USC

Documento de arquitetura de software

Vote na Boia

Versão 01.00 1 de junho de 2020

Responsáveis: Bianca Oliveira, Hellen Gouveia, Hugo Cavallari, Jonas Jaciel, Marli Terada, Otávio Jacobini

Histórico de revisões

Data	Versão	Descrição	Autor	Revisado por
11/06/2020	1.0	Versão inicial da documentação	Hellen	Marli
14/06/2020	1.1	Adicionados diagramas de casos de uso	Todos	Hellen/Bianca
15/06/2020	1.2	Especificação dos casos de usos	Bianca	Hellen
20/06/2020	1.3	Especificação das camadas	Bianca	Hugo
29/06/2020	1.4	Introdução e objetivo	Hellen	Otávio
04/07/2020	2.0	Finalização da documentação	Hellen	Bianca

SUMÁRIO

1 Introdução

Este documento contém as especificações e diagramas dos requisitos levantados do sistema 'VoteNaBoia'. Destinado aos projetistas e desenvolvedores como forma de auxiliar no desenvolvimento do sistema. Além dos requisitos e diagramas, este documento detalha a arquitetura utilizada para a implementação da API e da aplicação que realizará as requisições, bem como as tecnologias utilizadas, e propostas de melhorias para as próximas versões.

2 Objetivo

Além da API, desenvolvida utilizando a arquitetura em camadas, conforme orientação do professor, será desenvolvida uma aplicação web de forma a poder ser acessada, tanto pelo computador/notebook, quanto por dispositivos móveis. Por essa facilidade de disponibilizar uma em vários dispositivos é que optamos por utilizar o PHP combinado ao Laravel.

3 Requisitos e restrições arquiteturais

Requisito	Solução
Linguagem	Para o desenvolvimento da API, será utilizada a linguagem C# .Net A aplicação web será desenvolvida em PHP, utilizando o framework Laravel, combinado com Html, JavaScript e CSS
Plataforma	A API e a aplicação web serão disponibilizadas na plataforma AWS
Segurança	A API fará uso do padrão JWT para validar as requisições recebidas da aplicação web.
Persistência	O banco de dados utilizado será o MS SQL Server
Internacionalização (i18n)	A princípio, o aplicativo e a página web serão disponibilizados somente em Português-Br.

4 Visão Geral

Para o desenvolvimento da API, será utilizada a arquitetura em camadas, as quais estão ilustradas na imagem a seguir:

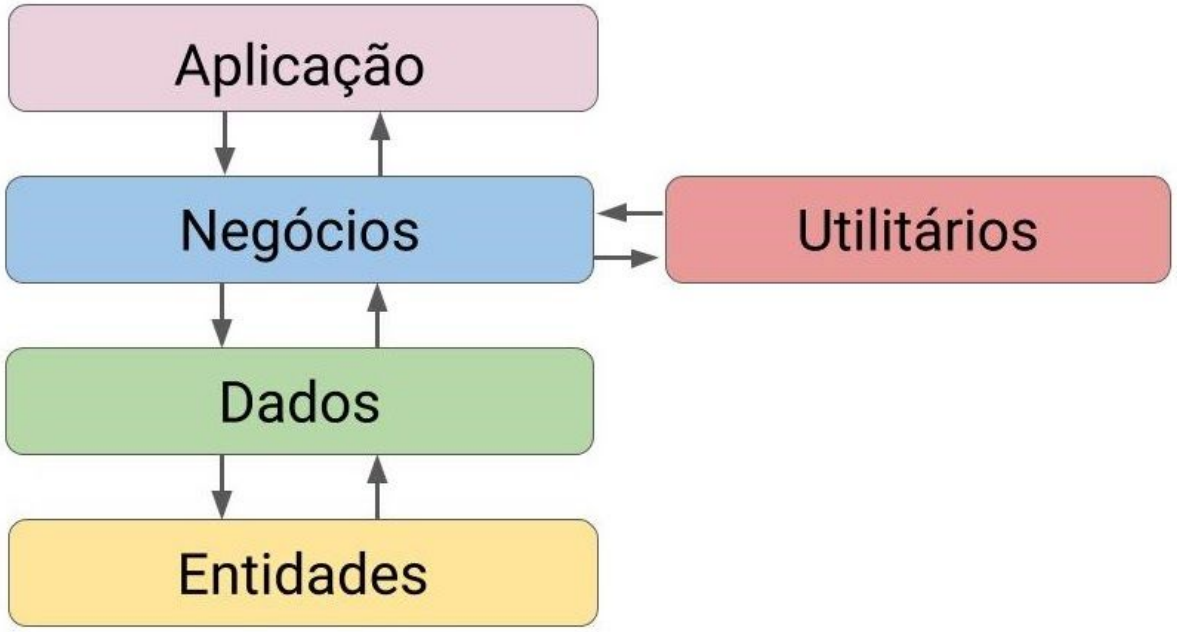


Figura 1 - Camadas do sistema.

4.1 Visão de caso de uso

Tabela 1 - Listagem de casos de uso.

Caso de uso	Descrição
CSU1	Cadastrar Aluno
CSU2	Cadastrar Restaurante
CSU3	Votar Restaurantes Semanal
CSU4	Solicitar Resultado Restaurantes Semanal
CSU5	Votar Restaurante Diário
CSU6	Solicitar Resultado Restaurante Diário
CSU7	Realizar Login

4.1.1 Casos de uso

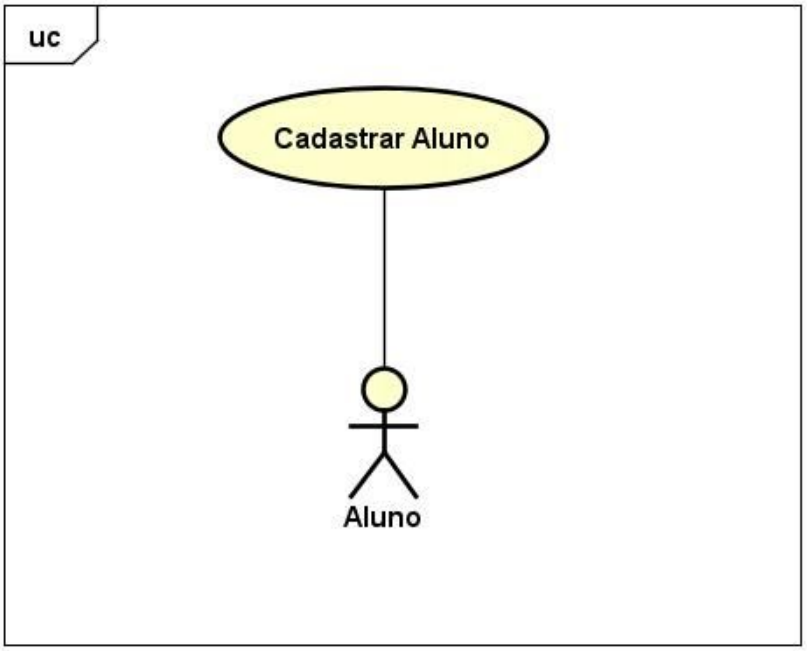


Figura 1 - CSU1 - Cadastrar Aluno

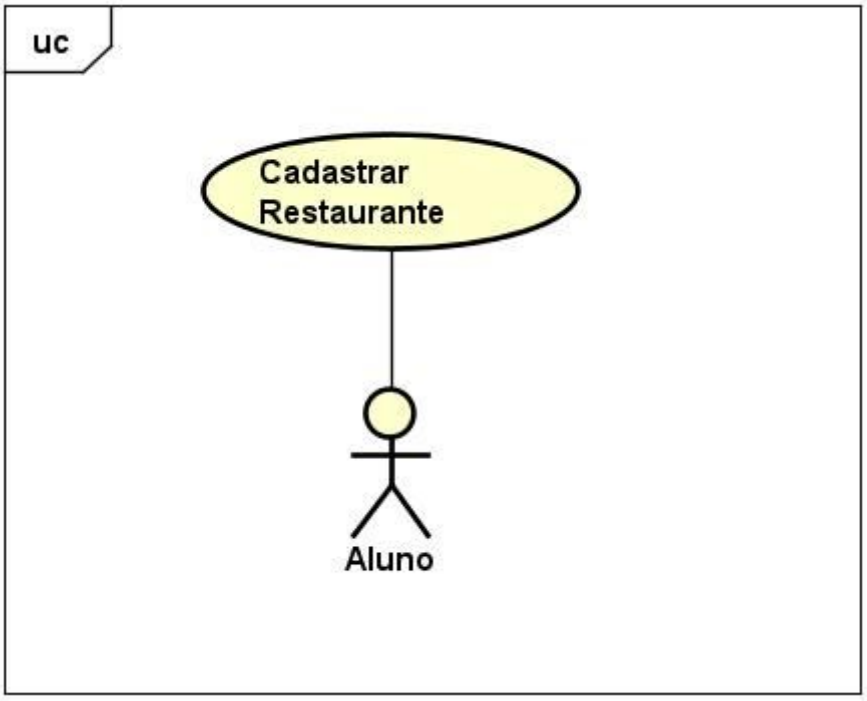


Figura 2 - CSU2 - Cadastrar Restaurante

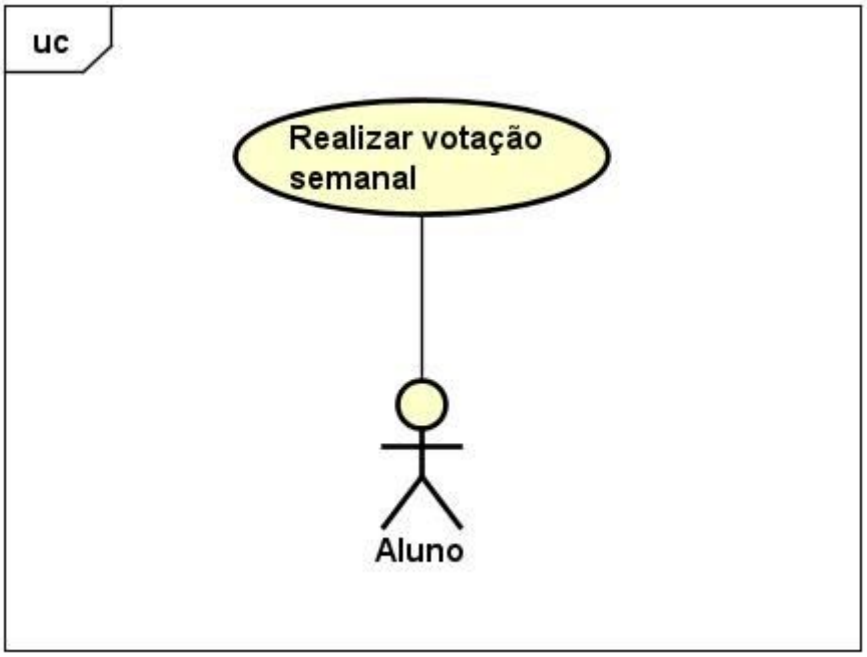


Figura 3 - CSU3 - Votar Restaurante Semanal

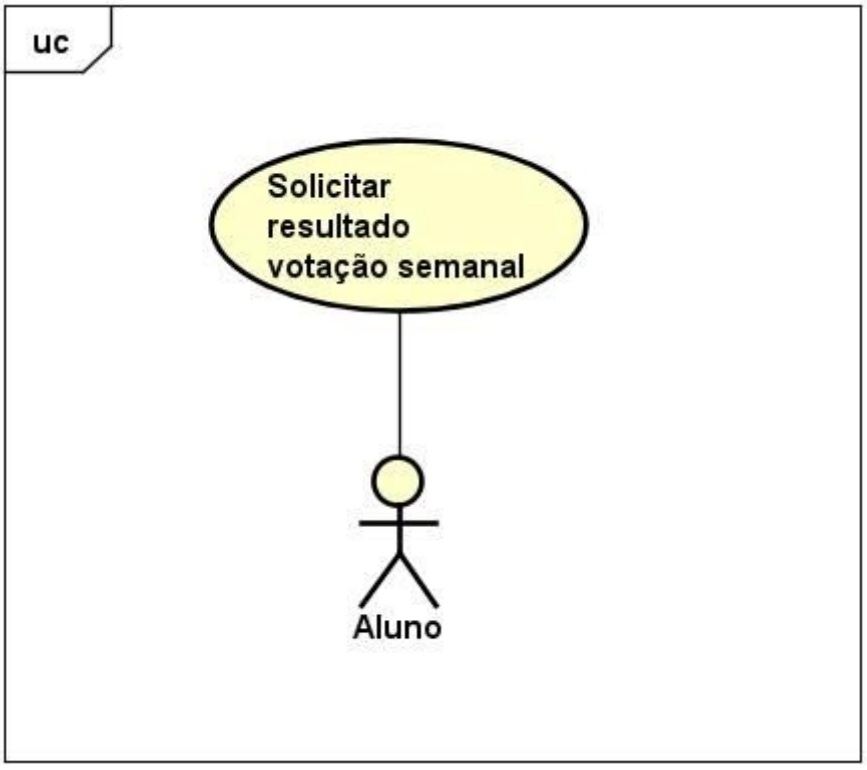


Figura 4 - CSU4 - Solicitar Resultado Semanal

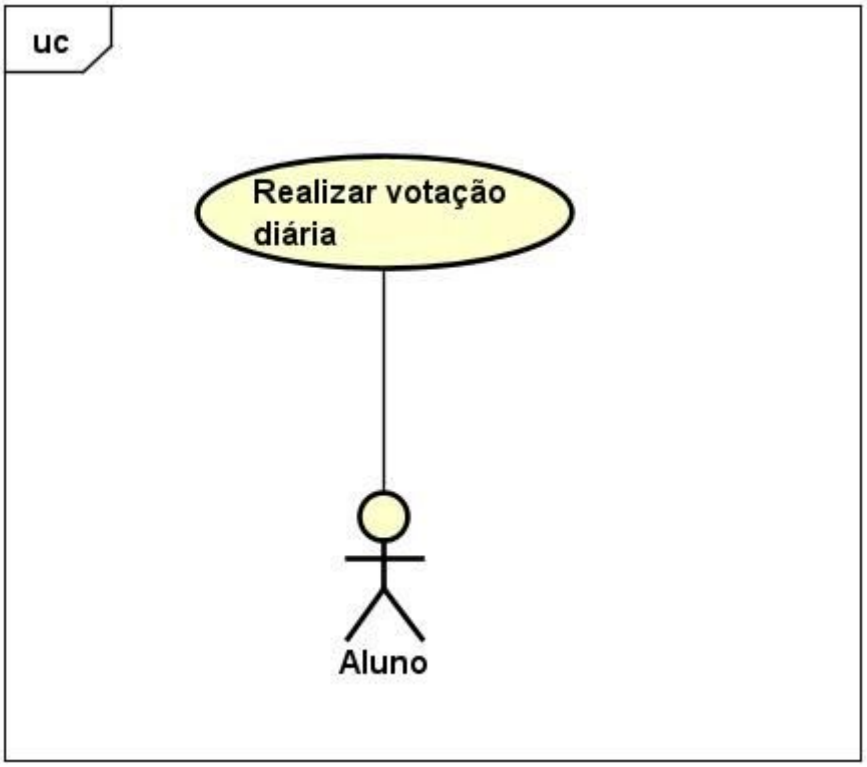


Figura 5 - CSU5 - Realizar Votação Diária

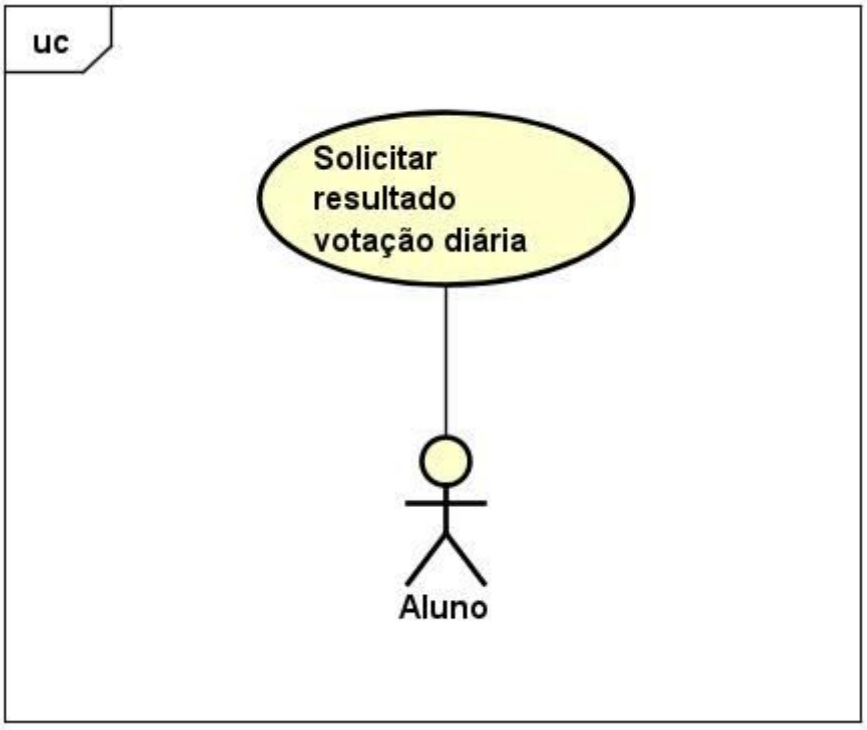


Figura 6 - CSU6 - Solicitar Resultado Diário

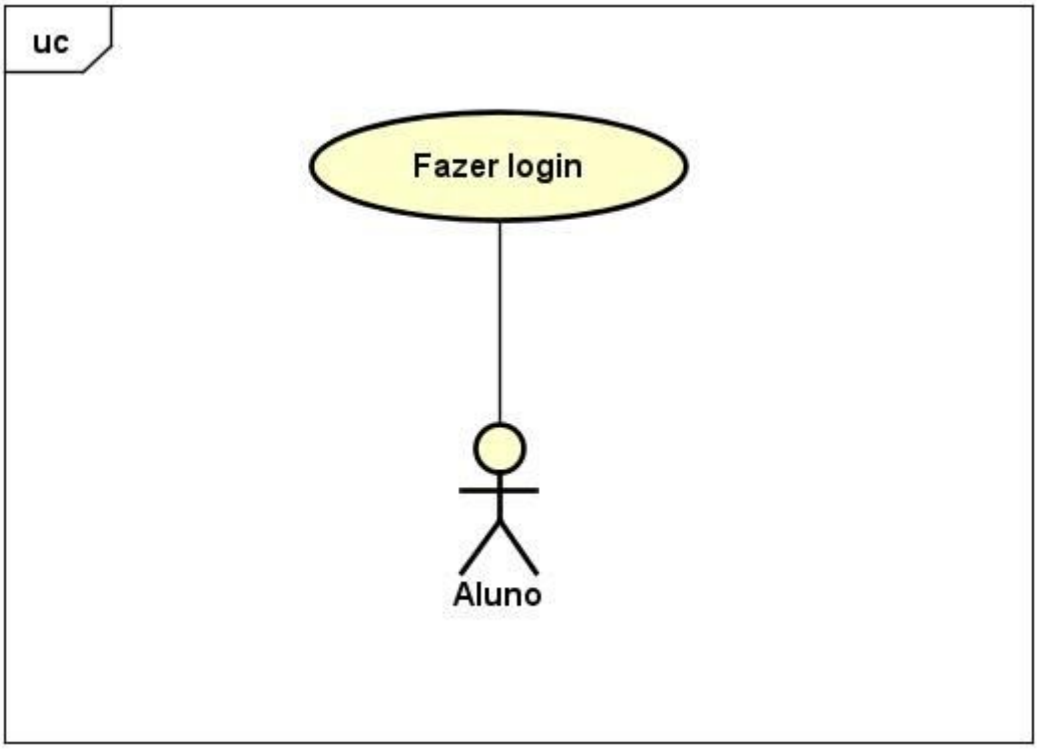


Figura 7 - CSU7 - Realizar Login

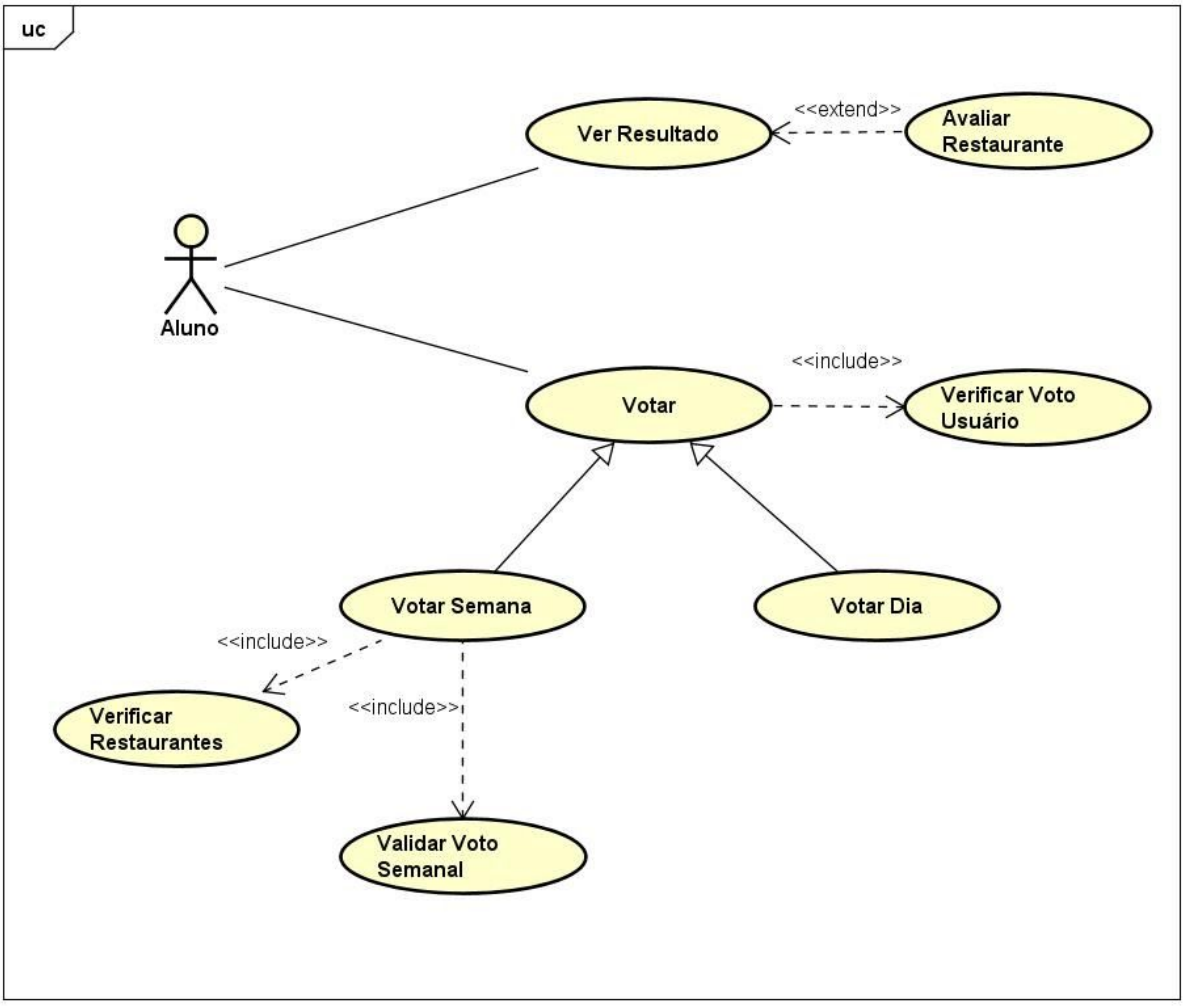


Figura 8 - CSU Votar (Visão completa)

4.2 Visão de implantação

O sistema será desenvolvido utilizando a linguagem de programação C# .Net Core, voltando seu foco para todos os tipos de dispositivos, já que sua arquitetura é baseada na utilização de api, proporcionando a disponibilização de funções ao meio externo.

Combinado à essa tecnologia, será utilizado o Entity Framework, o qual possibilitará o mapeamento do banco de dados da aplicação.

Para a implementação da aplicação web, responsável por consumir a api, utilizaremos a linguagem PHP combinada com o Laravel Framework.

4.3 Visão de implementação

Seguindo o padrão arquitetural de camadas, o sistema foi dividido em cinco camadas, sendo elas Camada de Aplicação, Camada de Negócios, Camada de Dados, Camada de Entidades e Camada de Utilitários. Essas camadas são separadas de acordo com sua influência e responsabilidade no sistema.

A primeira camada, de aplicação, é a camada que receberá as requisições externas, através de rotas (endpoints) programadas. Cada rota indicará a operação que deverá ser requisitada às camadas seguintes.

Ao receber uma requisição, o controller, presente na camada de aplicação, fará a requisição à camada de negócios, na qual estarão as regras de negócios, regras responsáveis por validar a inserção e alteração de dados, bem como, responsáveis por devolver informações de obtidas da terceira camada, camada de dados.

Na camada de dados ficarão os repositórios, responsáveis por persistir e consultar dados do banco de dados.

Para obter os dados da base de dados é necessário realizar o mapeamento das tabelas. Esse mapeamento ficará na camada de entidades, na qual, além das entidades, estarão os DTOs, que são os objetos responsáveis pela transferência de dados, ou seja, é através do DTO que os dados são passados ou obtidos das entidades.

Por fim, na camada de utilitários ficarão as classes utilizadas em todo o projeto. Por exemplo, validação de e-mail.

4.3.1 Camadas

4.3.1.1 Camada 1 - API

A camada de API é a camada de aplicação, na qual estarão os ‘endpoints’, é a camada que possibilita a disponibilização de funções à aplicações externas.
Os controllers dessa camada serão responsáveis por solicitar ou enviar dados para as outras camadas.

4.3.1.2 Camada 2 - BLL

A camada BLL é a camada de negócios, nela estarão as regras de negócio para não permitir que o mesmo usuário vote duas vezes no dia, não permitir que um restaurante já escolhido na semana seja escolhido novamente e que a votação seja encerrada no horário estabelecido. Também conterá a regra que eliminará os dois restaurantes mais votados na semana anterior da votação de escolha dos restaurantes da próxima semana.

4.3.1.3 Camada 3 - DAL

Essa é a camada responsável pela persistência dos dados no banco de dados.
É nessa camada que ficarão os repositórios, responsáveis pelas consultas e comandos de inserção de dados.

4.3.1.4 Camada 4 - Entities

Nessa camada ficarão os mapeamentos das tabelas do banco de dados.

4.3.1.5 Camada 5 - Helpers

Helpers são os utilitários usados nas validações das regras de negócios, que são reaproveitados para todas as classes da camada BLL.

5 Considerações para próximas versões

Como melhorias para versões futuras, será implementada a possibilidade de avaliar restaurantes visitados, onde essa avaliação atuará como um critério de desempate quando este ocorrer.
Serão disponibilizados também resultados como: restaurante mais votado no mês, restaurante mais votado por usuário e restaurante melhor avaliado.
Também serão acrescentadas notificações por e-mail, quando desejadas, quanto ao início e término das votações, tanto semanal, quanto diária, bem como dos resultados de cada uma.
Cogita-se também uma possível parceria com os restaurantes, onde o estabelecimento poderá receber uma notificação de que foi o escolhido do dia e a quantidade de clientes esperados. Dessa forma, poderá organizar o espaço para melhor acomodar os visitantes.
E a melhoria mais almejada é o desenvolvimento de um aplicativo nativo para smartphone, através do qual será possível disparar notificações push.

6 Anexos

6.1 Análise de requisitos

RF1. O sistema deverá permitir o cadastro de usuário, no caso o aluno do curso de pós-graduação, para que o mesmo esteja apto a realizar as votações.

RF2. Para realizar o cadastro do usuário, será disponibilizado um código de validação, através do qual será possível distinguir a qual turma o usuário pertence.

RF3. Após o envio dos dados, o cadastro do usuário deverá passar por uma moderação, que realizará a aprovação ou não do mesmo, como forma de garantir que somente alunos da turma de pós-graduação sejam cadastrados.

RF4. No cadastro do usuário deverá conter a configuração se o mesmo deseja ser notificado por e-mail quanto aos resultados das votações. Também deverão ter as informações de Nome, Email, Senha, Situação (ativo/inativo), Aprovação (aprovado/ não aprovado).

RF5. O sistema deverá permitir o cadastro de restaurantes que participarão das votações semanais e diárias. Nesse cadastro deverão ter os dados: Razão Social, Endereço, Tipo (churrascaria/fastfood/vegano/vegetariano,etc), Formas de Pagamento aceitas (dinheiro, cartão crédito/débito, picpay, etc), Avaliação, E-mail, Telefone, Situação (ativo/inativo) e Link para website/rede social.

RF6. Os restaurantes serão cadastrados pelos usuários do sistema, os quais poderão indicar/cadastrar restaurantes que desejam.

RF7. O sistema deverá emitir um alerta, via e-mail ou Whatsapp, para o restaurante escolhido para que o mesmo tenha ciência da quantidade de pessoas que irão para o estabelecimento.

RF8. O sistema deverá disponibilizar um relatório com os restaurantes mais votados do mês.

RF9. O sistema deverá disponibilizar um relatório com os restaurantes melhor avaliados no mês.

RF10. O sistema deverá disponibilizar um relatório com os restaurantes que o usuário mais votou.

RF11. O sistema deverá apresentar junto ao cadastro do restaurante o prato do dia.

RF12. O sistema deverá permitir a integração com Facebook para o cadastro do usuário.

RF13. O sistema deverá permitir a edição de dados do cadastro do usuário, dentre os quais, poderão ser editados: Nome, E-mail e Senha.

RF14. Votar Semana

1. O sistema deve realizar uma votação semanal, para eleger quais os restaurantes mais desejados pelos alunos.
   1.1 A votação semanal acontecerá uma vez por semana, ao final do último dia de aula da semana, ficando aberto para receber votos até as 13 horas do domingo.
   1.2 Essa votação deve eleger em quais restaurantes os alunos irão almoçar no decorrer da próxima semana.
   1.3 O sistema deve verificar quantos almoços acontecerão durante a semana, e permitir que cada usuário vote na quantidade de restaurantes necessários para atender à quantidade de almoço, o mesmo restaurante não pode ser selecionado mais de uma vez pelo mesmo usuário para a mesma votação.
   1.4 O sistema deve remover da lista os restaurantes mais votados na semana anterior, para que dessa maneira dê chance para que todos os alunos consigam ir

em seus restaurantes preferidos. Para definir a quantidade de restaurantes que não participaram da votação, o sistema deve verificar a parametrização.

2. Cada usuário pode votar uma única vez, selecionado a quantidade de restaurantes necessários.

2.1 Os votos não podem ser alterados.

2.2 Os votos devem ser privados, ou seja, os usuários não podem ver os votos uns dos outros, só podem ver seus próprios votos.

3. Ao final da votação, com a mesma encerrada, deve ser enviado um e-mail para os alunos com o resultado da votação.

RF14 – Votar Dia

1. O sistema deve realizar uma votação para determinar em qual ordem os restaurantes escolhidos na votação semanal serão visitados.

1.1 A votação diária inicia as 13hrs do dia anterior e permanece aberta até as 11hrs do dia do almoço.

2. O usuário pode votar uma única vez, e seu voto não pode ser alterado.
3. O voto deve ser privado.
4. Ao final da votação diária, o sistema deve enviar uma notificação, por e-mail e no aplicativo, informando que a votação foi encerrada e qual o restaurante selecionado.

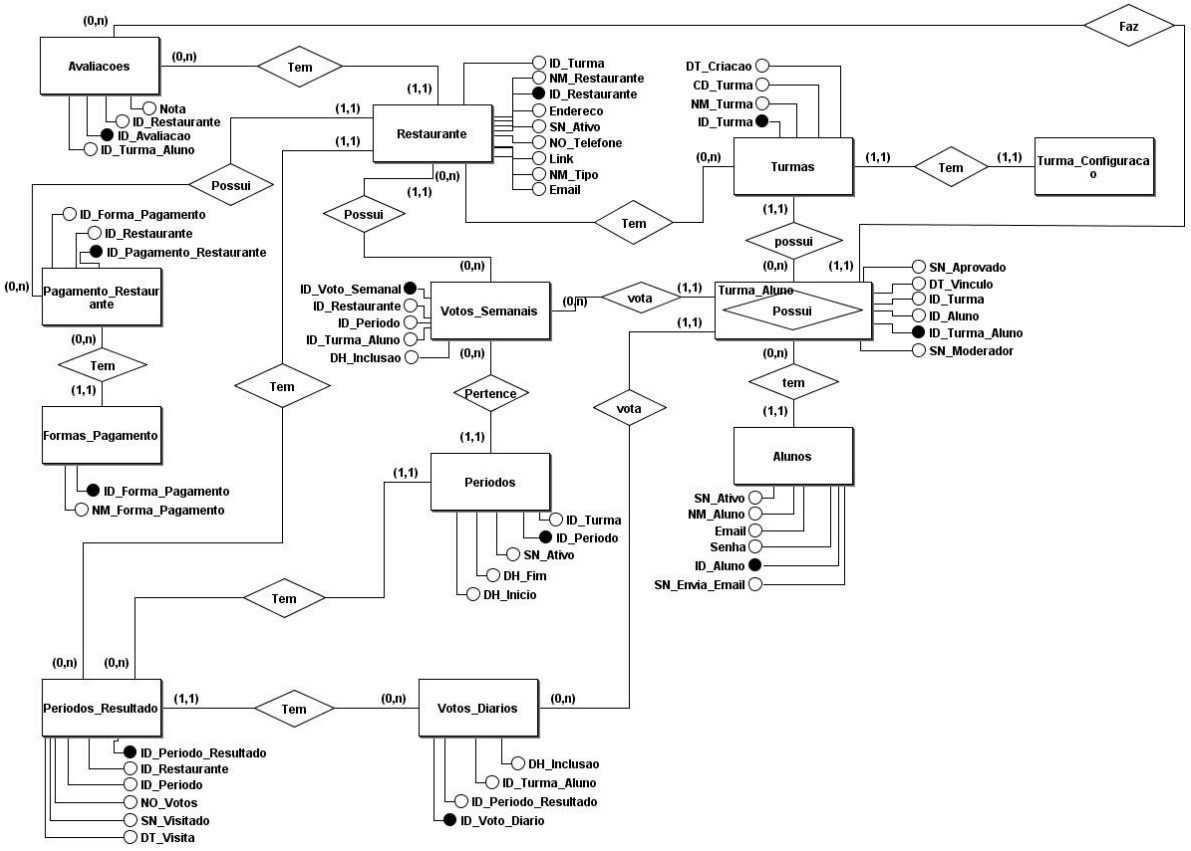
4.1 O resultado da votação deve exibir os 3 primeiros colocados, em ordem de votação, sendo que o visitado será o que está em primeiro lugar.

4.2 O restaurante selecionado deve sair da lista de votação dos próximos dias.

4.3 No último dia de almoço não será necessário realizar votação, o restaurante que será visitado deve ser o que sobrou. Neste dia será exibido diretamente o restaurante que sobrou e os dois próximos da lista semanal que não ficaram entre os selecionados da votação.

4.4 Ao final da votação de cada dia, será permitido ao usuário eleger uma nota para o restaurante, o usuário só pode votar uma vez para cada dia, caso o usuário já tenha atribuído uma nota para o restaurante no passado, essa nota deve ser substituída.

6.2 Diagrama do Banco de Dados



Desenvolvimento x Responsável

Descrição	Responsável
Criação do Layout	Otávio
Desenvolvimento da API	Bianca, Hellen
Desenvolvimento das telas da aplicação web	Hugo, Otávio, Jonas
Integração da API com aplicação web	Bianca, Hellen, Otávio