex-bolsistas, responsáveis pela realização do início deste projeto, com dois ex-bolsistas responsáveis por estudos de identidade visual, *layout* de sistema e a construção inicial do banco de dados e as demais demandas solicitadas na época. Surgiu então o que seria o Gerauca, um acrônimo para Geração Automática de Cardápios e sua logo, como apresentado imagem da figura 3.1.

Figura 3.1 – Logotipo Gerauca

Fonte: Sousa (2022)

Com a evolução do projeto, foram necessárias as duas bolsas de aprendizado técnico extras em 2019 citadas anteriormente.

3.3 A implementação do banco de dados e sua relação com a heurística.

Pelo fato de não ser o foco principal do trabalho, apresenta-se apenas uma breve explicação de como foi modelado o banco de dados para ser possível explicar as restrições impostas para o problema e as decisões tomadas na implementação da heurística. Para isso, foi utilizado o diagrama entidade relacionamento conceitual de alto-nível.

O diagrama de entidade relacionamento conceitual de alto-nível é uma representação gráfica na qual um projeto pode, inicialmente, se sustentar para a melhor compreensão de um nível global do problema, sem enfoque em detalhes de implementação.

Para o trabalho em questão, realizado por Sousa (2022) em seu Trabalho de Conclusão de Curso (TCC), foram utilizadas as seguintes entidades e relacionamentos:

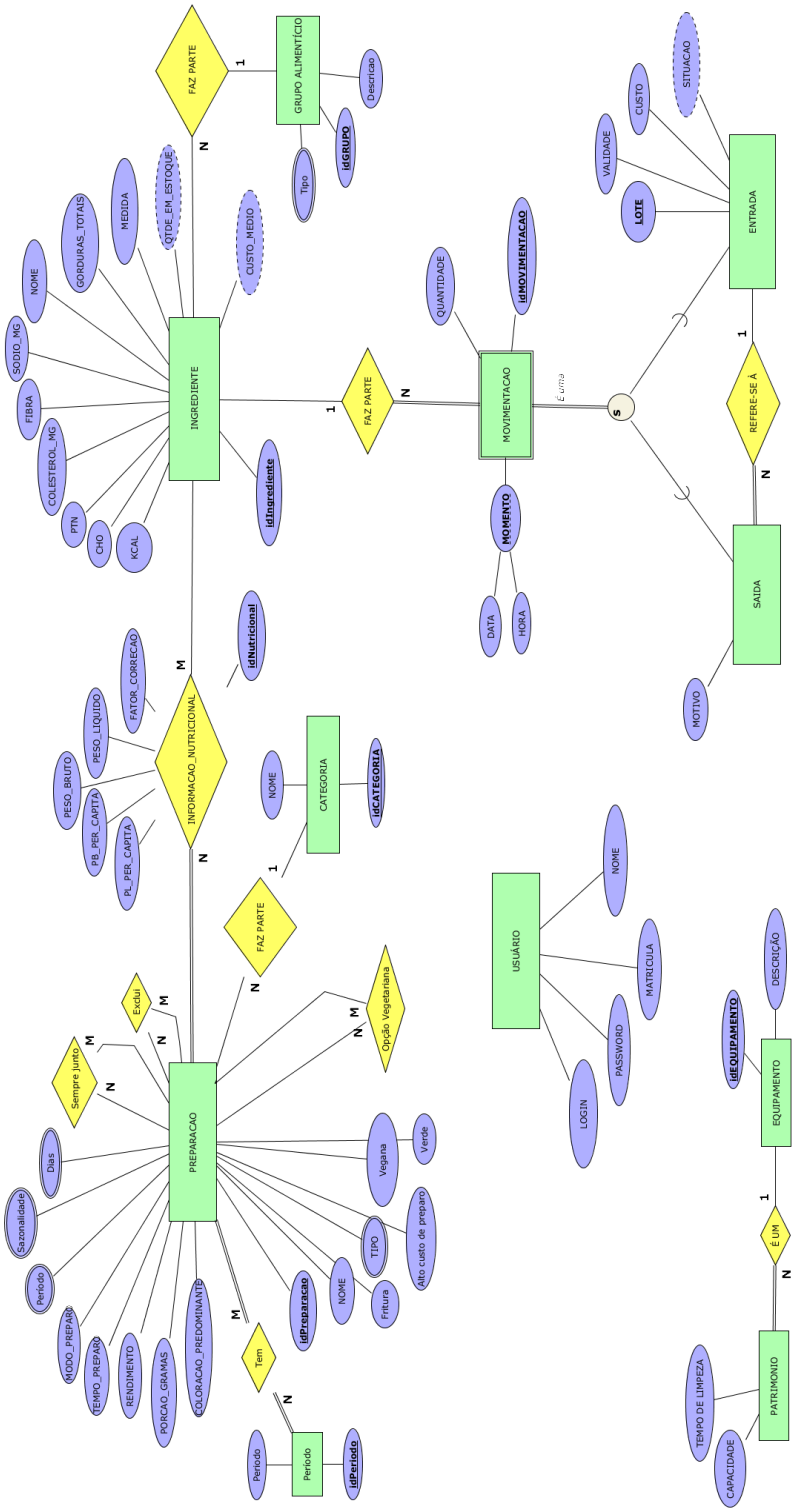
- Preparação faz parte de uma Categoria;

- Categoria faz parte de uma ou várias preparações;
- Preparação está Sempre Junta de nenhuma ou várias Preparações;
- Preparação Exclui nenhuma ou várias Preparações;
- Preparação é uma Opção Vegetariana de nenhuma ou de várias Preparações;
- Preparação tem nenhum ou vários Períodos;
- Períodos tem nenhum ou várias Preparações;
- Preparação possui nenhum ou vários Ingredientes;
- Ingrediente pertence a nenhuma ou várias preparações;
- Ingrediente faz parte de um Grupo Alimentício;
- Grupo Alimentício pertence a nenhum ou vários ingredientes;
- Ingrediente faz parte de nenhuma ou várias Movimentações;
- Movimentação pertence a um Ingrediente;
- Movimentação é uma Entrada ou Saída;
- Entidade Usuário;
- Equipamento é um Patrimônio;
- Patrimônio é nenhum ou vários Equipamentos.

Diante disso, as condições e restrições que a criação da heurística se baseou e que representa o contexto do restaurante universitário da UFLA, são:

- Restrição 1: Garante pelo menos 70% de utilização das preparações.
- Restrição 2: Uma mesma preparação não pode ser servida no almoço e no jantar.
- Restrição 3: Deve-se ter no mínimo uma e no máximo duas preparações proteicas por almoço e por jantar.

Figura 3.2 – Diagrama de Entidade Relacionamento



Fonte: Sousa (2022)

- Restrição 4: Deve-se ter duas saladas por almoço e por jantar.
- Restrição 5: Deve-se ter no máximo uma preparação (somente) vegetariana por almoço e por jantar.
- Restrição 6: Deve-se ter uma preparação vegana por almoço e por jantar.
- Restrição 7: Deve-se ter uma guarnição por almoço e por jantar. Há preparações proteicas que dispensam guarnição (por exemplo, escondidinho de carne seca).
- Restrição 8: Deve-se evitar, no mesmo cardápio, preparações que sejam excludentes em quesitos como uso de infraestrutura (espaço no forno; fritadeiras; número de formas), baixa aceitação pelos comensais, cor e textura.
- Restrição 9: Há preparações que aparecem sempre juntas em um cardápio (por exemplo, coxa/sobrecoxa de frango e creme de milho). Para uma preparação pode haver uma lista de possíveis opções para aparecerem juntas a esta.
- Restrição 10: 4 vezes carne de aves por semana.
- Restrição 11: 5 vezes carne bovina por semana.
- Restrição 12: 2 vezes carne suína por semana.
- Restrição 13: carne suína sempre com uma carne bovina ou de ave, como alternativa extra (caráter religioso).
- Restrição 14: 1 vez por semana deve ser servidos pescados ou pratos de alto custo ou preparação de difícil elaboração (panqueca, lasanha, galinhada, feijoada, etc).
- Restrição 15: 1 vez guarnição do tipo massa por semana.
- Restrição 16: 2 vezes caldos (polenta, canjiquinha, etc) por semana.
- Restrição 17: 1 vez guarnição seca (batata palha ou farofa) por semana.
- Restrição 18: 6 vezes guarnições do tipo refogado com legumes por semana.
- Restrição 19: 1 vez preparação do tipo fritura por semana.

- Restrição 20: As opções de salada podem ser compostas por 1 folha e 1 legume ou 2 folhas, mas nunca dois legumes, ou seja, proíbe dois legumes.
- Restrição 21: Preparações do mesmo grupo não devem aparecer no mesmo dia.
- Restrição 22: Não há refeições na janta do sábado e domingo.
- Restrição 23: Frutas não devem aparecer no almoço dos dias de semana.
- Restrição 24: Entre proteína e guarnição, não é permitido seco-seco.
- Restrição 25: Entre proteína e guarnição, não é permitido caldo-caldo.
- Restrição 26: Entre proteína e guarnição, não é permitido seca-massa.
- Restrição 27: Entre proteína e guarnição, não é permitido caldo-refogado.
- Restrição 28: Como forma de "forçar"uma boa variabilidade de cardápios, impôs-se que cada preparação proteica, guarnição e vegana apareçam no máximo uma vez por semana.
- Restrição 29: 14 preparações proteicas por semana.
- Restrição 30: 10 guarnições por semana.
- Restrição 31: no máximo 24 saladas por semana.
- Restrição 32: no máximo 1 fruta por semana.
- Restrição 33: 12 preparações veganas por semana.
- Restrição 34: no máximo 12 preparações vegetarianas por semana.

3.4 Heurística

Khan e Mir (2021) definem uma heurística como qualquer abordagem para solução de problemas, mesmo que seja uma abordagem auto-descoberta, que emprega um método prático que não é garantido como ótimo, mas ainda assim é suficiente para atingir uma meta ou aproximação da solução ótima, com uma complexidade algorítmica menor.

A construção da heurística é fundamentada no preceito de poder expandir o problema de modo facilitado e com menor complexidade.

3.5 Tipos de heurística

Existem alguns tipos de modo de construção de uma heurética, algumas delas são:

- Heurísticas de busca em vizinhança, com uma solução possivelmente viável ou não é buscado uma maneira de melhorar esta solução através de operações como remoção adição ou exclusão;
- Heurísticas sistemáticas: que visa utilizar uma arvore de soluções possíveis e o percorrer desta se dá por meio do que o problema pede. Tal como a estratégia *Backtracking*.
- Metaheurísticas: utiliza de escolhas previamente tomadas e que o algoritmo pode ou guardar em memória ou montar uma estrutura de aprendizado por si só.
- Heurísticas híbridas: visa unir um ou mais tipos de heurística a fim de solucionar um problema.

3.5.1 Algoritmo guloso

Um algoritmo guloso é uma maneira de pensar em um algoritmo que segue preceitos de realizar escolhas que são localmente ótimas, isto é, para cada interação do código selecionar a melhor opção possível dado as interações anteriores a fim de obter uma solução globalmente ótima.

Portanto, a heurística foi construída de modo construtivo por meio de um algoritmo guloso. "Um algoritmo guloso sempre faz a escolha que parece ser a melhor no momento. Isto é, ele faz uma escolha ótima para as condições locais, na esperança de que essa escolha leve a uma solução ótima para a situação global." (CORMEN, 2012).

Na heurística para o problema de geração de cardápios, entende-se a melhor escolha no momento, para um cardápio, por selecionar uma refeição dadas as condições das refeições anteriormente planejada. Por exemplo, se estivermos preparando refeição para o dia 10, devemos olhar nos últimos 10 dias quais foram as refeições servidas buscando otimizações de variação de cardápio e seguindo regras impostas para o problema. Como por exemplo, a inclusão de alternativa a carnes

de porco. Esta estratégia de seleção, possuindo somente como responsabilidade otimizar a escolha desta, reflete o que é para Cormen (2012), a escolha ótima para as condições locais, enquanto a situação global corresponde a geração de cardápios para um cronograma.

3.6 Justificativa

Para justificar a maneira como este trabalho foi feito, é necessária a compreensão da classe de problema que o mesmo pertence e para Seljak (2009) este problema, pertence à classe NP-difícil, dado que não se conhece algoritmos determinísticos que o resolvam em tempo polinomial.

Portanto nota-se que existem muitos estudos relacionados a geração de cardápios que utilizam-se de meios heurísticos. Metade dos estudos encontrados pela autora FARIA (2022) utilizam-se deste tipo de técnica.

4 IMPLEMENTAÇÃO

4.1 Base

Considerando um determinado dia, a heurística executa um laço de repetição para cada refeição e é escolhido, a princípio, 4 preparações para aquele cardápio, utilizando algoritmo guloso. Segue o pseudocódigo:

Código 4.1 – Pseudocódigo de base da heurística

```
1 para cada refeição do dia:
2       atualizar_atributos_referentes_a_refeição(refeição)
3       escolher(proteica)
4       escolher(guarnição)
5       escolher(salada)
6       escolher(vegetariano)
7 }
```

Fonte: Do Autor

No pseudocódigo, podemos ver que o *loop* principal da heurística é composto por duas partes: a atualização de atributos referente à refeição e à escolha de cada preparo, em ordem. Anterior à escolha de cada preparo, é necessário atualizar dados referentes a refeição. Armazena-se um identificador da refeição no planejamento de cardápios, e atualiza-se o dia e o período em que essa refeição será servida.

Posterior à atualização, escolhe-se, utilizando algoritmo guloso, o preparo de acordo com uma ordem de prioridade. Na implementação proposta, considerou-se como prioritários as preparações pertencentes aos seguintes grupos:

- proteica;
- guarnição;
- salada;
- vegetariano.

4.1.1 Escolhas das preparações

Para a escolha de cada preparação, são necessárias 3 etapas:

- Filtrar quais preparações não podem ser escolhidas;
- Selecionar a preparação que possui maior distância temporal entre a refeição atual e sua última escolha;
- Atualizar o cardápio final com a refeição escolhida e estruturas auxiliares que serão utilizadas para o cálculo da distância temporal.

4.1.2 Filtros

Os filtros são funções que delimitam as possíveis escolhas da instância, a fim de cumprir as restrições impostas pelo problema. Por exemplo, sempre deve ser servido *strogonoff* de carne e batata palha juntos. Portanto, caso o *strogonoff* de carne seja selecionado, deve-se eliminar todas as outras opções na escolha de guarnição que não sejam batata palha.

O inconveniente dessa estratégia é alcançar um nível de restrição de preparações a ponto de não restar nenhum preparo a escolher. Por isso, a cada filtragem, é verificado se ainda existe alguma preparação para o algoritmo escolher. Caso não existam preparações disponível, essa restrição é ignorada e o processo de filtragem continua. Segue o trecho de código:

Código 4.2 – Código de função para filtragem de refeições label

```
1 def filtrar(self, preparacoes, restricoes):
2     for restricao in restricoes:
3         temp = restricao(preparacoes)
4         if temp:
5             preparacoes = temp
6     return preparacoes
```

Fonte: Do Autor

No código 4.2, tem-se a definição e o corpo da função *filtrar*. Esta, possui dois parâmetros: uma lista de preparações possíveis e uma lista de funções que restringem essas preparações. Para

cada uma das restrições, a lista de preparações é passada como parâmetro e é feita a verificação mencionada anteriormente, para checagem de deleção total das preparações.

4.1.2.1 Implementação dos métodos de filtros

A implementação das restrições foi feita em uma classe nomeada de *Restrições*, onde cada método desta é uma função de filtragem de preparação. Apresenta-se cada um dos métodos a seguir:

- *restricaoSemJantaFds*: Restrição para limitar o oferecimento de jantar no final de semana. Não é utilizado na função *filtrar*, mas tem sua aplicação na função principal para esse fim;
- *restricaoSempreJuntos*: Restrição que filtra preparações de acordo com a restrição de sempre servir um conjunto de preparos juntos;
- *restricaoExcludentes*: Restrição que filtra preparações que exigem excludência de certos preparos;
- *excluirPorco*: Restrição que exclui preparações com a categoria igual à "*Porco*", utilizada para sempre ter alternativa a opção protéica de carne suína;
- *restricaoPerDia*: Restrição que exclui preparações que não possam aparecer em um determinado dia da semana. Também são excluídas preparações que não possam aparecer no período referente ao da refeição que está sendo escolhida;
- *restricaoVegana*: Restrição que exclui preparações vegetarianas que não são veganas;
- *restricaoQnt*: Generaliza restrições de quantidade de alimentos na semana por determinado atributo. A implementação de *restricaoQnt* é especificada em cada uma das seguintes funções:
  - *restricaoFritura*: filtrar por quantidade de preparações que são "fritura";
  - *restricaoDispensaGuarnicao*: Filtrar por quantidade de preparações protéicas que dispensam guarnição (disp_guarnicao);
  - *restricaoBovina*: Filtrar por quantidade de preparações protéicas de categoria "Bovina";

- *restricaoPescado*: Filtrar por quantidade de preparações protéicas de categoria “Pescado”;
- *restricaoPorco*: Filtrar por quantidade de preparações protéicas de categoria “Porco”;
- *restricaoAves*: Filtrar por quantidade de preparações protéicas de categoria “Aves”;
- *restricaoMassa*: Filtrar por quantidade de preparações de categoria “Massa”;
- *restricaoCaldo*: Filtrar por quantidade de preparações de categoria “Caldo”;
- *restricaoSeca*: Filtrar por quantidade preparações de categoria “Seca”;
- *restricaoRefogado*: Filtrar por quantidade de preparações de categoria “Refogado”.

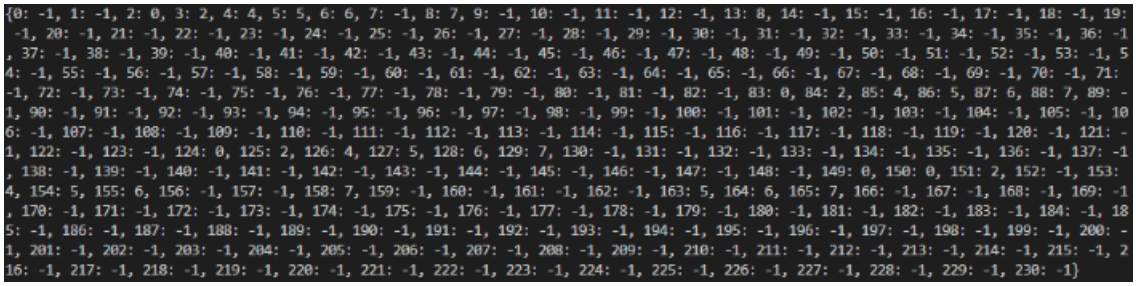
A classe Heurística herda a classe *Restricao*, tendo, então, acesso simples a qualquer uma das funções de restrição.

4.1.3 Preparação mais distante servida

A escolha da preparação mais distante é facilitada ao manter uma estrutura que foi chamada de *ultimaRefeicaoServido*. Essa estrutura é um dicionário em que a chave é o identificador da preparação e o valor é o identificador da última refeição que a preparação foi servida. A estrutura é inicializada com todas as chaves, do 0 até a quantidade de preparações possíveis da instância. O valor “-1” indica que a preparação não foi selecionada ainda.

Para escolher a preparação com maior distância entre a refeição atual e a sua última refeição servida, basta encontrar o menor valor do dicionário *ultimaRefeicaoServido*, e mantê-lo atualizado. Um exemplo desta estrutura é apresentada na figura 4.1.

Figura 4.1 – Exemplo de um dicionário *ultimaRefeicaoServido* no meio de uma excussão.



Fonte: Do Autor

4.1.4 Atualizar

No processo de atualização, três estruturas diferentes são alteradas: *(i)* a estrutura *qnt*, utilizada por alguns filtros; *(ii)* o dicionário *ultimaRefeicaoServido*, utilizado para escolher a preparação mais distante; por fim, *(iii)* a estrutura que armazena o planejamento final dos cardápios.

4.2 Arquivo xlsSave

Dado um resultado da heurística e o objeto da instância, é possível utilizar a função *printResultOnXls* para imprimir este resultado em um arquivo xlsx.

5 ESTUDO DE CASO

O estudo de caso consiste na implementação da heurística e a sua integração com o Gerauca, e certificar-se de que o mesmo esteja conciso com as exigências até aqui mostradas. Ainda será mostrada como foi possível a integração da planilha XLSX e sua conversão para as informações no banco.

5.1 Atividades realizadas

As atividades realizadas durante o período do programa de aprendizado técnico possuem vigência do dia de 14 de fevereiro de 2020 a 03 de novembro de 2020, totalizando nove meses aproximadamente.

A bolsa institucional do edital da PROAT (Programa de Aprendizado Técnico) possui vigência de um ano. Contudo, o autor do trabalho foi convocado como suplente após alguns meses do início da bolsa em razão da desistência do então bolsista. Além disso, a bolsa encerrou antes do previsto em virtude da impossibilidade do autor prosseguir com a bolsa devido a ingressão do mesmo em um regime CLT.

A conclusão deste trabalho foi realizada em virtude das exigências de conclusão do curso em bacharelado de Ciência da Computação. Todas as atividades, somadas ao planejamento e escrita deste relatório técnico, ultrapassam 510 horas.

5.1.1 Fevereiro de 2020

No primeiro mês foram realizados estudos e reuniões para a integração do mesmo no tema e no projeto, que já havia se iniciado.

5.1.2 Março de 2020

Foram implementadas as classes principais que modelam o problema em memória a tempo de execução e a leitura da planilha em XLSX para instâncias dessas.

5.1.3 Abril de 2020

Foram implementados métodos e funções que auxiliam a visualização das instâncias e dos resultados que a heurística gera.

5.1.4 Maio de 2020

Implementação inicial da heurística pelo modelo de distâncias entre as refeições.

5.1.5 Junho de 2020

Análises dos resultados com reuniões entre o autor, nutricionista e orientador. Além disso, foram implementadas melhorias e boas práticas de programação no código. Como por exemplo a modularização e otimização de algumas funções base da heurística.

5.1.6 Julho de 2020

Adaptações do código para a implementação da filtragem de preparações. Foram realizadas em virtude dos resultados obtidos no mês anterior.

5.1.7 Agosto de 2020

Foram realizados estudos de possíveis melhorias e novas ferramentas, como por exemplo o *hyperopt*. Além disso, deu-se início à escrita do relatório exigido pela PROAT.

5.1.8 Setembro de 2020

Implementação de uma interface gráfica para a heurística e facilitação na visualização do projeto.

5.1.9 Outubro de 2020

Correções gerais de erros no código da heurística e reuniões de finalização do projeto.

5.1.10 Junho de 2022

Reuniões de retomada do projeto e análise das alterações do mesmo.

5.1.11 Julho de 2022

Adaptação das restrições na heurística, implementação de exportação dos resultados da heurística em arquivo JSON. Adicionalmente, foram feitas correções gerais no Gerauca e implementação de exportação dos dados em JSON.

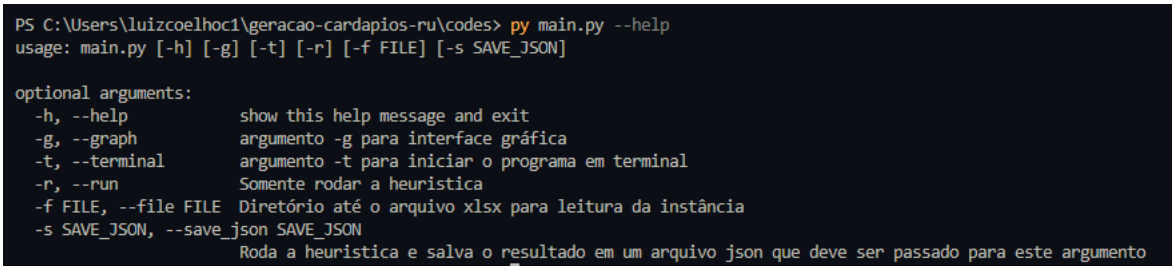
5.1.12 Agosto de 2022

Transformação dos dados, que anteriormente eram lidos em XLSX, para *inserts* no banco de dados.

5.2 Heurística

O sistema da heurística é acionado via *python* pelo arquivo *"main.py"*, e possui os seguintes parâmetros: -h, -g, -t, -r, -f e -s. Esses parâmetros com suas respectivas descrições podem ser vistos utilizando o parâmetro –help, como no exemplo da Figura 5.1.

Figura 5.1 – Exemplo de como utilizar o sistema heurístico



Fonte: Do Autor

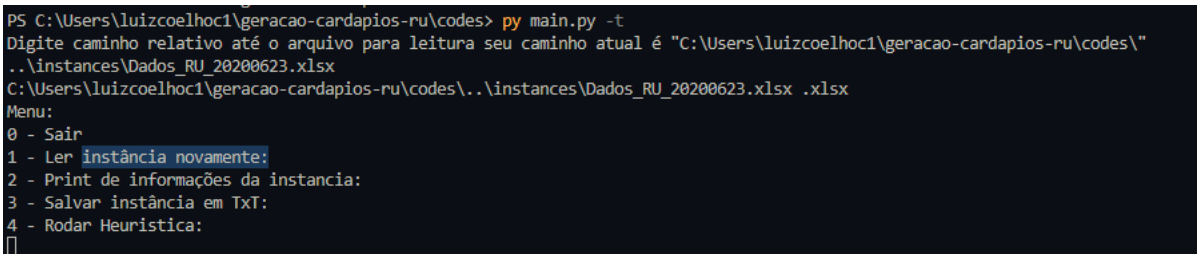
É possível a utilização do sistema heurístico de três formas principais e é necessário a compreensão de como utilizar o terminal para rodar programas Python:

- pelo terminal, utilizando o parâmetro -t;
- gerando e salvando os resultados em um arquivo JSON, utilizando o parâmetro -s;
- pela interface gráfica, utilizando o parâmetro -g.

5.2.1 Interface via terminal

Caso não seja passado o parâmetro -f com o caminho para arquivo de leitura, o programa obrigará o usuário a digitar este caminho para a leitura da instância logo no início. Após esse procedimento, é possível navegar entre os menus listados digitando o número correspondente do menu atual no qual o usuário se encontra. Por exemplo, na Figura 5.2, no menu inicial, é possível sair digitando 0. Também é possível digitar 2, por exemplo, para entrar em um outro submenu de opções para impressão na tela dos dados referentes à instância lida.

Figura 5.2 – Menu principal do sistema heurístico pelo terminal



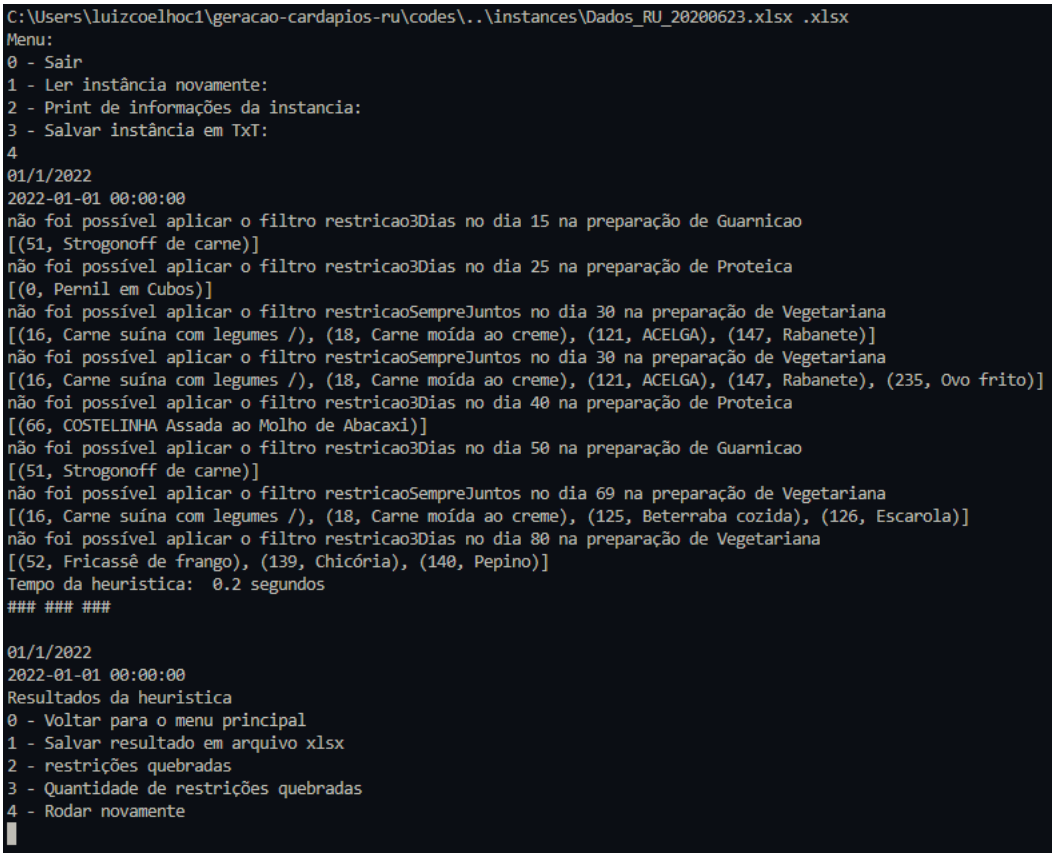
Fonte: Do Autor

Ainda na Figura 5.2, pode-se realizar a chamada da heurística na instância já lida pelo programa, digitando 4 no menu inicial. Após a realização da função heurística, o sistema apresentará o dia inicial, quais restrições possivelmente foram violadas, devido ao caráter heurístico e o tempo gasto dela heurística. Além disso, o programa mostra, no submenu de resultado da heurística, onde é possível voltar ao menu principal, salvar o resultado em um arquivo *".xlsx"*, e imprimir na tela novamente quais as restrições que não foram respeitadas. É possível, ainda, visualizar a quantidade de restrições não respeitadas e executar novamente a heurística, como apresentado na Figura 5.3. A segunda excussão é utilizada para caso altere o *input* do problema, não possuindo nenhum componente estocástico.

5.2.2 Salvar os resultados em um arquivo JSON

Para esta forma de utilização, é necessário os parâmetros -r, -f e -s. O -r indica para o sistema que não é desejado outras formas de interação com o programa que não a execução a heurística. O parâmetro -f é necessário para indicar que o arquivo de instância é utilizado, uma vez que não há interação com o usuário para perguntar qual é este arquivo. Já o parâmetro -s é necessário para

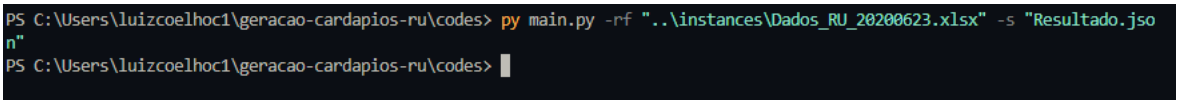
Figura 5.3 – Sistema Heurístico gerando cardápio via terminal



Fonte: Do Autor

obter os resultados da heurística em um arquivo JSON. Um exemplo de utilização é possível de ser visto na Figura 5.4.

Figura 5.4 – Sistema Heurístico gerando cardápio silenciosamente



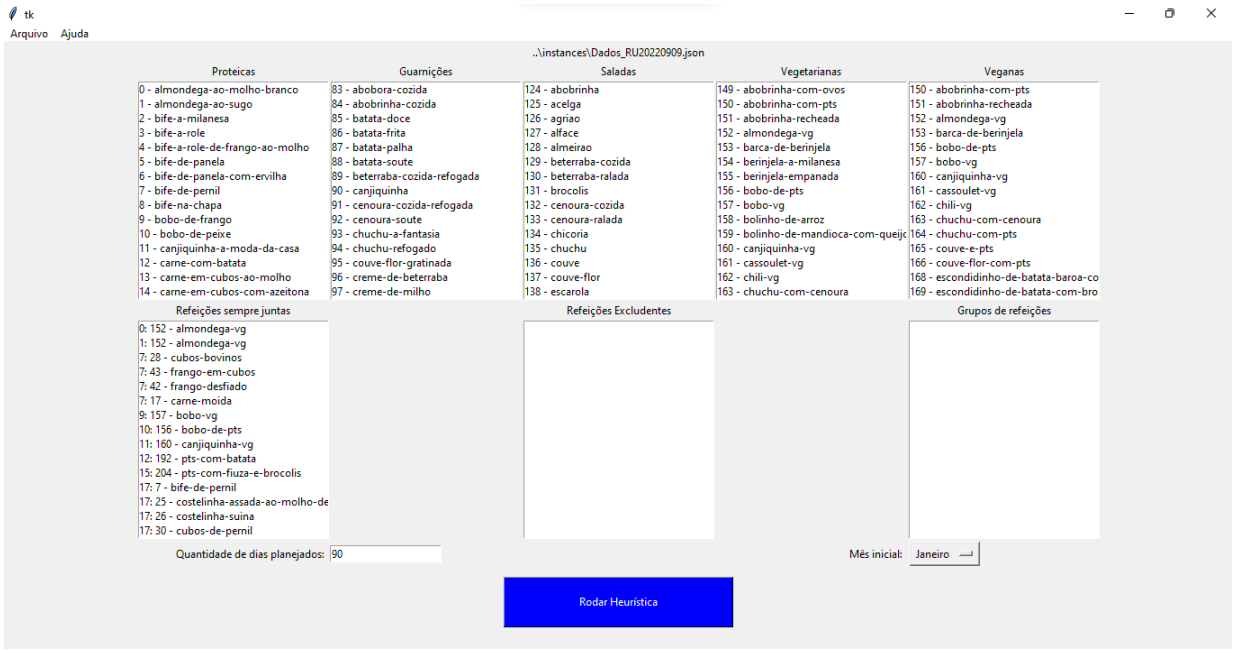
Fonte: Do Autor

5.2.3 Interface gráfica

Para utilizar a interface gráfica, deve-se substituir o parâmetro -t para o -g. Contudo, também é necessário digitar o local para o arquivo de instância, seja pelo parâmetro -f ou após a inicialização do programa.

Na tela de abertura do programa, que se encontra na Figura 5.5, é possível visualizar a instância lida e dois parâmetros de entrada para configuração de quantos dias e quais dias o cardápio é gerado. Ainda nesta tela, é possível localizar um botão, que tem como funcionalidade a geração do cardápio via heurística.

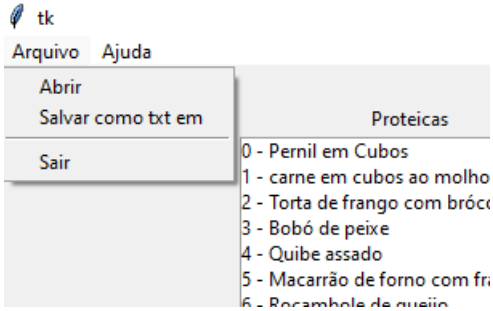
Figura 5.5 – Menu principal do sistema heurístico via interface gráfica.



Fonte: Do Autor

É possível, ainda, ler e salvar novas instâncias pelo menu superior esquerdo como apresentado na Figura 5.6.

Figura 5.6 – Menu superior do sistema heurístico via interface gráfica

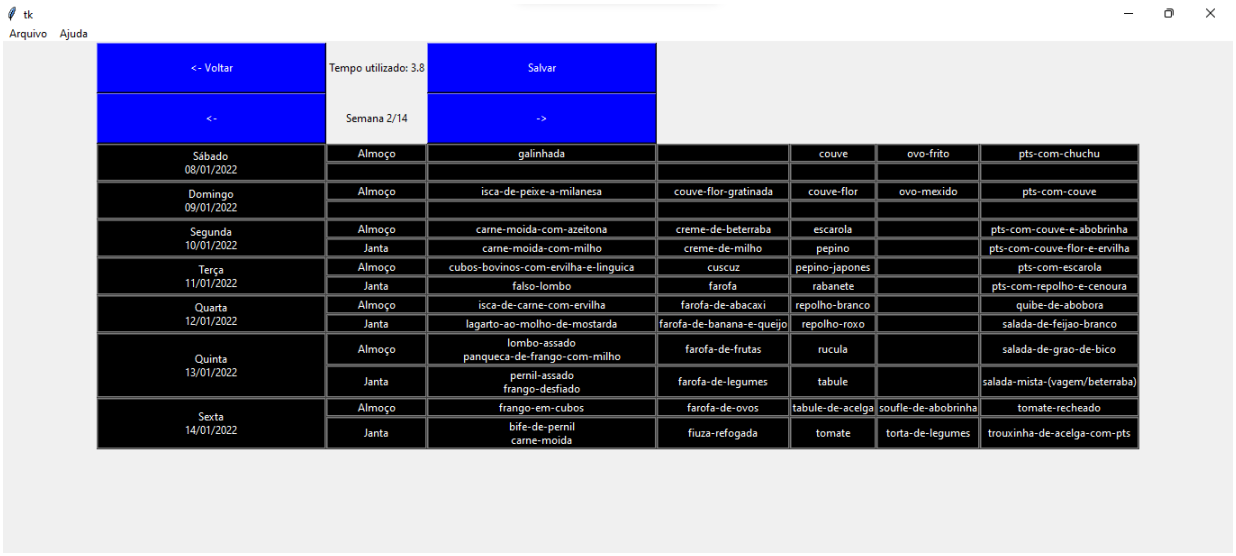


Fonte: Do Autor

Ao clicar no botão *"Rodar Heurística"*, será gerado um novo planejamento de cardápio por meio da heurística e irá para uma nova tela de visualização do cardápio gerado. (Ver a Figura

5.7). Nessa tela é possível a visualização do planejamento no formato de planilhas separado por

Figura 5.7 – Tela de resultados da geração de cardápio via interface gráfica no sistema heurístico.



Fonte: Do Autor

semanas. Para a navegação entre as semanas, utiliza-se os dois botões localizados acima da tabela com os caracteres “<-” indicando a requisição das preparações na semana anterior a qual o sistema mostra, e os caracteres “->” indicando a requisição das preparações na semana posterior a qual o sistema mostra.

Ainda, é possível salvar os resultados em um arquivo *“.xlsx”*, clicando no botão salvar.

5.3 Gerauca

O menu no canto superior esquerdo (Figura 5.8) do sistema dá acesso ao usuário a algumas páginas: *homepage*, página de contatos, ingredientes, preparações, exportar e geradores. O menu *"Geradores"* foi adicionado na alteração com as modificações relativas a este trabalho.

Figura 5.8 – Menu modificado no Gerauca.

Fonte: Do Autor

Na página de exportação (Figura 5.9) foi adicionado um novo botão cuja função é a exportação dos dados no formato *".json"*.

Foi adicionado em seu *layout* a página de geradores (Figura 5.10). Nela, é possível gerar um arquivo *".json"* pelo modelo heurístico e abre a possibilidade para a inclusão do modelo matemático da mesma forma que o método heurístico foi adicionado.

Figura 5.9 – Tela de exportação com novo botão para a exportação em *".json"*.

Fonte: Do Autor

Figura 5.10 – Tela de geradores para a geração de cardápio, seja de modo heurístico, seja com o modelo matemático.

Fonte: Do Autor

6 CONCLUSÃO

O discente pôde participar da criação de uma solução para um problema real resolvido através de algoritmos de otimização. Após os os desafios enfrentados, envolvendo as dificuldades que cercam a elaboração de uma solução eficiente, conclui-se que uma solução bem sucedida nesse contexto envolve a integração de metodologia adequada e conhecimento dos especialistas que vão operar o sistema.

A criação da heurística envolve outros projetos no qual o discente pôde incluir alguns desses em uma única plataforma. Contudo o foco principal do mesmo foi no desenvolvimento do algoritmo heurístico para a solução do problema de geração automática de cardápios.

Enfrentando dificuldades no planejamento em virtude da entrada tardia do discente no projeto e tendo a responsabilidade de integrar muitas partes do projeto em um.

Portanto testes mais exaustivos no Gerauca deverão ser feitos para tornar a solução escalável durante todo o semestre letivo da UFLA. Além disso deverá ser feito melhores visualizações das informações geradas pelo Gerauca.

Mesmo diante disto o discente conseguiu concluir o projeto com a ajuda do professor Mayron César de Oliveira Moreira, o professor Ramon Gomes Costa e a professora Andreza Cristina Beezão Moreira, com apoio da professora Ana Paula Piovesan Melchiori e das ex-nutricionistas do RU Emília Cristina Mões e Fernanda Cristina de Souza. A ajuda de Gustavo Rodrigues Sousa também foi de grande relevância para este trabalho.

A UFLA, também, foi suma importância para o acontecimento deste trabalho uma vez que a mesma proporcionou ao discente conhecimento suficiente para o discente conseguir desenvolver ou pesquisar o necessário. Disciplinas como Introdução aos Algoritmos, Estruturas de Dados, Paradigmas de Linguagens de Programação, Algoritmos em Grafos, Práticas de Programação Orientada a Objetos, Sistemas Gerenciadores de Banco de Dados, Metodologia de Pesquisa, Programação Matemática e Sistemas Distribuídos possui co-relação e portanto contribuíram para a realização do mesmo.

REFERÊNCIAS

CORMEN, T. **Algoritmos: teoria e prática**. [S.l.]: GEN LTC; 3ª edição, 2012. ISBN 978-8535236996.

FARIA, Y. R. D. Problema de planejamento de geração de cardápios: Uma revisão sistemática da literatura. 2022.

GAZONI, E.; CLARK, C. **openpyxl - A Python library to read/write Excel 2010 xlsx/xlsm files — openpyxl 3.0.10 documentation**. 2022. (Acessado em 26/08/2022). Disponível em: <https://openpyxl.readthedocs.io/en/stable/>.

KHAN, A.; MIR, M. Heuristic. 12 2021.

MENDES, G. **UFLA ultrapassa 20 mil estudantes formados na graduação.** 2021. (Acessado em 26/08/2022). Disponível em: <https://ufla.br/noticias/ensino/14642-ufla-ultrapassa-a-marca-de-20-mil-estudantes-formados-na-graduacao>.

ROSSUM, G. van. **Python 0.9.1 part 01/21**. 2020. Disponível em: <https://www.tuhs.org/Usenet/alt.sources/1991-February/001749.html>.

SELJAK, B. K. Computer-based dietary menu planning. p. 414–420, 2009.

SOUSA, G. R. Criação do sistema de banco de dados como apoio para cardápios automatizados de restaurantes universitários. 2022.