INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA

CAMPUS MONTEIRO

CST EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

GABRIEL OLIVEIRA FLORENCIO DA SILVA

PATRÍCIA DOS SANTOS PEREIRA

ROSENATO BARRETO DE LIMA

TRABALHO DE CONCLUSÃO DE CURSO

SisRest: Um Sistema Web para o Restaurante Estudantil do IFPB

MONTEIRO

2023

GABRIEL OLIVEIRA FLORENCIO DA SILVA
PATRÍCIA DOS SANTOS PEREIRA
ROSENATO BARRETO DE LIMA

TRABALHO DE CONCLUSÃO DE CURSO
SisRest: Um Sistema Web para o Restaurante Estudantil do IFPB

Trabalho de Conclusão de Curso (TCC) apresentado ao Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba, Campus Monteiro, formatado na Modalidade Projeto de Implementação, como pré-requisito para obtenção do Título de Tecnólogo em Análise e Desenvolvimento de Sistemas, sob orientação do Prof. Me. Giuseppe Anthony Nascimento de Lima.

MONTEIRO
2023

Dados Internacionais de Catalogação na Publicação – CIP
Bibliotecária responsável Porcina Formiga dos Santos Salgado CRB15/204
IFPB Campus Monteiro.

S586t Silva, Gabriel Oliveira Florencio da.; Pereira, Patrícia dos Santos.; Lima, Rosenato Barreto de.

Trabalho de conclusão de curso SisRest: um sistema Web para o restaurante estudantil do IFPB / Gabriel Oliveira Florencio da Silva; Patrícia dos Santos Pereira; Rosenato Barreto de Lima – Monteiro-PB. 2023.

46 fls. : il.

TCC (Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas) - Instituto Federal de Educação, Ciência e Tecnologia da Paraíba - IFPB campus, Monteiro.

Orientador: Prof. Me. Giuseppe Anthony Nascimento de Lima.

1. Software - Desenvolvimento 2. Restaurante Estudantil 3. IFPB – Campus Monteiro I Titulo

CDU 004.453

GABRIEL OLIVEIRA FLORENCIO DA SILVA
PATRÍCIA DOS SANTOS PEREIRA
ROSENATO BARRETO DE LIMA

TRABALHO DE CONCLUSÃO DE CURSO

SisRest: Um Sistema Web para o Restaurante Estudantil do IFPB

Trabalho de Conclusão de Curso (TCC) apresentado ao Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba, Campus Monteiro, formatado na Modalidade Projeto de Implementação, como pré-requisito para obtenção do Título de Tecnólogo em Análise e Desenvolvimento de Sistemas, sob orientação do Prof. Me. Giuseppe Anthony Nascimento de Lima.

BANCA EXAMINADORA

Prof. Me. Giuseppe Anthony Nascimento de Lima
Professor do IFPB (Orientador)

Prof. Me. Julierme Silva de Araújo
Professor do IFPB (Examinador)

Prof. Esp. Wagner de Oliveira Santos
Professor do IFPB (Examinador)

Aprovado e permitida a publicação.

Monteiro-PB, 06 de outubro de 2023.

gov.br
Documento assinado digitalmente
WANDERLEY ALMEIDA DE MELO JUNIOR
Data: 07/11/2023 21:24:00-0300
Verifique em https://validar.iti.gov.br

Prof. Me. Wanderley Almeida de Melo Júnior
Coordenador do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

AGRADECIMENTOS

Nossa gratidão a Deus, por nos proporcionar força e sabedoria ao longo dessa jornada acadêmica.

Aos nossos amigos, por todo o apoio e pela ajuda, que muito contribuíram na nossa vida acadêmica.

Aos nossos pais, irmãos, companheiros e filhos, que nos incentivaram nos momentos difíceis e compreenderam a nossa ausência enquanto nos dedicávamos à realização deste trabalho.

Ao professor Giuseppe Lima, nosso orientador, por ter desempenhado a função com dedicação, doando-nos seu tempo e conhecimento.

Aos professores do Curso de ADS, que contribuíram no processo de ensino-aprendizagem em nossa formação, com sabedoria e dedicação.

A todos, que direta ou indiretamente, participaram no desenvolvimento desse trabalho.

RESUMO

Esse trabalho objetivou contribuir com o processo de gestão do Restaurante Estudantil do IFPB Campus Monteiro, que fornece refeições gratuitas para os alunos da instituição. Atualmente, o controle de informações de refeições é realizado manualmente, em que os alunos contemplados informam antecipadamente as refeições do dia que desejam, sem assistência de uma aplicação de software, havendo a gestão da quantidade de refeições diárias por meio de uma lista diária (planilha), que é enviada a um fornecedor. Portanto, foi projetada e desenvolvida uma aplicação *web*, o SisRest, a fim de diminuir o esforço desse controle, por meio do acesso às informações sobre a concessão de refeições para contemplados em editais do Programa Institucional de Alimentação Estudantil. A aplicação foi desenvolvida sob boas práticas de concepção de software, a partir de uma arquitetura multicamadas, considerando importação de dados de beneficiários e matrículas do sistema da instituição (SUAP), com um *back-end* Spring Boot sob uma API de serviços REST e *front-end* sob o *framework* React.js, com acesso à lógica da aplicação via Axios. Serão demonstradas as técnicas e tecnologias nesse desenvolvimento, incluindo-se as especificações e modelos de software utilizados nesse processo, assim como se consolidou a primeira versão da aplicação.

Palavras-chave: Desenvolvimento de software; Aplicação Web; Restaurante Estudantil.

ABSTRACT

This work aimed to contribute to the student restaurant management process of the IFPB Campus Monteiro student restaurant, which provides free meals to the institution´s students. Currently, the control of meal information is carried out manually, in which eligible students inform with antecedence the meals they need by day without assistance from a software application, aiming the management of a daily list for each diary meal and its quantities that must be sent to a meal supplier (in a spreadsheet). Therefore, the web application SisRest was design and developed in order to reduce the effort of this control, by accessing information about the meals that must be provided to the granted students of the institutional food program. The application was developed under good software design practices based on a multi-layer architecture and considered the data imported from the institution system (SUAP) about student beneficiaries and their enrollment´s data. A Spring Boot back-end was developed to provide a REST services API and the front-end was conceived with the React.js framework, accessing business logic via Axios. The techniques and technologies in this development will be demonstrated, including the specifications and software models used in this process, as well as how the first version of the application was achieved.

Keywords: Software Development; Web Application; Student Restaurant.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ADS	Análise e Desenvolvimento de Sistemas
API	*Application Programming Interface*
CAEST	Coordenação de Apoio ao Estudante
CRUD	*Create, Read, Update, Delete*
DAO	*Data Access Object*
HTTP	*Hypertext Transfer Protocol*
IFPB	Instituto Federal de Educação, Ciência e Tecnologia da Paraíba
JPA	*Java Persistence API*
JSON	*JavaScript Object Notation*
MVC	*Model View Controller*
PNAE	Programa Nacional de Alimentação Escolar
PCD	Pessoa com Deficiência
PoC	*Proof of Concept* (Prova de Conceito)
REST	*Representational State Transfer*
SQL	*Structured Query Language*
SUAP	Sistema Unificado de Administração Pública
UML	*Unified Modeling Language*

SUMÁRIO

1 INTRODUÇÃO

1.1 CONTEXTO E PROBLEMATIZAÇÃO (DOMÍNIO DO PROBLEMA)

O restaurante estudantil do Instituto Federal da Paraíba (IFPB) atende aos alunos da instituição com alimentação diária, que é oferecida, de acordo com o total de vagas ofertadas através de editais específicos, realizados dentro do Programa de Alimentação Estudantil da instituição. Para se ter uma ideia, somente em 2023, no Campus Monteiro, foram concedidas 80 vagas no âmbito do Programa de Alimentação do IFPB, entre almoço e jantar (IFPB, 2023).

Os alunos contemplados possuem uma rotina de acesso ao restaurante, devendo informar com antecedência os dias da semana e qual refeição (café, almoço e jantar) necessitam, havendo confirmação pela Coordenação de Assistência Estudantil (CAEST-MT). Após isso, a fim de se evitar o desperdício, o estudante deve se comprometer em confirmar, com a devida antecedência, as refeições que irá consumir. A CAEST-MT então processa essas confirmações no início do dia, gerando uma lista diária com a relação de pessoas que terão acesso ao restaurante e o quantitativo de refeições a serem disponibilizadas por fornecedores contratados pela instituição.

Uma primeira entrevista com o departamento responsável foi realizada com o intuito de compreender o esforço de concessão de refeições e de controlar o acesso sem desperdício, em que se verificou que esse processo era realizado ainda com a manipulação de planilhas eletrônicas.

Conjuntamente, tem havido o uso do principal sistema de informação da instituição, o SUAP (IFPB, 2017), no apoio a esse controle do restaurante estudantil. Através dele, os estudantes podem inserir os dados necessários para confirmar ou cancelar aquelas refeições em dia que irão consumir ou justificar faltas em refeições confirmadas.

1.2 JUSTIFICATIVA

A maior limitação encontrada na gestão do restaurante consiste no esforço para a geração da lista diária dos alunos, visto que ela é essencial para a operacionalização da concessão de refeições.

O SUAP ainda auxilia no processo de obtenção de listagens eletrônicas com informações dos alunos beneficiários em editais vigentes, mas a verificação daqueles habilitados à acessarem no dia ainda exige uma filtragem manual pela CAEST-MT,

sem a assistência pelo sistema, que envolve o cruzamento de informações sobre atualização, suspensão ou revogação de concessões, inclusive daqueles estudantes com pendências. Por exemplo, um estudante consegue confirmar antecipadamente a sua refeição via SUAP, mas se ele acumular faltas ou não estiver mais matriculado, ainda poderá continuar a confirmar sua ida no dia, cabendo à CAEST-MT verificar isso e retirá-lo da lista diária enquanto ele não sanear tal pendência.

Portanto, daí se evidencia a importância da geração da lista diária na operacionalização do fornecimento diário de refeições, que demanda rápido cruzamento dessas informações pelo departamento. Conforme explicitado, esse controle também envolve os servidores que trabalham no restaurante, que verificam a vigência de autorização do acesso aos estudantes, assim como envolve os fornecedores das refeições, que deverão garantir a disponibilização a todos aqueles estudantes esperados, para cada refeição diária.

Portanto, a elaboração de uma aplicação *web* que cobrisse essa operação com maior segurança e rapidez facilitaria esse controle, que se baseia em um plano de cardápios semanais rotativos. Adicionalmente, se suportaria a geração de relatórios sobre o uso do restaurante, por exemplo, em um determinado período.

Com essa premissa, foi iniciado o desenvolvimento da aplicação SisRest, a partir da disciplina “Projeto I”, do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas (ADS), em que foram realizadas atividades de concepção de software e de prototipação como as de engenharia de requisitos e de projeto de software a partir de estudo de viabilidade tecnológica com provas de conceito (PoC). Em seguida, na disciplina de “Projeto II”, foi preestabelecida uma lista de entregáveis, desenvolvidos e testados sob um regime de iterações, dentro de um processo incremental.

Complementarmente, destaca-se que o projeto do SisRest intencionou o alcance de vantagens e benefícios para os usuários do restaurante, pois, além da informatização da lista diária de confirmação de refeições, eles poderiam, inclusive, dispor as suas restrições alimentares e obter as informações nutricionais do cardápio semanal de refeições, mais comodamente, algo que, todavia, não é possível com o SUAP.

1.3 OBJETIVOS

1.3.1 OBJETIVO GERAL

O presente trabalho de conclusão de curso objetivou consolidar as especificações e o desenvolvimento da aplicação *web* SisRest, que visa apoiar a operacionalização do serviço de distribuição de refeições aos estudantes do Restaurante Estudantil do Campus Monteiro, a partir do estudo de caso da demanda da CAEST-MT.

1.3.2 OBJETIVOS ESPECÍFICOS

Visando o alcance do objetivo geral, foram estabelecidos os seguintes objetivos específicos para este trabalho, considerando as características e as atividades para o desenvolvimento do produto de software SisRest:

i. Elaborar a análise e projeto da aplicação com base em prototipação e de forma centrada no usuário;
ii. Desenvolver e implantar os componentes de software em frentes separadas, simultaneamente, considerando projetos de *back-end* e *front-end*, respectivamente para a camada de apresentação e de negócios da aplicação;
iii. Planejar e realizar testes do sistema.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 CONTEXTUALIZAÇÃO DO PÚBLICO-ALVO E DAS APLICAÇÕES DO PRODUTO DE SOFTWARE

Para a rápida compreensibilidade das oportunidades e aplicabilidades do SisRest, aplicou-se o modelo de análise de negócio Lean Canvas.

Criado por Ash Maurya, esse modelo possui segmentos de especificação em 09 blocos que auxiliam a determinar o valor agregado de uma solução para uma problemática, a partir de hipóteses que precisam ser validadas logo cedo por *startups* (SEBRAE, 2019). O Quadro 1 interpreta a aplicação *web* SisRest sob esse modelo.

Quadro 1 – Lean Canvas do SisRest

Problema	• Esforço para obtenção e gestão da lista diária de refeições, com base em confirmações antecipadas de usuários do restaurante estudantil. • Ausência de gestão informatizada quanto ao cardápio semanal, informação nutricional e sobre as necessidades alimentares específicas (vegetarianos, veganos, alergias entre outras) de usuários.
Segmento de Cliente	• Estudantes de instituições públicas e privadas de ensino. • Funcionários dessas instituições que gerenciam a oferta de refeições.
Proposta de Valor	• Agregar mais informação sobre o cardápio do restaurante e sobre o seu uso por beneficiários, assim como mais segurança e comodidade no processo antecipado de confirmação diário de consumo, para cada uma das refeições do dia.
Solução	• Confirmação antecipada de refeições do dia. • Controle do estado de acesso ao restaurante mais integrado, da concessão dos dias de acesso até o controle do registro do consumo de refeições. • Geração de relatórios facilitada por gestores do restaurante sobre o seu uso.
Canais	• Página de *download* da ferramenta. • Redes sociais.
Receitas	• Editais de fomento interno.
Estrutura de Custos	• Desenvolvedores mantenedores da aplicação. • Infraestrutura de implantação.
Métricas-Chave	• Volume de usuários cadastrados na aplicação. • Volume de downloads. • *Feedback* positivo nos canais • Pesquisa de satisfação com usuários
Vantagem Competitiva	• Aplicação *web* de fácil acesso e aprendizado por usuários. • Integração com a rotina de planejamento da lista diária de refeições com as demais atividades e dados de gerenciamento de uso, orientada por editais de concessão.

Fonte: Os autores.

2.2 Conceitos e Trabalhos Relacionados ao Domínio do Problema do Software

A regulamentação vigente no IFPB que trata sobre o restaurante estudantil se baseia na Política de Assistência Estudantil da instituição, que foi aprovada pela Resolução nº 16/2018 – CONSUPER (IFPB, 2018). Entre outros objetivos, essa política visa garantir a permanência e o êxito dos estudantes, bem como a igualdade de oportunidades socioeconômicas. Ainda, é nela que se designa que o Programa de Alimentação Institucional deve oportunizar aos estudantes o acesso a uma alimentação adequada e saudável, na perspectiva de assegurar condições indispensáveis ao pleno desenvolvimento acadêmico, social e de convivência estudantil, por meio de serviços de alimentação implantados nos campi.

Também é importante ressaltar o alinhamento dessa política com a regulamentação estabelecida pelo Programa Nacional de Alimentação Escolar (PNAE), seguindo as diretrizes da Lei nº 11.947/2009 (BRASIL, 2009) e da Resolução CD/FNDE nº 06/2020 (MEC, 2020) e suas alterações. Portanto, o PNAE visa oferecer recursos financeiros federais, buscando promover hábitos alimentares saudáveis e contribuir para o crescimento, desenvolvimento, aprendizagem e rendimento escolar dos alunos de todas as modalidades da educação básica. Considerando isso, as instituições educacionais devem promover uma gestão eficiente dessa alimentação.

No caso do IFPB, a instrumentalização dela ocorre por meio de seus restaurantes estudantis, em que os beneficiários são selecionados por meio de editais com um quantitativo de vagas predefinido, reservando-se no mínimo 5% delas para pessoas com deficiência (PCDs). A partir da obtenção do Índice de Vulnerabilidade Social (IVS), cada estudante inscrito é então priorizado e classificado em um edital.

No caso do Campus Monteiro (IFPB, 2023), aqueles que foram contemplados com a concessão devem proceder com a solicitação de acesso ao departamento responsável, reservando os dias e respectivas refeições (ex.: almoço e jantar) que necessitam utilizar, em que a manutenção da concessão é condicionada à regularidade da matrícula e às frequências nas atividades do curso. Ainda, a inassiduidade ao restaurante estudantil sem apresentação de justificativa após um certo tempo (por exemplo, 05 dias úteis) também pode ocasionar o cancelamento da concessão. Por fim, na tentativa de diminuir desperdícios, o estudante contemplado deve confirmar até às 8h do dia da refeição, se irá comparecer, pelo sistema de informação principal da instituição, o SUAP. Conforme já evidenciado, esse

cruzamento de informações e controle da concessão de acesso ao restaurante todavia vem sendo realizado manualmente pelo assistente social do campus.

Adicionalmente, as respectivas refeições dentro de um planejamento em rodízio de cardápios semanais é determinada por um profissional nutricionista, também vinculado à CAEST-MT, que deve estar atento às restrições alimentares indicadas por eles.

2.3 TECNOLOGIAS UTILIZADAS

Nesta seção serão apresentadas e justificadas as tecnologias selecionadas no desenvolvimento dos projetos de software de *back-end* (camadas de lógica de negócio e de persistência) e de *front-end* (camada de interface com o usuário) do SisRest, assim como aquelas para testá-lo.

2.3.2 BACK-END

A camada de lógica da aplicação do SisRest foi desenvolvida com o já bastante consolidado *framework open source* Spring Boot[1], que exige um ambiente de execução em linguagem Java[2] (portável), auxiliando na rápida implementação de componentes e testes em projetos *web* baseado em serviços. Com ele é possível a simplificação da configuração de dependências e do *build* de aplicações, considerando *starters*, bibliotecas para provimento desses serviços, aspectos de segurança e de controle de acesso, persistência, entre outros (SPRING, 2023).

A camada lógica da aplicação foi construída para ser acessada por meio de serviços *web* sob o protocolo de transferência de estado representacional REST, que opera sob o protocolo HTTP a requisição de recursos ao servidor que executa tal lógica, cujos dados da aplicação respondidos em diversos formatos (por exemplo, JSON).

Na persistência de dados, houve adesão à dependência Spring Data JPA[3], que suporta o mapeamento objeto-relacional dentro das especificações da interface de programação padrão JPA, a partir de objetos Repository que equivalem a objetos DAO (de acesso a dados, com operações básicas de CRUD prontas), que podem ser estendidos para provimento de métodos de consultas mais personalizados. O

[1] Disponível em: https://spring.io/projects/spring-boot. Acesso em: 15 ago. 2023.
[2] Disponível em: https://www.oracle.com/br/java/. Acesso em: 15 ago. 2023.
[3] Disponível em: https://spring.io/projects/spring-data-jpa. Acesso em: 15 abr. 2023.

Hibernate foi a implementação da JPA utilizada, acessando-se uma instância de banco de dados relacional Postgresql[4].

Para garantir a segurança do sistema, foram implementados recursos de autenticação e de autorização, a partir da dependência Spring Security[5], controlando o acesso aos serviços REST de acordo com o perfil do usuário. Foi empregado o uso de *tokens* de autenticação com criptografia, sob o padrão JWT[6] (JSON Web Token), que evita ataques de interceptação e falsificação, havendo a concessão de um *token* ao usuário, logo que ele se autenticar pela primeira vez na aplicação.

Como base de dados de autenticação de contas de usuários, foi utilizada uma dependência de acesso a serviço de autenticação Oauth2 do Google[7], sob o domínio de Internet da instituição. Esse serviço é provido por uma API Java, com a qual se pode configurar o acesso se dispondo uma chave específica, sob a qual o usuário é redirecionado a uma URL para autorizar o SisRest a confirmar a sua identidade, a partir de uma tela de autenticação provida pelo próprio Google (SINGH, 2018).

2.3.3 FRONT-END

A camada de apresentação ou de interface com o usuário foi construída sob o *framework* React.js[8], que pode ser executado em ambientes de execução da linguagem JavaScript. Ele permite a organização da implementação do código da interface a partir de componentes reutilizáveis, facilitando a sua manutenção.

Em tempo de desenvolvimento, auxiliando no ambiente de execução do projeto do *front-end*, adotou-se o Node.js[9], que disponibiliza um servidor HTTP configurável, também, empregando-se no gerenciamento de dependências de pacotes JavaScript o NPM (Node Package Manager).

A principal biblioteca de componentes de interface adicionada foi o PrimeReact[10], no *script* equivalente ao componente de aplicação React.js. Ela fornece componentes correspondentes aos *widgets* de apresentação e de entrada de dados, já prontos e parametrizáveis, como menus, botões, campos de formulários, inclusive para obtenção de listagens de entidades com métodos de filtragem prontos.

---

[4] Disponível em: https://www.postgresql.org/. Acesso em: 17 abr. 2023.
[5] Disponível em: https://spring.io/projects/spring-security. Acesso em: 18 abr. 2023.
[6] Disponível em: https://datatracker.ietf.org/doc/html/rfc7519. Acesso em: 26 mar. 2023.
[7] Disponível em: https://console.cloud.google.com/apis. Acesso em: 26 mar. 2023.
[8] Disponível em: https://pt-br.react.dev. Acesso em: 15 ago. 2023.
[9] Disponível em: https://nodejs.org/en/download. Acesso em: 05 abr. 2023.
[10] Disponível em: https://primereact.org/. Acesso em: 07 abr. 2023.

Complementarmente ao React.js foi aplicada a biblioteca TailwindCSS[11], visando a facilitação da personalização da estilização sob descritores de estilo CSS, sob os componentes prontos da biblioteca de interface PrimeReact.

2.3.4 TESTES

No desenvolvimento dos testes do sistema SisRest foram utilizadas as seguintes tecnologias, o JUnit[12] (testes unitários de funcionalidades de código dentro da camada de lógica da aplicação) e o Selenium[13] (testes de aceitação, simulando interações a partir da camada de interface com o usuário), para maior confiabilidade do software.

[11] Disponível em: https://tailwindcss.com/. Acesso em: 12 fev. 2023.
[12] Disponível em: https://junit.org/junit5/. Acesso em: 15 ago. 2023.
[13] Disponível em: https://www.selenium.dev/. Acesso em: 15 ago.2023.

3 DESENVOLVIMENTO E RESULTADOS OBTIDOS

3.1 ENGENHARIA DE REQUISITOS

Segundo Sommerville (2011), os requisitos definem o que um sistema é capaz de fazer, considerando o provimento de serviços com finalidade determinada e as restrições de funcionamento dos mesmos, em que o processo para a sua descoberta, análise, documentação e verificação é chamado de engenharia de requisitos.

Os requisitos funcionais do SisRest (Quadro 2) foram obtidos a partir da aplicação das seguintes técnicas de levantamento: (i) entrevistas com o cliente e usuários, para obtenção de pontos de vista da assistente social e da nutricionista do IFPB Campus Monteiro; (ii) oficinas, com essas profissionais, acerca de como era gerada a lista diária de refeições manualmente, considerando estudantes e o uso do SUAP; e (iii) análise de documentos, como a planilha eletrônica de estudantes contemplados em um edital de concessão, a partir do SUAP, assim como também a própria lista diária de refeições e editais de concessão já realizados.

Quadro 2 – Requisitos funcionais da aplicação

MÓDULO	ID	REQUISITO FUNCIONAL
Acesso e Conta	RF.ACC.01	Acesso ao sistema através de credenciais de conta
	RF.ACC.02	Manter contas do tipo assistente social, nutricionista e beneficiário
Editais	RF.EDT.01	Importação de alunos do SUAP (matrículas e e-mails via arquivos CSV de alunos e de editais)
	RF.EDT.02	Gerenciamento de beneficiários oriundos do SUAP por edital
	RF.EDT.03	Aprovação dos dias de acesso solicitados pelo estudante.
	RF.EDT.04	Gerenciamento de editais de concessão
Nutricional	RF.NUT.01	Acesso ao cardápio por estudantes
	RF.NUT.02	Gerenciamento da sequência de cardápios semanais
	RF.NUT.03	Gerenciamento de refeições com informações nutricionais
Estudante Beneficiário	RF.EST.01	Cancelamento de uma refeição pelo estudante
	RF.EST.02	Justificação da falta pelo estudante
	RF.EST.03	Inscrição em lista de espera diária de refeição
	RF.EST.04	Avaliação de uma refeição consumida pelo estudante
	RF.EST.05	Registro de restrições alimentares pelo estudante
	RF.EST.06	Solicitação dias de acesso ao restaurante pelo estudante num edital
Controle Diário de Refeições	RF.CDR.01	Consolidação do quantitativo diário, pelas refeições do dia
	RF.CDR.02	Consolidação da lista de estudantes diária, por refeições do dia
	RF.CDR.03	Verificação diária do refeições canceladas do dia por estudantes
	RF.CDR.04	Gerenciamento diário das presenças e das faltas do estudante
	RF.CDR.05	Impressão da lista por refeições do dia e respectivos estudantes
	RF.CDR.06	Envio da quantidade por refeições do dia ao fornecedor

Fonte: Os autores.

Já os requisitos não-funcionais foram obtidos a partir de algumas das características de qualidade da norma ISO/IEC 9126-1 (ISO, 2001), em que se destacam os apresentados no Quadro 3.

Quadro 3 – Requisitos não-funcionais da aplicação

CARACTERÍSTICA	ID	REQUISITO NÃO-FUNCIONAL
Funcionalidade	RNF.FUNC.01	Leitura e formato de QR Code no registro de consumo de refeições
	RNF.FUNC.02	Arquitetura do *back-end web* em Java com Spring *Framework*.
	RNF.FUNC.03	Arquitetura do *front-end* com React.js e Tailwind CSS.
Confiabilidade	RNF.CONF.01	Persistência objeto-relacional (PostgreSQL com Hibernate)
Portabilidade	RNF.POR.01	Acesso multiplataforma por navegadores padrão
Usabilidade	RNF.USA.01	Elementos textuais e iconografia em linguagem objetiva e de fácil compreensão
Manutenibilidade	RNF.MAN.01	Aplicação de testes unitários, de integração, de sistema e de aceitação com JUnit, Mockito, Selenium.
	RNF.MAN.02	Repositório de código com controle de versão no Github.

Fonte: Os autores.

Cada um dos requisitos funcionais e não funcionais foram especificados, considerando os seguintes elementos: (i) a justificação mais as regras de negócio sobre cada um dos funcionais (quem pode acessar, quando e de que forma, condições, aspectos obrigatórios, etc); e (ii) a justificação mais os parâmetros de aceitação (de verificação de atendimento) sobre cada um dos não-funcionais. Ainda, foi implementado um processo de validação de requisitos, em que os clientes oportunamente os revisaram e convalidaram.

3.2 PROJETO COMPORTAMENTAL E ESTRUTURAL

Na fase de análise de requisitos foram modelados os diagramas UML considerados mais essenciais, como o de casos de uso (Figura 1) e o de classes, esse último em nível conceitual (Figura 2), somente entidades do domínio do negócio, atributos e associações.

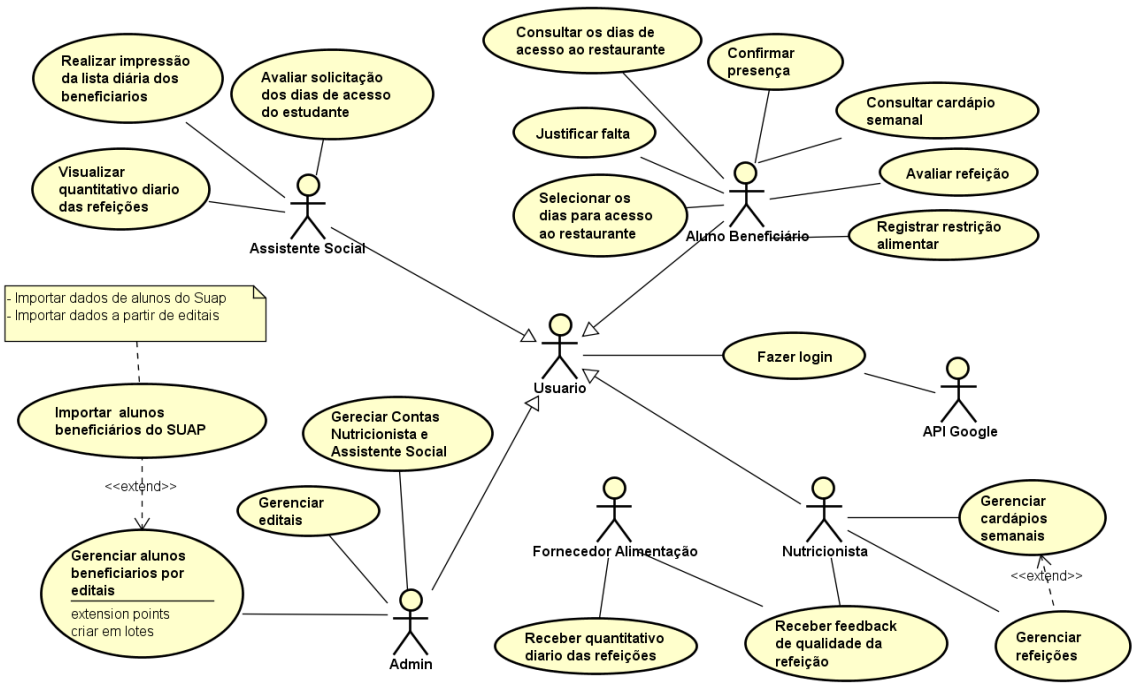
Ambos os diagramas foram atualizados ao longo da evolução do software, a partir de informações com *stakeholders* do projeto. O diagrama de classes conceitual não foi evoluído para um diagrama de projeto propriamente, mas ainda assim serviu para a implementação das entidades informacionais do negócio, inclusive para avaliar

os mapeamentos de persistência objeto-relacional aplicáveis (um para um, um para muitos, muitos para muitos, operações em cascata, identificadores, entre outros).

Detalhando o diagrama de casos de uso, destaca-se a disposição de casos sob os atores “Aluno Beneficiário” que exerce no sistema o papel de fazer a solicitação e confirmação de refeições, bem como o de visualizar o cardápio semanal. Já o ator “Nutricionista”, pode gerenciar cardápios oferecidos e avaliar as solicitações de restrições alimentares. Ainda, o ator “Assistente Social” faz o gerenciamento das listas diárias, avaliação e confirmação dos pedidos de acesso ao sistema. Por fim, o ator “Fornecedor Alimentação”, deve receber o quantitativo de cada refeição diária e obter retorno sobre a qualidade das mesmas.

O ator “Admin” (administrador) pode ser um “Assistente Social” ou “Nutricionista” liberado para tal por outro administrador preexistente. Administradores realizam casos de uso mais sensíveis, como o CRUD de editais, de contas de nutricionistas ou de assistente sociais e de estudantes beneficiários. Nesse último caso, pode haver também a partir da importação em lote, a partir de dois arquivos do SUAP, um contendo a lista de beneficiários do edital e outro as matrículas e e-mails dos mesmos.

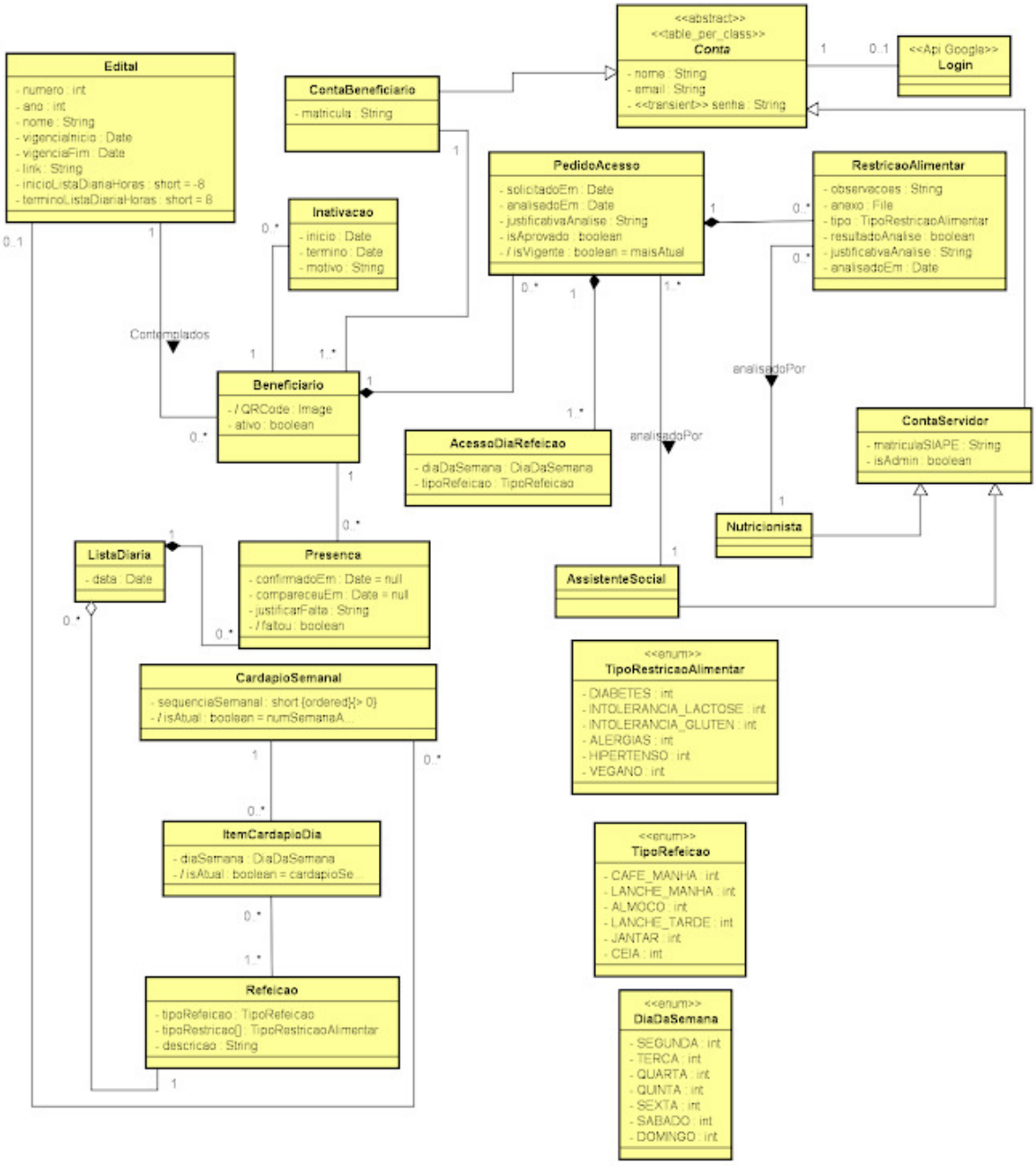
Figura 1 – Diagrama de casos de uso



Fonte: Os autores.

No diagrama de classes, foram detalhadas as principais entidades do domínio do negócio, com destaque para as classes “Cardápio Semanal”, em que se pode dispor cada “Refeicao” cadastrada pelo “Nutricionista” (entre almoço, jantar, etc, determinando-se se sua oferta ainda está “ativa”) sob objetos do tipo “ItemCardapioDia”. Esses últimos designam o dia da semana em que a refeição ocorrerá. Cada cardápio de itens é sequenciado (atributo “sequenciaSemanal”), possibilitando que a aplicação implemente o rodízio de refeições, a partir de uma determinada semana do ano.

Figura 2 – Diagrama conceitual de classes



Fonte: Os autores.

A classe “Edital” é a classe que designa temporalmente quais as concessões de acesso que estão ativas (a partir da associação com a classe “Beneficiario”) e o itinerário de refeições fornecidas (via instâncias da classe “CardapioSemanal”, entre as ativas). Outra entidade importante é a classe “ListaDiaria” (marcada pelo atributo “data”), em que se sabe as refeições servidas em um dado dia e quais os alunos estão aptos a acessarem a mesma, registrando-se pela classe “Presenca” quando a confirmaram, quando consumiram a refeição e se faltaram.

A gestão de acesso ocorre por meio da entidade “ContaServidor”, especializada em “Nutricionista” e “AssistenteSocial”; e da entidade “ContaBeneficiario”. Essa última libera ocorrências da classe “Beneficiario”, sob uma mesma matrícula de um estudante contemplado em um “Edital”, em que se pode aplicar eventuais inativações ou reativações da concessão, inclusive dispondo o campo “justificativa” sobre cada instância da classe “Inativacao” relacionada.

O supertipo “Conta” inclui o e-mail da conta institucional de qualquer tipo de conta de usuário da aplicação, de forma que ele referencia a autenticação via serviço provido pelo Google, com OAuth2. Portanto, não foi necessário o armazenamento de valores de senha na aplicação, visando uma maior segurança.

Ainda na fase de análise de requisitos, foram modelados, sob demanda, alguns diagramas comportamentais UML, como o de atividades, para compreender processos de negócio e alguns diagramas de estados, para qualificar as transições de momentos de entidades importantes, como Beneficiário, Presença, Inativação e Lista Diária.

3.3 PROJETO ARQUITETURAL

O SisRest foi desenvolvido sob uma arquitetura *web* de três camadas, uma das mais comuns no desenvolvimento de sistemas, havendo: a camada de interface com o usuário; a camada de lógica de negócios e a camada de persistência de dados.

Nessa arquitetura, os componentes do software são agrupados de forma a serem implementados e implantados estrategicamente em separado. Portanto, essa abordagem possibilita um desenvolvimento de software mais fácil de manter e também de escalar, sem afetar os componentes envolvidos (RICHARDS; FORD, 2020).

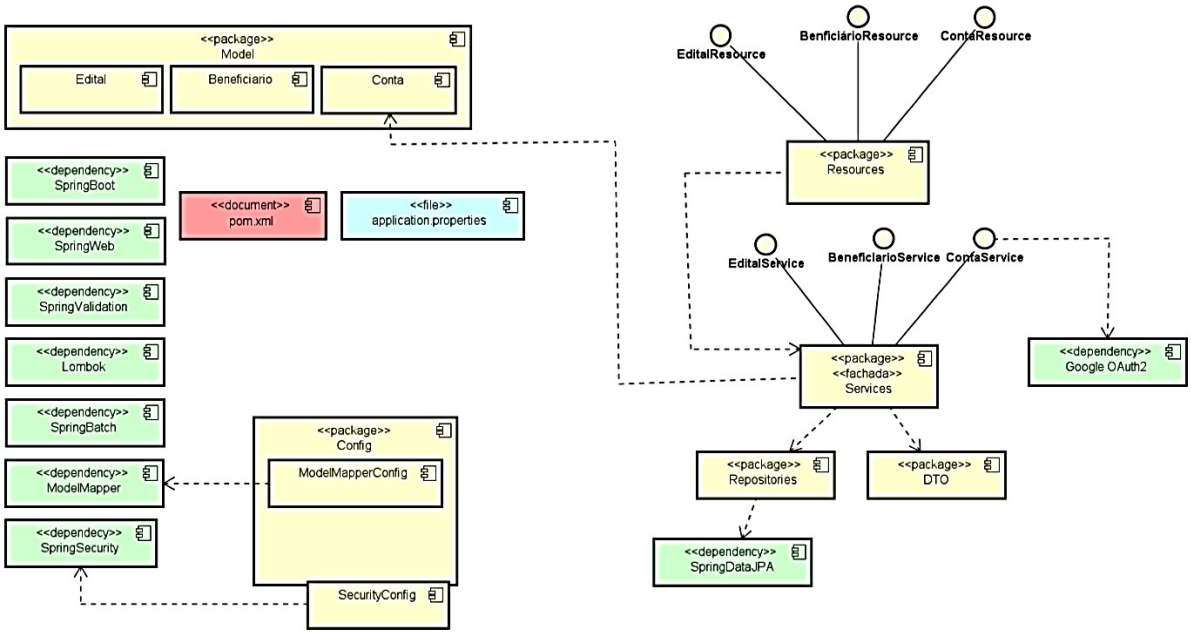
A Figura 3 detalha a organização do projeto com o *back-end*, englobando componentes das camadas de lógica de negócios e de dados da aplicação. Houve a

influência da disposição proposta pelo *framework* Spring, com o desenvolvimento de fachadas de acesso simplificado às funcionalidades que processam as regras sob as entidades do modelo do negócio mais agregadoras, a partir de interfaces providas com designação “Service” no diagrama.

O acesso externo remoto à lógica da aplicação se dá pelas interfaces providas com a designação “Resource” no diagrama, cada uma contendo os *endpoints* que mapeiam sob o protocolo REST as URIs de requisição que acionam as funcionalidades providas pelas fachadas “Service” correlacionadas no servidor.

Para a camada de dados, há o provimento de componentes sob a interface “Repository” (dentro do componente do tipo pacote de mesmo nome, Figura 3) do Spring Data JPA, em que cada entidade persistente do negócio passa a conter um objeto de acesso a dados (DAO) com os métodos de CRUD básicos e os mecanismos para dispor-lhes consultas personalizadas.

Figura 3 – Diagrama de componentes UML do projeto de *back-end*

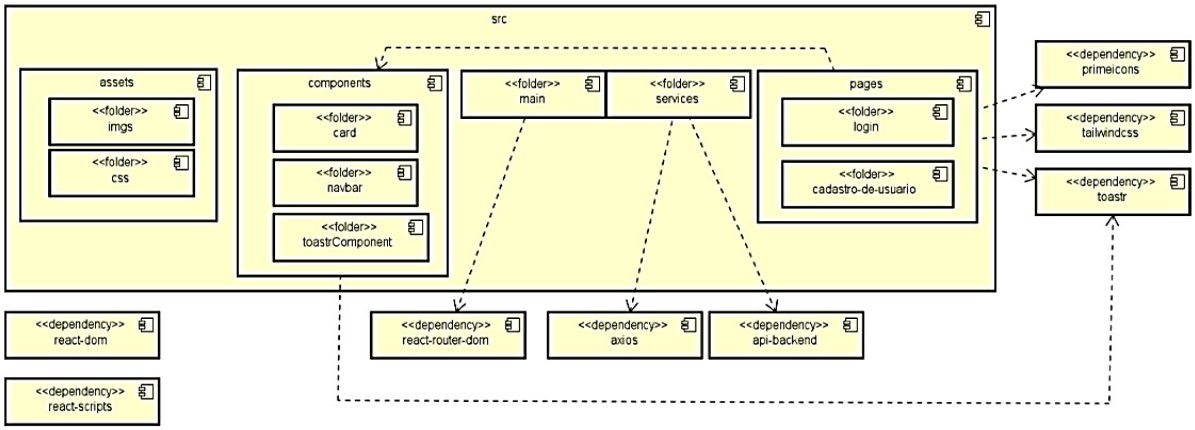


Fonte: Os autores.

Já a Figura 4, demonstra a organização do projeto de *front-end*, equivalente à camada de interface com o usuário. Nela o software ficou organizado através de diretórios específicos que abrigam os *script* de código correlatos, sendo assim, os referentes às telas do sistema ficaram distribuídos no componente pasta (*folder*) de

nome “pages” e os códigos mais reutilizáveis, como por exemplo, de menus da aplicação e outros componentes personalizados de apresentação, ficaram na pasta “components”. Já os códigos relacionados aos serviços de acesso à API do *back-end* da aplicação, ficaram no diretório “services”. A pasta “assets” reúne artefatos estáticos como imagens e estilização (CSS) da aplicação. No diretório “main” está o código relacionado com a configuração da aplicação e as rotas (endereços HTTP) de acesso às páginas da aplicação.

Figura 4 – Diagrama de componentes UML do projeto de *front-end*



Fonte: Os autores.

3.4 PROJETO E IMPLEMENTAÇÃO DA INTERFACE COM O USUÁRIO

A prototipação da interface é uma técnica para demonstrar conceitos e experimentar opções de projeto no uso de um software. Ela é utilizada, normalmente, sob uma metodologia de desenvolvimento rápido e interativo para que os clientes e/ou usuários possam prever o uso do sistema o mais cedo possível (SOMMERVILLE, 2011).

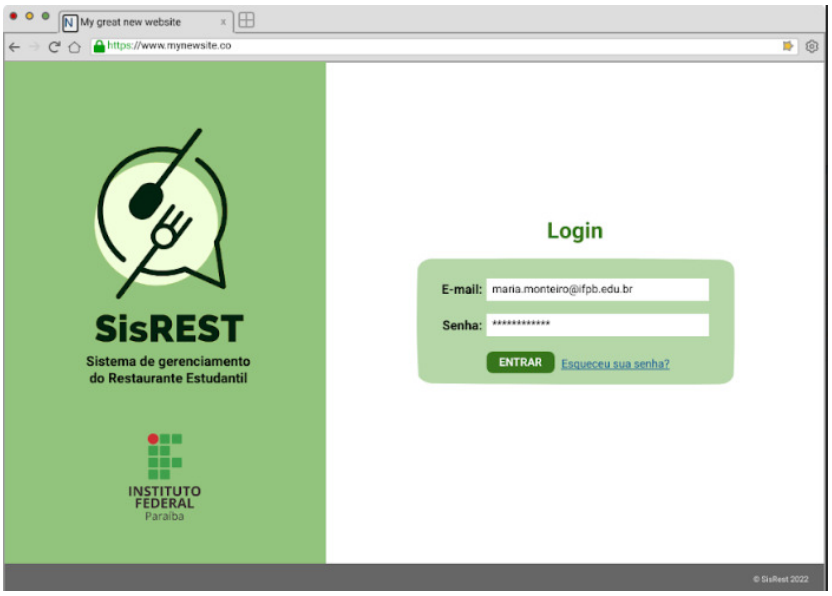
A interface do SisRest foi projetada sob três módulos essenciais, utilizando a ferramenta Balsamiq[14], a saber: uma para o usuário “Assistente Social”; uma para o “Nutricionista”; e uma para o “Estudante Beneficiário”, em que todos devem acessar o sistema a partir de uma tela de autenticação (Figura 5).

Ao se autenticar, um “Assistente Social”, encontra em seu módulo as ferramentas necessárias para gerenciar os programas de alimentação estudantil de maneira eficaz, gerenciando usuários beneficiários por edital e adicionando outros

[14] Disponível em: https://balsamiq.com/. Acesso em: 06 jun. 2023.

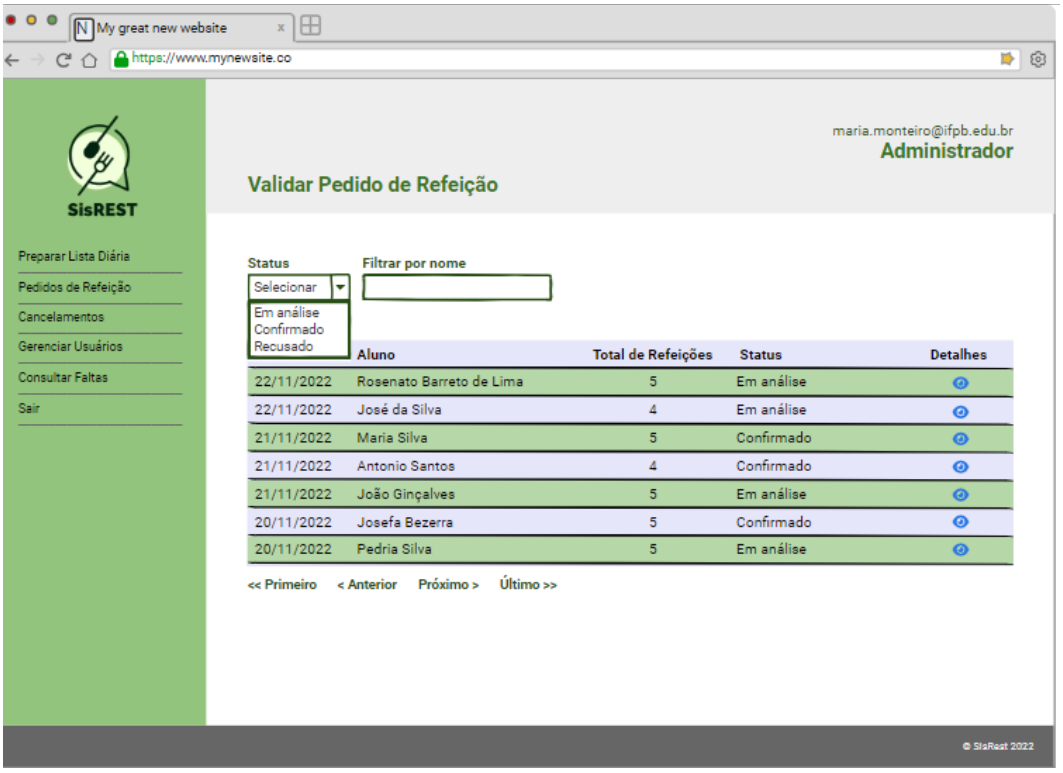
nutricionistas e assistentes sociais, entre outras opções no menu principal (a tela da Figura 6, contém um menu à esquerda que ilustra um assistente social que é administrador).

Figura 5 - Tela de Fazer Login



Fonte: Os autores.

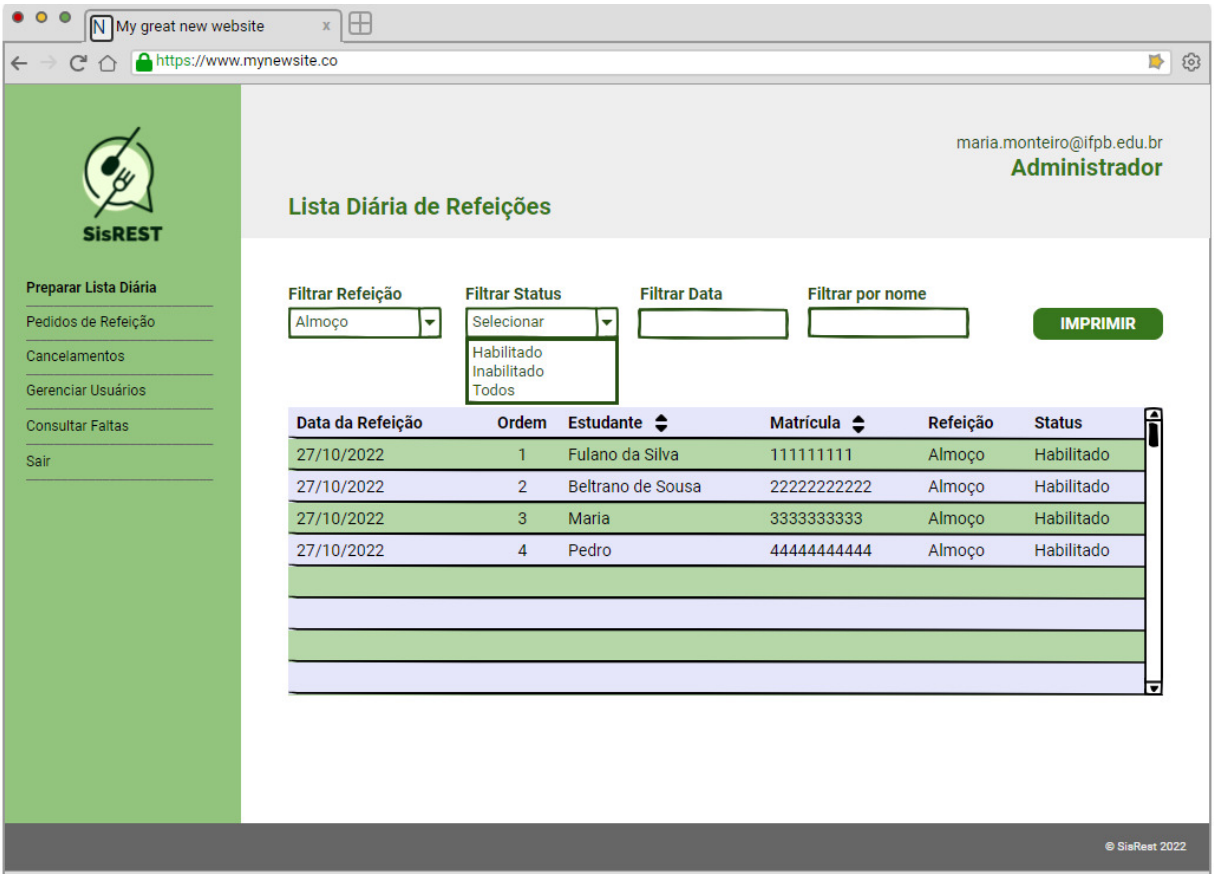
Figura 6 – Tela de Validações de Pedido



Fonte: Os autores.

Ainda no módulo do “Assistente Social” há a interface para o gerenciamento da lista diária de beneficiários, entre os estudantes que terão acesso às refeições em um determinado dia (Figura 7), sendo possível filtrá-los pelo tipo de refeição e por ordem alfabética, também.

Figura 7 – Tela Lista Diária de Refeições

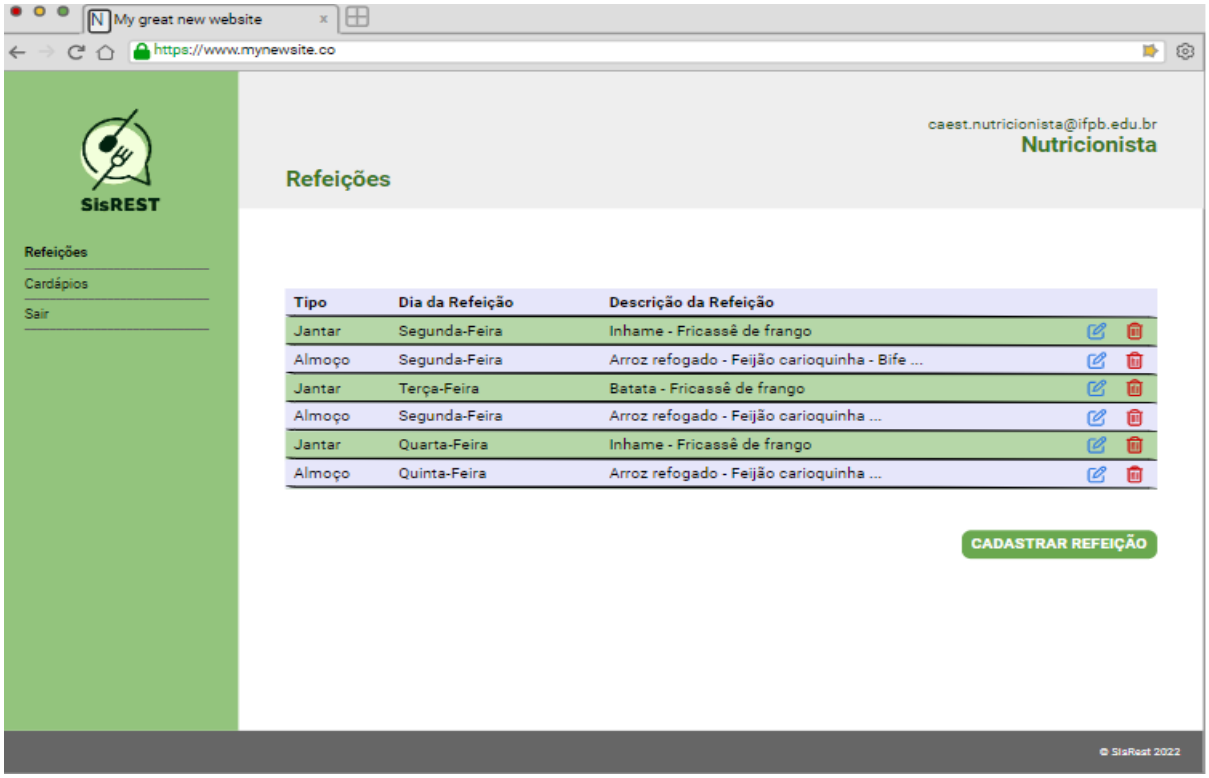


Fonte: Os autores.

No módulo do usuário “Nutricionista”, logo após ele se autenticar, ele terá acesso ao planejamento de cardápios. Em seu módulo é possível realizar um cadastro reutilizável e detalhado sobre os pratos (nomeadamente “refeições”, vide Figura 8), incluindo a sua designação para pessoas com restrições alimentares.

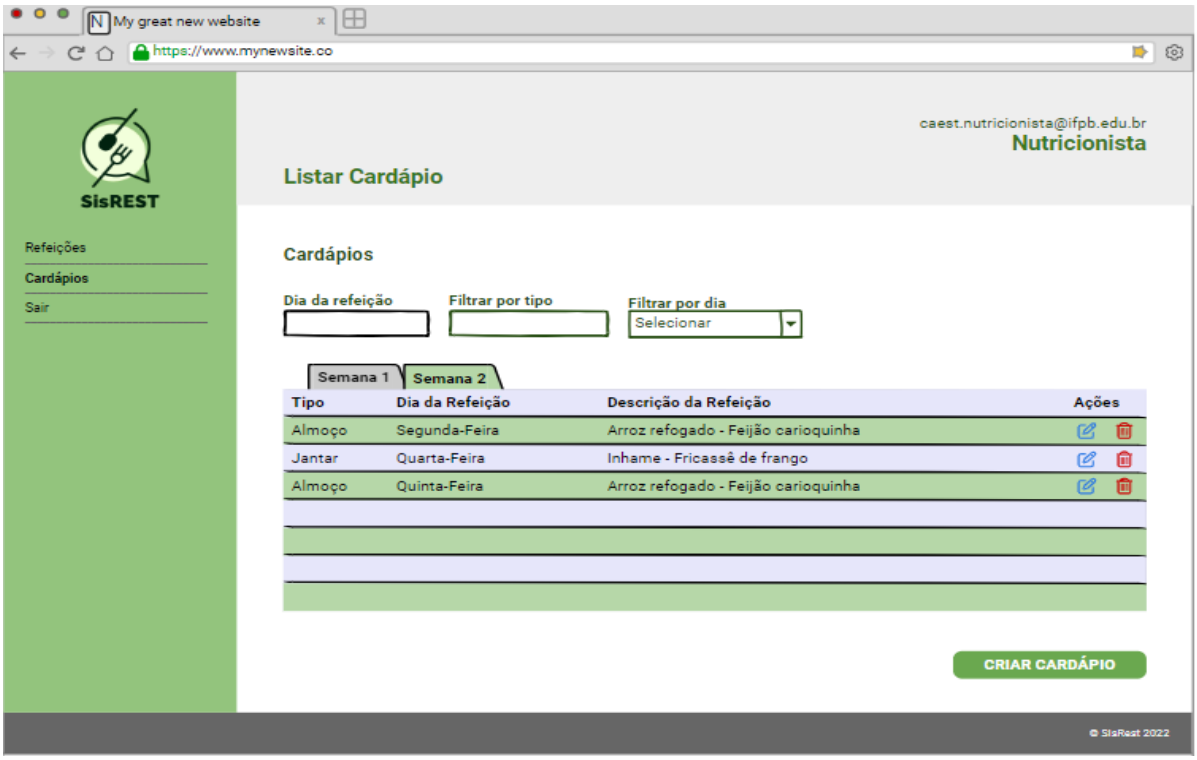
Também é possível a criação de cardápios semanais (vide abas na tela da Figura 9), dispondo-se a sequência cíclica vigente em um edital. Ainda, um nutricionista, para cada dia da semana, pode atribuir quais as refeições do dia, entre almoço, jantar, etc.

Figura 8 – Tela Refeições



Fonte: Os autores.

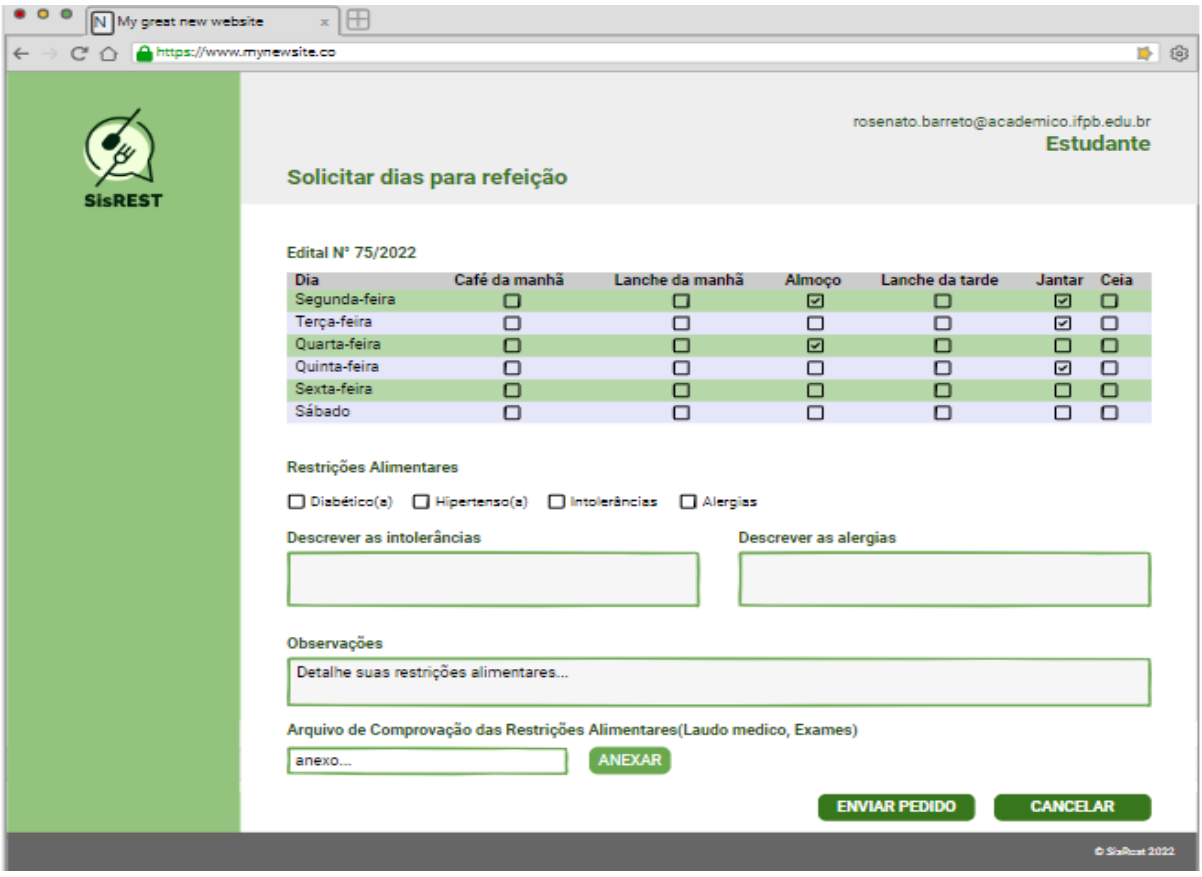
Figura 9 – Tela Listar Cardápios Semanais



Fonte: Os autores.

Por fim, para o módulo “Estudante”, logo que se autenticar no SisRest, ele pode realizar a solicitação de pedidos de acesso às refeições (Figura 10), podendo designar quais tipos de refeições do dia que necessita entre os dias da semana, assim como as suas restrições alimentares. As restrições exigem a anexação de um arquivo correspondente ao laudo de médico ou outro profissional de saúde, cuja conformidade é analisada por um “Nutricionista”. Complementarmente, o “Assistente Social”, verifica a disponibilidade dos dias e refeições pela capacidade de fornecimento do restaurante, aprovando ou não cada solicitação de pedido de acesso à refeição.

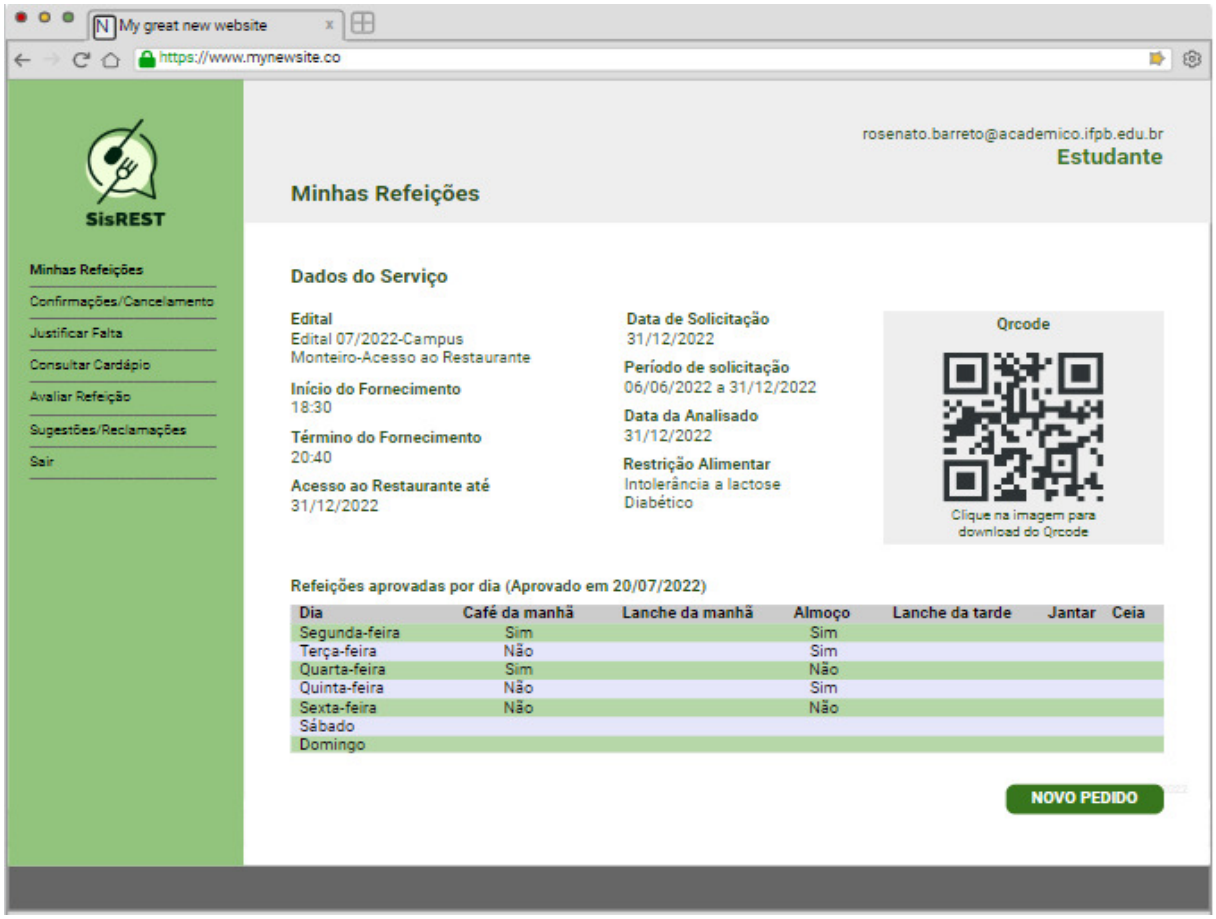
Figura 10 – Tela solicitar dias de refeição num edital



Fonte: Os autores.

Ainda destacando outras funcionalidades do módulo “Estudante” (Figura 11, menu à esquerda), ele também pode acessar os dados de sua concessão (em um edital vigente) e o QR-Code que o identifica, que pode ser lido toda vez em que utilizar o restaurante, facilitando o seu registro de presença numa refeição do restaurante.

Figura 11 – Tela “Minhas Refeições”



Fonte: Os autores.

Até então nesta seção as telas apresentadas correspondem aquelas em tempo de prototipação. O aspecto das telas equivalentes finais, após a implementação do código da aplicação, pode ser verificado nos apêndices deste trabalho.

3.5 PROJETO GERENCIAL DO SOFTWARE

A aplicação SisRest foi desenvolvida utilizando uma metodologia baseada em princípios do método Scrum.

De acordo com Wazlawick (2013), esses princípios são consistentes com o manifesto Ágil e são usados para orientar as atividades de desenvolvimento dentro de um processo que incorpora as seguintes atividades de desenvolvimento: requisitos, análise, projeto, evolução e entrega. Criado por Schwaber e Sutherland, no Scrum ocorre a segmentação do desenvolvimento do projeto em interações (*sprints*). Cada iteração objetiva atingir um bloco de entregáveis do projeto, priorizados a partir de

uma lista prévia que engloba a obtenção de todo o produto de software (*product backlog*).

O plano de projeto do SisRest envolveu duas grandes fases. A primeira, a de concepção, utilizou um método evolucionário baseado em prototipação de software por modelos, incluindo a própria interface com o usuário, com provas de conceito em código. A segunda fase, envolveu a entrega de pacotes de funcionalidades, num método incremental baseado em Scrum, em 06 *sprints*, cada uma 15 dias em média. A cada iteração os entregáveis (Quadro 4) eram verificados junto ao cliente.

Quadro 4 – Plano de iterações do SisRest

ITERAÇÃO	DURAÇÃO EM DIAS	ENTREGÁVEIS
1	15	Lógica do CRUD de estudante beneficiário
		Telas do CRUD de usuário beneficiário
		Telas do CRUD de editais
		Testes da iteração
		Autenticação de usuários com Google
2	15	Tela de importação em lote de estudantes beneficiários de um edital
		Lógica de importação em lote de estudantes beneficiários por arquivo CSV do SUAP de beneficiários sob um edital de concessão.
		Lógica do CRUD de estudante beneficiário (continuação)
		Testes da iteração
3	15	Atualização da lógica de importação em lote de estudantes beneficiários adicionando segundo arquivo CSV do SUAP para reconhecimento de e-mails e matriculas de estudantes.
		Telas do CRUD de servidores (assistentes sociais e nutricionistas)
		Lógica do CRUD de servidores (assistentes sociais e nutricionistas)
		Testes da iteração
4	20	Telas do CRUD de cardápio semanal e do catálogo de refeições
		Lógica do CRUD de cardápio semanal e do catálogo de refeições
		Controle de acesso do usuário autenticado aos módulos do sistema
		Telas e lógica para solicitação de dias de acesso com registro de restrições alimentares do estudante em um edital
		Ajustes nos métodos de filtragem de cadastros
		Testes da iteração
5	15	Telas e lógica para aprovação de dias de acesso e indicação de restrições alimentares do estudante em um edital
		Telas e lógica para consultar cardápio semanal rotativo
		Telas e lógica para consultar dias de acesso de estudantes
6	15	Ajustes e melhorias para consolidação das entregas já realizadas

Fonte: Os autores.

Um método adaptado do Kanban foi empregado para descrição, atribuição de responsáveis e controle do estado de atividades (*tasks*), considerando cada

entregável do quadro acima. Um documento eletrônico foi utilizado, dispondo-se equivalentemente as listas de cartões de tarefas, marcando-as como com os estados de realização “atribuída”, “pendente” e “concluída”.

Esse controle da realização do sistema também foi realizado a partir da marcação das classes no diagrama UML já apresentado, com um esquema de cores para determinar se todo o código e testes envolvendo cada classe foi apropriadamente finalizado: verde para “concluído”; amarelo para “em desenvolvimento” e vermelho para “em débito técnico”. A equipe aproveitou bastante esse controle para dialogar com mais precisão o saneamento desses débitos técnicos, eventualmente dispondo comentários em UML sobre os elementos do diagrama para saneá-los.

3.6 IMPLEMENTAÇÃO DO PROTÓTIPO

A implementação do SisRest será descrita primeiramente pelos detalhes no projeto de *back-end*, cujo projeto de código foi gerado e configurado com auxílio da ferramenta online Spring Initializr[15], a partir da seleção das dependências necessárias para o desenvolvimento do software.

Inicialmente foram implementadas as classes equivalentes às entidades do negócio e mapeadas com anotações JPA. Na sequência, eram desenvolvidos os objetos DAO sob a interface “Repository” do *framework* Spring Data JPA, as fachadas de serviço com a lógica de negócios envolvendo a entidade e por fim as classes de serviço com os respectivos mapeamentos de requisição REST, cuja implementação é cliente das fachadas.

Considerando o tamanho do *back-end*, foram desenvolvidas 15 classes de negócio e, respectivamente, na mesma quantidade, as classes correlatas DAO do tipo “Repository”, assim como as classes de fachadas de lógica da aplicação (“services”) e as classes de serviço REST (“resources”).

Destaca-se que a necessidade de autenticação no SisRest a partir de contas institucionais do IFPB com o Google foi primeiramente experimentada com um componente importado como dependência no *front-end* React.js, denominado “React Google Login”[16] . Depois, verificou-se a necessidade de verificação da autenticidade ocorrer toda por meio do próprio *back-end*, para maior segurança, ele mesmo

[15] Disponível em: https://start.spring.io/. Acesso em: 08 mar. 2023.
[16] Disponível em: https://www.npmjs.com/package/@react-oauth/google. Acesso em: 09 mar. 2023.

acessando o serviço do Google sob OAuth2 diretamente, a partir da geração de uma chave de aplicação para aplicação SisRest. O usuário passou então a acessar uma tela redirecionada de autenticação e autorização do próprio provedor. Essa configuração foi codificada na classe “OAuth2RedirectHandler”.

Outro aspecto peculiar da implementação do SisRest é a importação de arquivos CSV do sistema SUAP, considerando a relação entre uma listagem de matrículas de estudantes com os seus e-mails e a listagem de estudantes contemplados em um mesmo edital. Durante a implementação houve uma dificuldade inicial para o cruzamento dessas informações, que facilitam a inclusão de beneficiários de um edital e as respectivas contas aptas de acesso pelo e-mail institucional. A biblioteca gratuita OpenCSV[17] foi adicionada como dependência do *back-end*, a qual possui uma API que facilita o percurso sobre registros de um arquivo CSV, ou seja, a partir de cada dado separado por vírgula, de cada linha de registro, sendo a primeira linha a designadora do nome dos atributos.

Posteriormente, esse procedimento de leitura e importação foi substituído pelo Spring Batch[18], que faz parte do *framework* Spring, que contém as dependências e toda uma API pronta para a leitura e processamento em lotes de diversos formatos de arquivos, incluindo CSV. Destaca-se que a disposição do código que ficou mais coesa com o *framework*, pois ela exigiu basicamente a implementação de um objeto “JobLauncher”, que lança uma “Job” (ou seja do “trabalho”, cujo o código de configuração é ilustrado na Figura 12) de processamento de transformação de dados. Esse, por sua vez, é constituído de uma ou mais “Steps” (etapas), cada uma dispondo os códigos para: (i) ler registros de uma fonte de dados em lote (objeto “ItemReader”); (ii) processar a sua transformação no formato de destino (“ItemProcessor”); e (iii) efetivamente carregar dados transformados num repositório de destino (“ItemWriter”).

A “Job” do SisRest foi configurada com o nome “importarBeneficiarios”, em seguida, configurando-se (Figura 12): (i) na linha 03, o autoincremento de identificadores de registros processados; (ii) na linha 04, a chamada à execução da primeira “Step”, a partir do método “contaEstudanteStep”, que importa o CSV equivalente a lista de estudantes matriculados com os seus e-mails; (iii) na linha 05, a designação de em havendo a conclusão com êxito, para que se proceda com a execução da segunda “Step”, via método “beneficiarioStep”, que equivale ao

[17] Disponível em: https://opencsv.sourceforge.net/. Acesso em: 26 fev. 2023.
[18] Disponível em: https://spring.io/projects/spring-batch. Acesso em: 27 fev. 2023.

processamento do CSV da lista de estudantes beneficiários num edital, cruzando-os com aqueles dados dos registros já importados na “Job” anterior.

Figura 12 – “Job” Spring Batch (importação de estudantes beneficiários do SUAP)

```
1. public Job job() {
2. return jobBuilderFactory.get("importarBeneficiarios")
3.             .incrementer(new RunIdIncrementer())
4.             .flow(contaEstudanteStep())
5.             .on("COMPLETED").to(beneficiarioStep())
6.             .end()
7.             .build();
8. }
```

Fonte: Os autores.

Quanto ao gerenciamento do código e controle de versão do código do projeto SisRest, foi utilizado o serviço gratuito do GitHub, havendo um projeto para o *back-end* [19] e outro para o *front-end* [20] .Até a primeira versão foram realizados, respectivamente, 40 e 186 *commits* nos mesmos.

No início do desenvolvimento do *back-end* foram usadas apenas 03 *branches*, a principal “main” e mais duas outras, uma para cada desenvolvedor, em que ao final era realizada a operação de mesclagem (*merge*) do código para a *branch* principal.

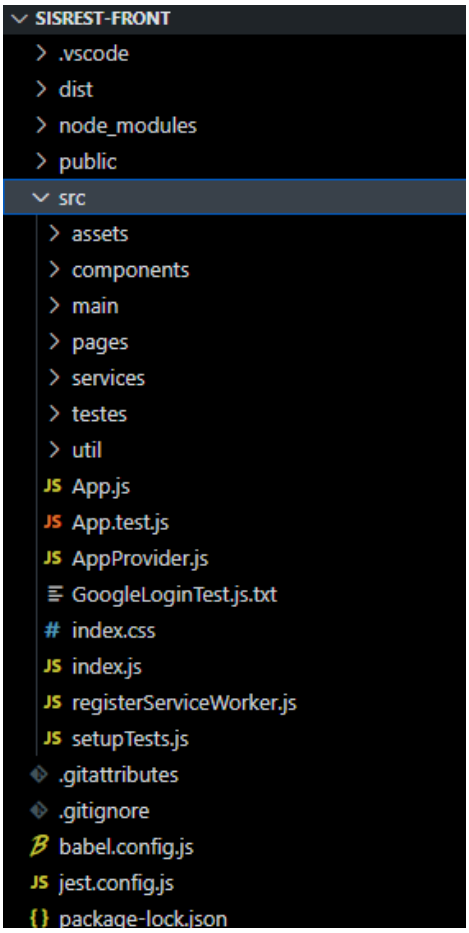
Devido a alguns conflitos nessas *branches*, ocasionalmente se perdiam algumas das alterações entre elas nesse processo. Foi pensada uma forma melhor de criar *branches* por *features*, de forma que, logo que surgia uma nova funcionalidade a ser desenvolvida, um nova *branch* era criada com o nome da *feature* nova. Foi possível com isso resolver esses conflitos, melhorando o desenvolvimento em paralelo, mas, isso envolveu também uma mudança nas operações de *merge*. No lugar de fazê-las sob a *branch* principal, passou-se a realizar uma operação de *pull request*, que se desdobra em uma requisição de *merge*, permitindo uma janela de análise de possíveis conflitos no código, efetivando-se o *merge* somente quando esses conflitos estivessem saneados.

No desenvolvimento do *front-end* foi utilizada apenas a *branch* principal “main”, considerando que havia apenas um desenvolvedor dedicado ao mesmo. O código desse projeto em React.js foi inicialmente gerado por meio da ferramenta Create

[19] Link do repositório do *back-end*: https://github.com/santospatricia11/SisRest.
[20] Link do repositório do *front-end*: https://github.com/rosenatobarreto/SisREST-Front.

React App[21], que dispõe toda estrutura básica de código e de dependências. Foram sendo desenvolvidos os demais *scripts* de código com base em equivalências com o projeto preestabelecido no diagrama de componentes já apresentado do *front-end* (Figura 4), refletida na distribuição de pastas obtida com o IDE Visual Code Studio (Figura 13).

Figura 13 – Estrutura de pastas do projeto de código do *front-end*



Fonte: Os autores.

3.7 PROJETO E EXECUÇÃO DE TESTES E VERIFICAÇÃO DE QUALIDADE

A verificação da qualidade sob da aplicação *web* SisRest foi projetada desde a primeira iteração, focando-se na realização de testes unitários automáticos com o JUnit (Tabela 1) sob as fachadas de serviços, objetivando a verificação de erros básicos no cadastro de entidades.

[21] Disponível em: https://create-react-app.dev/. Acesso em: 27 fev. 2023.

Tabela 1 – Relação de suítes de testes unitários automatizados com JUnit

SUÍTE DE TESTES	OBSERVAÇÕES	TESTES
Beneficiários	Entradas de dados válidas, nulas e inválidas.	12
Cardápio Semanal	Entradas inválidas, nulas e ciclos de atualizações.	5
Lista Diária	Formato da lista e presença de dados inválidos.	9
Pedido de Acesso	Entradas de dados válidas, nulas e inválidas.	8
Restrição Alimentar	Entradas de dados válidas, nulas e inválidas.	5
Conta	Verificação de credenciais válidas e inválidas.	7

Fonte: Os autores.

Também foram realizados testes de consumo dos *endpoints* da API REST da aplicação, consumindo, em separado, cada uma das requisições, a partir de simulações com a ferramenta Postman[22]. Essas simulações serviram de apoio aos desenvolvedores do *front-end* sobre como as requisições deveriam ser construídas e como seriam as respostas esperadas contendo os respectivos dados.

Complementarmente, foram realizados testes de aceitação automatizados com o Selenium (Tabela 2), a fim de detectar a adequação funcional da aplicação.

Tabela 2 – Relação de testes de aceitação automatizados com Selenium

MÓDULO	OBSERVAÇÕES	SUÍTES DE TESTES
Assistente Social	Simulação de *login*, entradas válidas e inválidas em cadastros diversos.	6
Beneficiário	Simulação de *login*, entradas válidas e inválidas em cadastros diversos.	5
Nutricionista	Simulação de *login*, entradas válidas e inválidas sob o cardápio de refeições.	5

Fonte: Os autores.

Por fim, mais alguns testes de aceitação manuais foram realizados com o representante do cliente, a partir da manipulação da interface com o usuário, a fim de se verificar a sua adequação funcional e a sua usabilidade, em que eram coletadas demandas de ajustes, que logo foram saneadas na iteração final do desenvolvimento.

[22] Disponível em: https://learning.postman.com/. Acesso em: 05 abr. 2023.

4 CONSIDERAÇÕES FINAIS

Neste trabalho foi proposto e apresentada as características de uma aplicação *web* para o gerenciamento do restaurante estudantil do IFPB Campus Monteiro, o SisRest.

A aplicação obtida conseguiu integrar, em um só ferramenta, os serviços para facilitação da geração e controle da lista de acesso diário de estudantes ao restaurante, com base na importação de beneficiários e de dados de matrículas oriundas de módulos do SUAP, com possibilidade de habilitação ou inabilitação de seus acessos.

Complementarmente, o registro de informações nutricionais do catálogo de refeições do restaurante pode ser facilmente disposto e reutilizado a partir de um sistema de rodízio, a partir da configuração de cardápios semanais numa sequência, dentro do escopo de um edital. Ainda, o estudante beneficiário pode solicitar a aprovação dos dias da semana e respectivas refeições do dia em que obterá acesso, logo no início de um edital, mantendo-se a comprovação dos dados acerca de restrições alimentares, como intolerâncias e alergias, a fim de que o serviço de restaurante possa designar refeições compatíveis.

Ainda, a versão obtida do SisRest foi apropriadamente convalidada com os representantes do setor do campus responsável pela gestão do restaurante, a CAEST.

4.1 TRABALHOS FUTUROS

Considerando esta primeira versão do SisRest, faz-se necessário o desenvolvimento de algumas funcionalidades que não puderam ser implementadas devido ao tempo disponível, porém elas foram apropriadamente projetadas neste trabalho.

Inclusive, isso abrange a realização de um teste de homologação completo, em caráter experimental, aplicando-se o SisRest na rotina do restaurante do campus, a fim de verificar a sua adequação em funcionamento, realizando-se prontamente a correção de eventuais falhas remanescentes.

Destacam-se os seguintes recursos dentre aqueles pendentes, visando a disponibilização apropriada do SisRest para o seu público final:

i. A geração do QRcode para facilitar no registro de acessos de estudantes ao restaurante;
ii. A avaliação de refeições, considerando comparecimento de beneficiários na constatação de suas presenças no restaurante;
iii. O registro, justificativa e consulta de faltas de beneficiários;
iv. O registro e notificação sobre o estado em filas de espera, em tempo real, considerando redistribuição de refeições não utilizadas em um dia de serviço do restaurante.

REFERÊNCIAS

BRASIL. **Lei nº 11.947, de 17 de junho de 2009.** Dispõe sobre o atendimento da alimentação escolar e do Programa Dinheiro Direto na Escola aos alunos da educação básica. Disponível em: https://www.planalto.gov.br/ccivil_03/_ato2007-2010/2009/lei/l11947.htm. Acesso em: 22 jun. 2023.

IFPB. **Sobre o SUAP**. 27 set. 2017. Disponível em: https://www.ifpb.edu.br/ti/assuntos/catalogo-de-servicos/SUAP/copy_of_sobre-o-SUAP. Acesso em: 08 mar. 2023.

IFPB. **Resolução-CS nº 16/2018, de 17 de agosto de 2018**. Política de Assuntos Estudantis do IFPB. Disponível em: https://www.ifpb.edu.br/orgaoscolegiados/consuper/resolucoes/2018/resolucoes-aprovadas-pelo-colegiado/resolucao-no-16/view. Acesso em: 03 mar. 2023.

IFPB. Edital nº 01/2023, de 22 de fevereiro de 2023 – Programa de Alimentação (Campus Monteiro). **IFPB**, Monteiro, Paraíba, 2023. Disponível em: https://www.ifpb.edu.br/monteiro/editais/direcao-geral/2023/edital-no-01-2023/edital-no-01-2023-programa-de-alimentacao.pdf/view. Acesso em: 25 ago. 2023.

ISO. **ISO/IEC 9126-1:2001**: Software engineering - Product quality - Part 1: Quality model, jun. 2001. Disponível em: https://www.iso.org/standard/22749.html. Acesso em: 25 set. 2023.

MEC. Conselho Deliberativo do FNDE. **Resolução nº 06, de 08 de maio de 2020**. Dispõe sobre o atendimento da alimentação escolar aos alunos da educação básica no âmbito do Programa Nacional de Alimentação Escolar – PNAE. Disponível em: https://www.gov.br/fnde/pt-br/acesso-a-informacao/legislacao/resolucoes/2020/resolucao-no-6-de-08-de-maio-de-2020/@@download/file. Acesso em: 27 jul. 2023.

RICHARDS, M.; FORD, N. **Fundamentals of Software Architecture**: An Engineering Approach. O'Reilly Media, 2020. ISBN 9781492043423.

SEBRAE. **Aprenda sobre o quadro Lean Canvas e comece sua startup**. 29 mar. 2019. Disponível em: https://sebrae.com.br/sites/PortalSebrae/ufs/pb/artigos/aprenda-sobre-o-quadro-lean-canvas-e-comece-sua-startup,08c7190f394c9610VgnVCM1000004c00210aRCRD. Acesso em: 27 jul. 2023.

SINGH, R. Spring Boot OAuth2 Social Login with Google, Facebook, and Github - Part 1. **Blog Callicoder**, 06 nov. 2018. Disponível em: https://www.callicoder.com/spring-boot-security-oauth2-social-login-part-1/>. Acesso em: 21 fev. 2023.

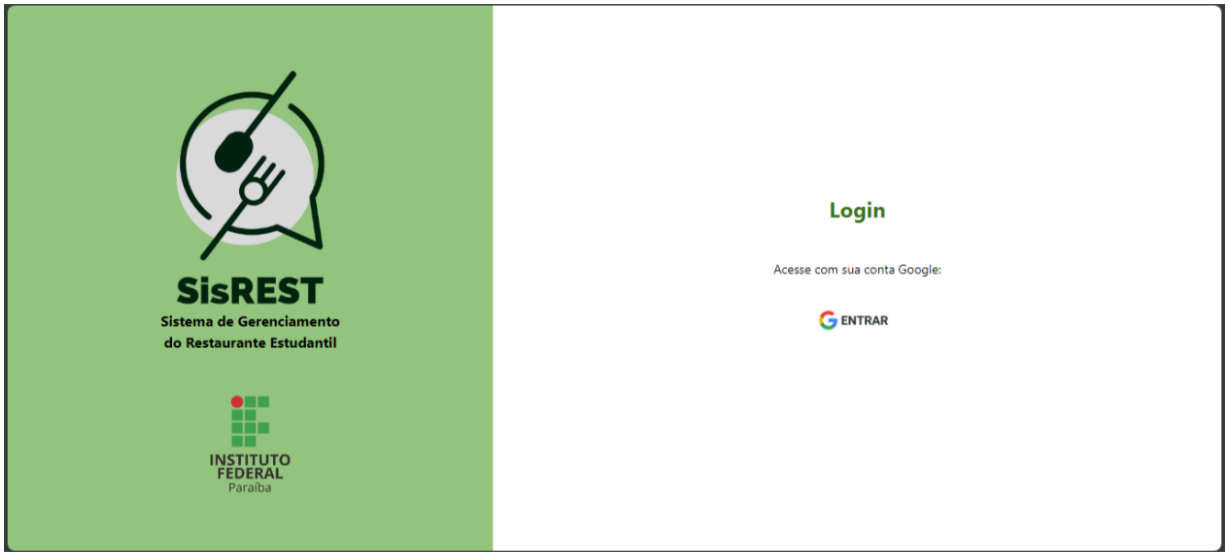
SOMMERVILLE, I. **Engenharia de Software**. 9. ed. Tradução Ivan Bosnic e Kamila de O. Gonçalves. São Paulo: Pearson Prentice Hall, 2011.

SPRING. **Spring Boot**. Disponível em: https://spring.io/projects/spring-boot. Acesso em: 27 de jul. de 2023.

WAZLAWICK, R. S. **Engenharia de software**: conceitos e práticas. Rio de Janeiro: Elsevier, 2013.

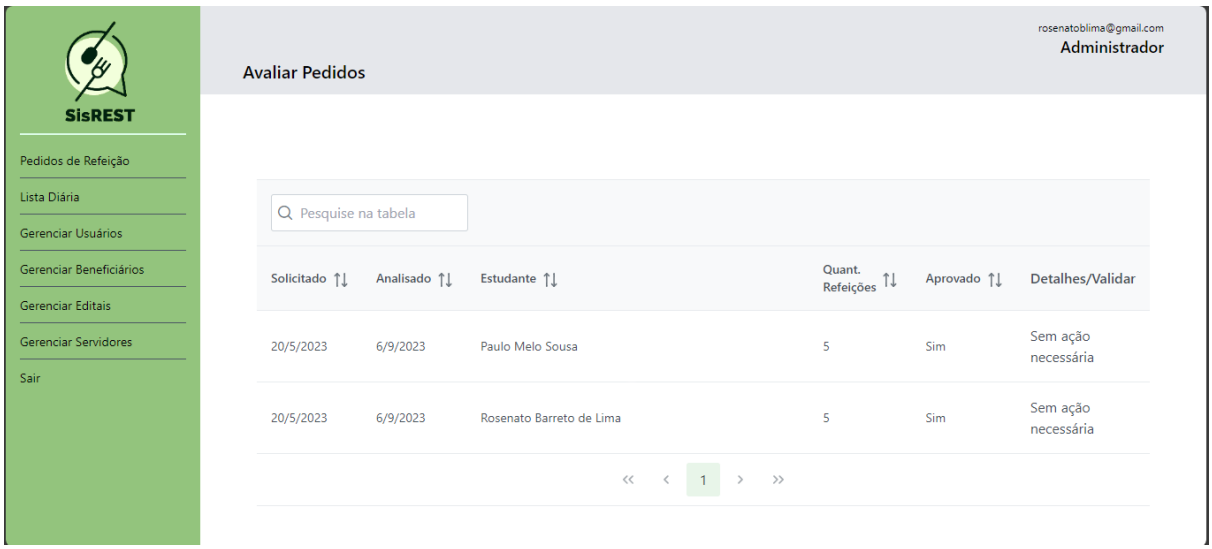
APÊNDICE A – TELAS DO MÓDULO ASSISTENTE SOCIAL

Figura 14 - Tela de fazer *login*



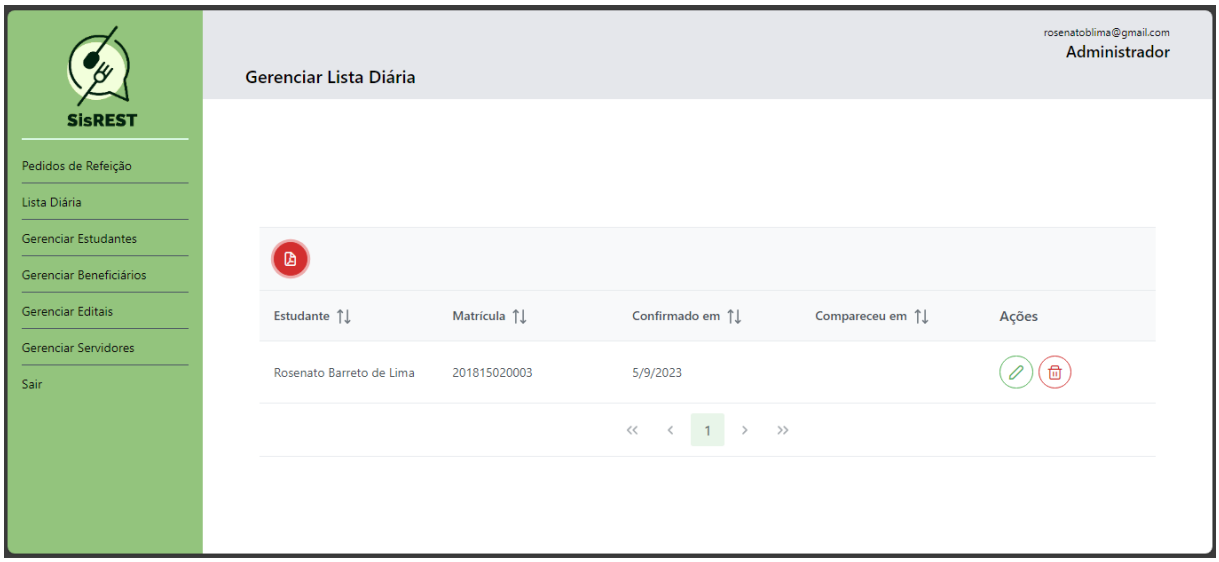
Fonte: Os autores.

Figura 15 – Tela opção menu “Pedidos de Refeição”



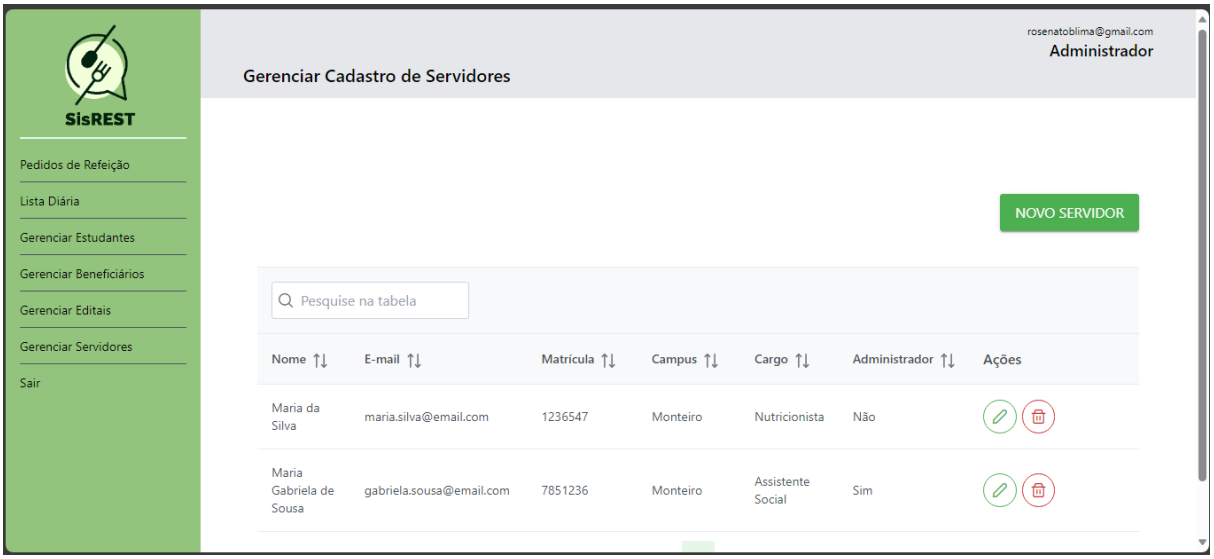
Fonte: Os autores.

Figura 16 – Tela opção menu “Lista diária” (de refeições)



Fonte: Os autores.

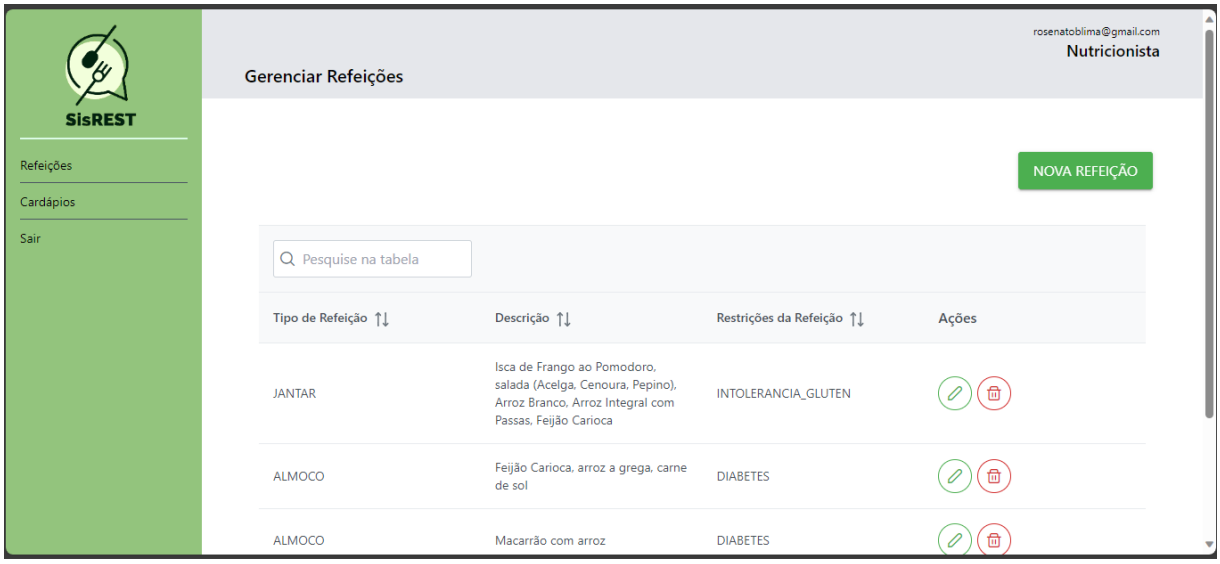
Figura 17 – Tela opção menu “Gerenciar Servidores” (nutricionistas e assistentes sociais)



Fonte: Os autores.

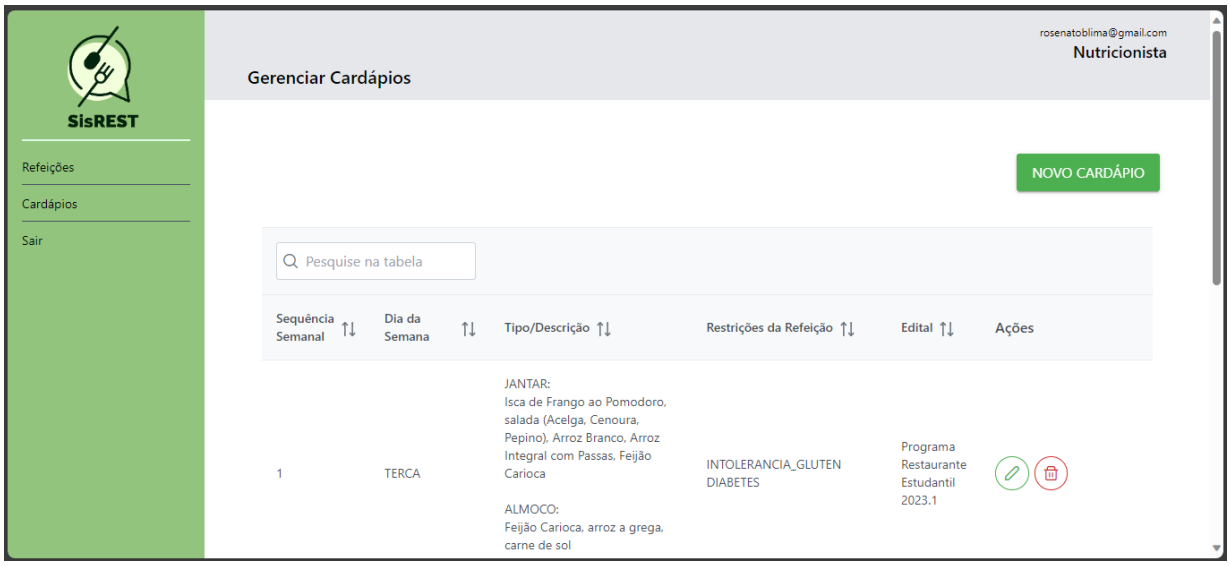
APÊNDICE B - TELAS DO MÓDULO NUTRICIONISTA

Figura 18 – Tela Refeições



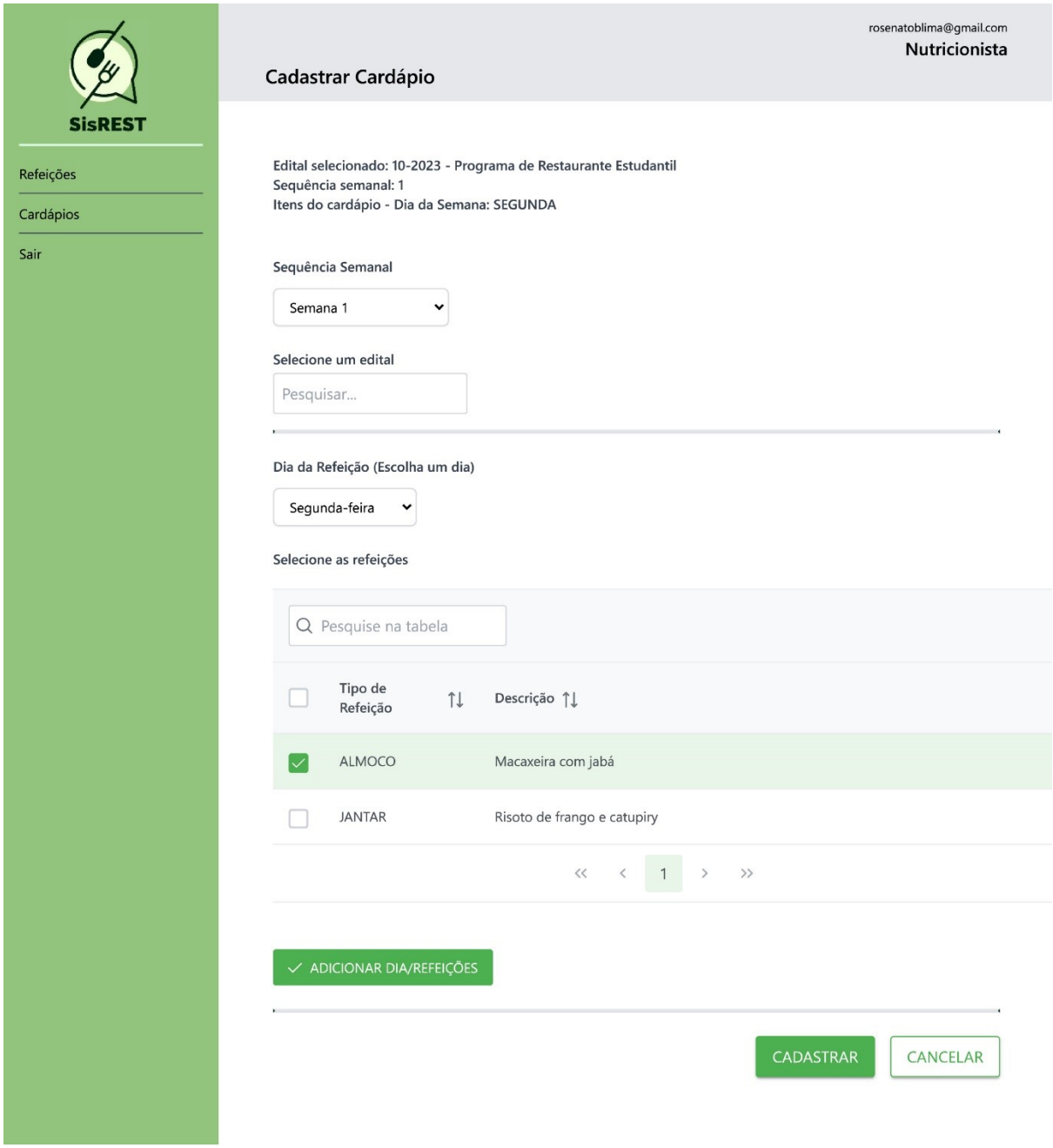
Fonte: Os autores.

Figura 19 – Tela cardápios semanais



Fonte: Os autores.

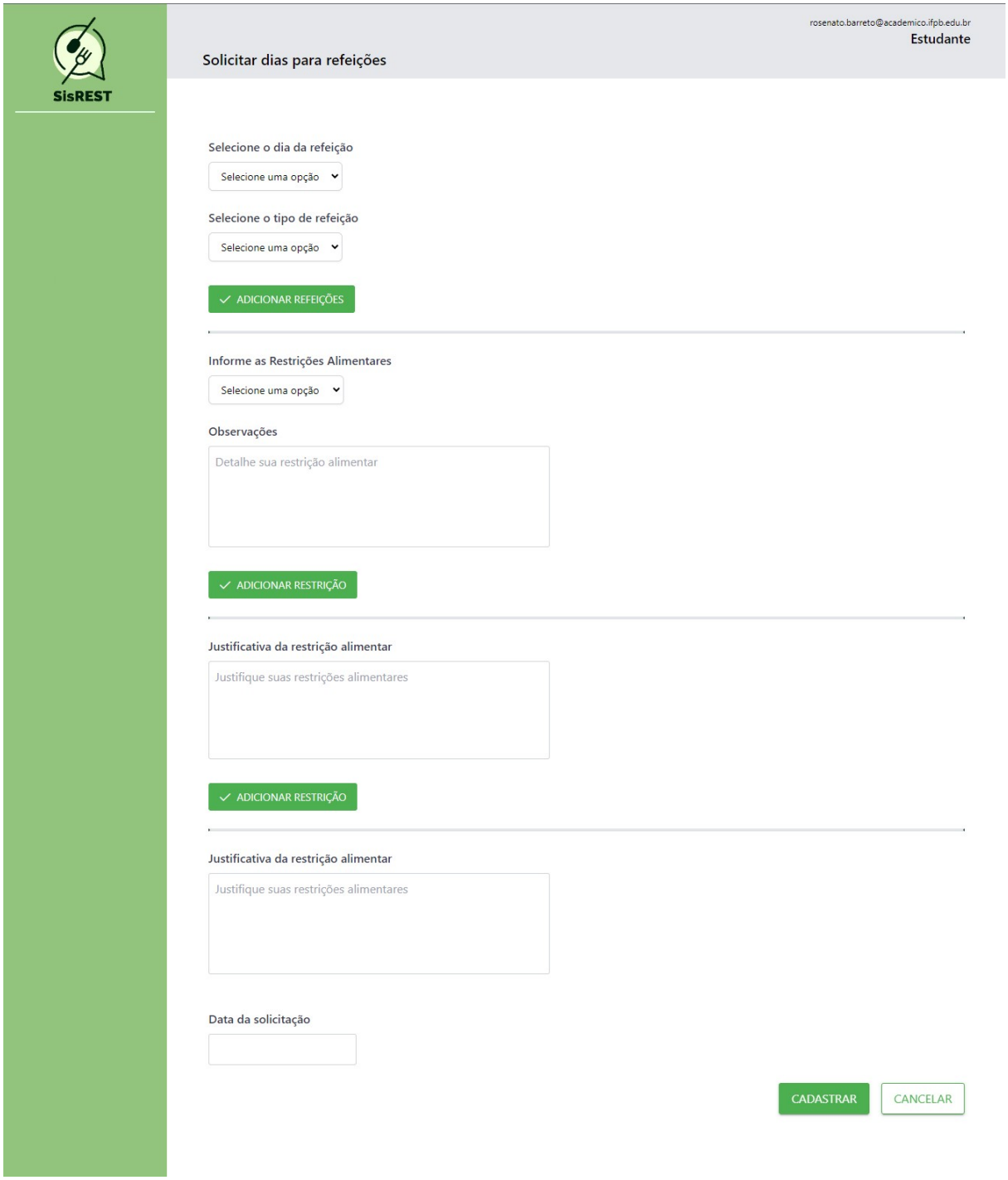
Figura 20 – Tela “Cadastrar Cardápio” do dia da semana em edital (numa sequência semanal)



Fonte: Os autores.

APÊNDICE C – TELAS DO MÓDULO ESTUDANTE

Figura 21 – Tela “Solicitar dias para refeições” num edital e restrições alimentares



Fonte: Os autores.

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA
Campus Monteiro
Pb-264, S/N, Serrote, CEP 58500-000, Monteiro (PB)
CNPJ: 10.783.898/0008-41 - Telefone: (83) 3351-3700

Documento Digitalizado Ostensivo (Público)

TCC

Assunto: TCC
Assinado por: Gabriel Oliveira
Tipo do Documento: Anexo
Situação: Finalizado
Nível de Acesso: Ostensivo (Público)
Tipo do Conferência: Cópia Simples

Documento assinado eletronicamente por:

- **Gabriel Oliveira Florencio da Silva**, **ALUNO (201715020037) DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS - MONTEIRO**, em 07/11/2023 21:27:16.

Este documento foi armazenado no SUAP em 07/11/2023. Para comprovar sua integridade, faça a leitura do QRCode ao lado ou acesse https://suap.ifpb.edu.br/verificar-documento-externo/ e forneça os dados abaixo:

Código Verificador: 988682
Código de Autenticação: 839f88eb93