UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
PRÓ-REITORIA DE GRADUAÇÃO
CENTRO MULTIDISCIPLINAR DE ANGICOS
CURSO DE LICENCIATURA EM COMPUTAÇÃO E INFORMÁTICA

JOÃO VITOR COSTA CARDOSO

# IMPLEMENTAÇÃO DE SISTEMA WEB PARA CONTROLE DE RESTAURANTE UNIVERSITÁRIO

ANGICOS
2021

JOÃO VITOR COSTA CARDOSO

# IMPLEMENTAÇÃO DE SISTEMA WEB PARA CONTROLE DE RESTAURANTE UNIVERSITÁRIO

Monografia apresentada a Universidade Federal Rural do Semi-Árido como requisito para obtenção do título de Licenciado em Computação e Informática, sob orientação do Prof. Dr. Francisco de Assis P. V. de Arruda.

ANGICOS

2021

C268i Cardoso, João Vitor Costa.
Implementação de sistema web para controle de restaurante universitário / João Vitor Costa Cardoso. - 2021.
46 f. : il.

Orientador: Francisco De Assis Pereira Vasconcelos De Arruda.
Monografia (graduação) - Universidade Federal Rural do Semi-árido, Curso de Computação e Informática, 2021.

1. Especificação de requisitos de software. 2. Desenvolvimento de sistema web. 3. Restaurante universitário. 4. Integração contínua. 5. Implantação contínua. I. Arruda, Francisco De Assis Pereira Vasconcelos De, orient. II. Título.

JOÃO VITOR COSTA CARDOSO

# IMPLEMENTAÇÃO DE SISTEMA WEB PARA CONTROLE DE RESTAURANTE UNIVERSITÁRIO

Monografia apresentada a Universidade Federal Rural do Semi-Árido como requisito para obtenção do título de Licenciado em Computação e Informática.

Defendida em: 31/06/2021.

**BANCA EXAMINADORA**

Assinado digitalmente por FRANCISCO DE ASSIS PEREIRA VASCONCELOS DE ARRUDA:04700207418
Localização: CMA - UFERSA
Data: 2021.06.08 10:46:34-03'00'

Prof. Dr. Francisco de Assis Pereira Vasconcelos de Arruda (UFERSA)
Orientador-Presidente

Thatiana Cunha Navarro Diniz

Assinado de forma digital por THATIANA CUNHA NAVARRO DINIZ:05332814402
Dados: 2021.06.09 15:59:50 -03'00'

Profa. Dra. Thatiana Cunha Navarro Diniz (UFERSA)
Membro Examinador

SARA GUIMARAES NEGREIROS: 11051711401

Digitalmente assinado por: SARA GUIMARAES NEGREIROS:11051711401
E-mail: SGUIMARAAES@GMAIL.COM
Data: 2021/06/09 19:36:37 -03'00'

Profa. Sara Guimarães Negreiros (UFERSA)
Membro Examinador

## DEDICATÓRIA

Dedico este trabalho para aquela que com seu imenso amor de mãe me motiva a lutar por cada sonho estabelecido, minha avó paterna, Neusa Felipe da Silva.

## AGRADECIMENTOS

Agradeço a Deus pelo discernimento concedido em momentos de incertezas e dias difíceis.

Agradeço ao meu orientador de trabalho de conclusão de curso II, professor Francisco de Assis Pereira Vasconcelos de Arruda, pelas orientações dadas para a realização desse trabalho.

Agradeço ao professor José Gildo de Araújo Júnior, pelas valiosas e ímpares orientações no trabalho de conclusão de curso I, pela publicação do nosso artigo científico, resultado desse trabalho, pelas histórias de vida contadas durante suas aulas, pelos conselhos dados em sua sala durante nossas reuniões e pelo conhecimento técnico ensinado em suas aulas.

Agradeço a minha família, em especial meu Pai, José Eudes Cardoso da Silva, por estar sempre comigo durante esses anos de graduação.

Agradeço a minha namorada, Francisca Larissa Barbosa da Silva, pela compreensão e palavras de apoio em momentos difíceis no processo de desenvolvimento desse trabalho.

Agradeço aos meus amigos, Gustavo de Araújo Costa e João Érico Bezerra de Sena pela parceria de sempre.

## RESUMO

O *software desktop* utilizado atualmente para gerenciar as atividades do Restaurante Universitário (RU) da Universidade Federal Rural do Semi-Árido (UFERSA) apresentou uma série de limitações referentes à manutenibilidade, versionamento e distribuição de novas versões de forma rápida, que implicam tanto na dificuldade de desenvolver novas funcionalidades quanto de manter as funcionalidades existentes. Baseando-se em tecnologias que prezam pela produtividade e interoperabilidade, neste trabalho apresentam-se as etapas de desenvolvimento do sistema *web* que se propõe a facilitar e automatizar alguns aspectos do sistema atual, garantindo assim melhor administração e controle do restaurante universitário. O Web RU foi desenvolvido utilizando-se a linguagem de programação Java em conjunto com o ecossistema de ferramentas Spring, Git e GitHub para o versionamento do código e o Heroku para a implantação do sistema em ambiente de produção, aplicando os conceitos de Integração Contínua e Implantação Contínua.



**Palavras-chave:** Especificação de requisitos de software; Desenvolvimento de sistema web; Restaurante universitário.

## ABSTRACT

The desktop software currently used to manage the Universidade Federal Rural do Semi-árido (UFERSA)'s University Restaurant (UR) presented a series of limitations related to software maintenance, versioning, and faster distribution of new versions, which results in difficulty in the development of new features and the maintenance of older features. Based on technologies that value the principles of productivity and interoperability, this work presents the steps to the development of a system that aims to facilitate and automate some aspects of the current system, grating a better administration and control to the university restaurant. The Web RU system was developed using the Java programming language with Spring tools ecosystem, Git and GitHub for code versioning, and Heroku for production deployment, also applying concepts of continuous integration and continuous deployment.



**Keywords:** Software requirements specification; Web system development; University restaurant.

## LISTA DE FIGURAS

## LISTA DE QUADROS

## LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| UFERSA | Universidade Federal Rural do Semi-Árido |
| SUTIC | Superintendência de Tecnologia da Informação e Comunicação |
| CI | *Continuous Integration* |
| CD | *Continuous Deployment* |
| RU | Restaurante Universitário |
| PROAE | Pró-reitoria de Assuntos Estudantis |

**SUMÁRIO**

# 1 INTRODUÇÃO

O restaurante universitário (RU) da Universidade Federal Rural do Semi-Árido (UFERSA) tem como objetivo oferecer refeições ao público acadêmico com qualidade e agilidade. Para isso, dispõe de recursos humanos e computacionais para garantir sempre o melhor atendimento. Dentre os recursos computacionais, utiliza um sistema de computador para gerenciar as atividades.

O sistema utilizado atualmente foi construído para utilização em formato *desktop* com sistema operacional Windows, o que se mostrou, ao longo do tempo, uma série de limitações que inviabilizam sua manutenibilidade, distribuição multicampi e aprimoramento apresentado após a evolução de versões. Essas limitações fazem com que o fluxo de atividades gerenciais do RU perca celeridade, dificultando as atividades administrativas e fiscalização contratual do compromisso firmado entre a UFERSA e a empresa vencedora da licitação para prover os serviços no espaço do restaurante.

Além das limitações apresentadas, a Superintendência de Tecnologia da Informação e Comunicação (SUTIC) da UFERSA apresenta carência de mão de obra disponível para trabalhar na evolução do sistema atualmente em uso. As demandas da SUTIC em relação ao sistema em uso atualmente foram apresentadas em um vídeo (disponível no *YouTube*[1]), no qual pode ser visto o estado atual do sistema e as melhorias pretentidas.

Isto posto, esse trabalho tem o objetivo de apresentar uma solução de *software* para o restaurante universitário utilizando uma arquitetura web (chamado Web-RU) que preserva as mesmas regras de negócio do sistema do RU atual, modificando a arquitetura (de *desktop* para *web*) e incrementando-o com automação de funcionalidades de migração de cadastro de alunos e avaliação da qualidade por parte dos usuários do serviço.

A seguir apresentam-se a justificativa, os objetivos do trabalho, o referencial teórico, os resultados obtidos e as considerações finais. No capítulo 2 consta a descrição das limitações identificadas no sistema atual utilizado para

[1] https://youtu.be/16IvSxLLtNk

gerenciar o RU. No capítulo 3 são descritos os objetivos deste trabalho. No capítulo 4 são apresentados os conceitos necessários ao entendimento do trabalho. No capítulo 5 é descrito um trabalho relacionado ao tema da pesquisa. No capítulo 6 estão descritos os materiais e métodos utilizados para o desenvolvimento do sistema *web*. No capítulo 7 é apresentado o desenvolvimento das funcionalidades do Web RU. No capítulo 8 são apresentados os resultados obtidos com este trabalho.

# 2 JUSTIFICATIVA

O sistema utilizado atualmente, concebido para a arquitetura *desktop* e implementado no ambiente *Delphi* (linguagem de programação *Object Pascal*), apresenta limitações que inviabilizam as atividades gerenciais do RU, dentre elas:

**Manutenção:** Questões referentes ao paradigma de orientação a objetos (encapsulamento, herança, polimorfismo) e padrões de projeto (*design patterns*), que dão celeridade tanto ao processo de manutenção quanto ao desenvolvimento de sistemas de *software* não foram contemplados na versão atual do sistema do RU. Como consequência direta, há a necessidade de investirem-se muitos recursos (tempo, capital humano, estrutura física) no processo de manutenção do software: localizar, entender, corrigir e testar as falhas levantadas. Quanto mais o sistema cresce (são adicionadas novas funcionalidades e há correções de funcionalidades antigas) mais complexo torna-se o processo de mantê-lo. Soma-se a isso, ainda, a carência de mão de obra apta a trabalhar com o ambiente *Delphi* no contexto atual da SUTIC.

**Geração de relatórios:** Há limitações referentes a geração de relatórios condizentes com a necessidade da universidade, em especial na fiscalização dos contratos de agentes terceirizados para o RU.

**Atualização de dados:** Limitações referente a importação dos dados, que obriga a cada semestre a realização de inserção manual dos dados dos alunos ingressantes e remoção dos alunos egressos no sistema.

**Disponibilização:** Por tratar-se de uma aplicação *desktop* executando localmente, novas versões do *software* implicam em novo empacotamento, novas instalações e configuração do ambiente, o que acaba por prolongar, em grande medida, a disponibilização de versões com novas funcionalidades ou melhorias.

**Distribuição:** A medida que o RU vai sendo estabelecido em outros campi da mesma universidade, a manutenção de um sistema *desktop*, onde cada campus possui sua própria versão do sistema e executa localmente as atividades, potencializa e inviabiliza a capacidade de manutenção por parte da SUTIC.

**Versionamento:** Há dificuldade em produzir, manter e gerir novas versões do sistema do RU.

Diante disso, apresenta-se justificável a concepção e implementação de um novo sistema que corrija os problemas atuais e que ofereça uma arquitetura capaz de facilitar a atualização do software de acordo com as demandas apresentadas pela instituição com relação ao RU.

# 3 OBJETIVOS

Nesta seção estão descritos o objetivo geral e os objetivos específicos do trabalho.

## 3.1 Objetivo geral

Desenvolver um sistema *web* que preserva as mesmas regras de negócio do sistema de RU atual, incrementando-o com a automação de funcionalidades de cadastro e avaliação da qualidade do serviço por parte dos usuários do RU.

## 3.2 Objetivos específicos

Os objetivos específicos do trabalho são:

- Especificar um sistema *web* que atenda as demandas atuais da SUTIC;
- Implementar uma funcionalidade que controle a ativação/inativação de subsídios ou bolsas com base na assiduidade dos alunos;
- Desenvolver um módulo de avaliação dos aspectos do RU por parte dos alunos.

# 4 FUNDAMENTAÇÃO TEÓRICA

Nesta seção são apresentados os conceitos utilizados como base teórica para a proposta de solução de *software* apresentada neste trabalho. Na subseção 4.1 é apresentada a definição de processo de *software* e quais atividades são inerentes a um processo de *software*. Na subseção 4.2 é apresentada a definição de especificação de requisitos, requisitos funcionais e quais fontes de requisitos foram utilizadas na fase de levantamento de requisitos para o Web RU. Por fim, na subseção 4.3 é apontada a definição de validação de requisitos e descrita qual técnica foi utilizada para validar os requisitos funcionais do Web RU.

## 4.1 Processo de Software

"Um processo de software é um conjunto de atividades que leva à produção de um produto de *software*" (SOMMERVILLE, 2007, p. 42). Em linhas gerais, um processo de *software* é o que define quais atividades e a sequência que devem ser executadas para o desenvolvimento de um produto de *software*. Sommerville (2007, p.43) define algumas atividades fundamentais inerentes ao processo de *software* que, embora possam ser adaptadas de acordo com o processo a ser utilizado, ainda assim fazem parte do mesmo. Essas atividades são:

- Especificação de *software*: a funcionalidade do *software* e as restrições sobre sua operação devem ser definidas.
- Projeto e implementação de *software*: o *software* que atenda à especificação deve ser produzido.
- Validação de software: o *software* deve ser validado para garantir que ele faça o que o cliente deseja.
- Evolução de *software*: o *software* deve evoluir para atender às necessidades mutáveis do cliente.

A proposta de *software* apresentada neste trabalho engloba duas das atividades mencionadas. A especificação de *software*, no que tange ao

levantamento e validação dos requisitos; o projeto e implementação, com a codificação do sistema *atendendo aos requisitos do cliente*.

## 4.2 Especificação de requisitos

Segundo Sommerville (2007, p.50) elicitar requisitos “é o processo de derivação de requisitos de sistema através da observação de sistemas existentes, discussões com usuários potenciais e compradores, análise de tarefas. Isso pode envolver o desenvolvimento de um ou mais modelos de sistema e protótipos.”

Ainda segundo Sommerville (2007), a especificação de software ou engenharia de requisitos é o processo para compreender e definir quais serviços são necessários, identificando as restrições de operação e de desenvolvimento do sistema.

Pohl e Rupp (2015) definem três tipos de fontes de requisitos: *stakeholders* (interessados diretos e indiretos), documentos e sistemas em operação. Para a elicitação dos requisitos do Web RU foram utilizados como fontes os *stakeholders*, na figura do funcionário da SUTIC, em conjundo com análise do sistema utilizado atualmente.

Sommerville (2007) classifica os requisitos em funcionais, não funcionais ou requisitos de domínio. O entendimento sobre requisitos funcionais é que “são declarações de serviços que o sistema deve fornecer, como o sistema deve reagir a entradas específicas e como o sistema deve se comportar em determinadas situações. Em alguns casos, os requisitos funcionais podem também estabelecer explicitamente o que o sistema não deve fazer” (SOMMERVILLE, 2007, p. 80). No trabalho apresentado neste documento, após a obtenção das restrições e funcionalidades que o sistema proposto deveria contemplar, foram definidos os requisitos funcionais.

## 4.3 Validação de requisitos

“A validação de requisitos dedica-se a mostrar que os requisitos realmente definem o sistema que o usuário deseja. A validação de requisitos se

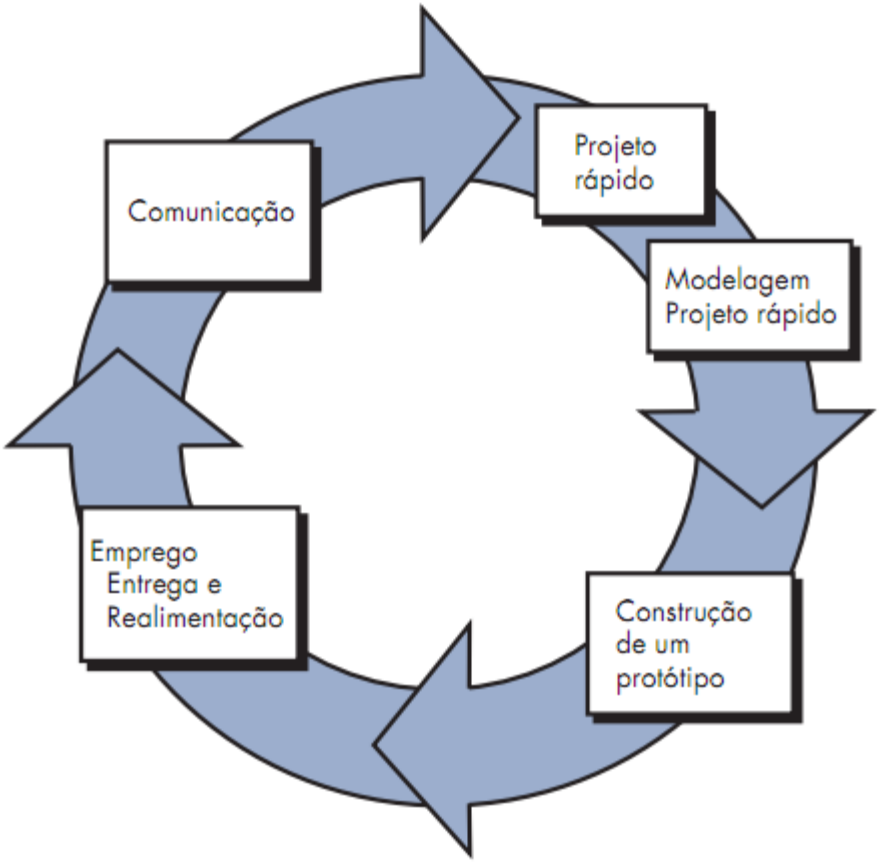
sobrepõe à análise; está relacionada à descoberta de problemas com os requisitos.” (SOMMERVILLE, 2007, p.105).

Sommerville (2007) elenca três técnicas de validação de requisitos, a saber: revisões de requisitos, que consiste da análise dos requisitos por uma equipe; geração de casos de teste, onde os requisitos são postos a testes; e a prototipação que, em linhas gerais, é o primeiro artefato do *software* produzido e é utilizado com fins de demonstração de aspectos inerentes ao sistema. A técnica escolhida para a validação dos requisitos do Web RU foi a prototipação. Essa técnica preza pela agilidade na entrega de artefatos testáveis e dispensa uma equipe de revisores técnicos de documentos de requisitos, pois o cliente se encarrega de decidir sobre o que manter ou descartar das funcionalidades apresentadas no protótipo.

### 4.3.1 Prototipação

Conceitualmente podemos definir um protótipo como sendo: “uma versão inicial de um sistema de software usado para demonstrar conceitos, experimentar opções de projeto e, geralmente, conhecer mais sobre o problema e suas possíveis soluções.” (SOMMERVILLE, 2007, p.271). Pressman (2011) sugere um paradigma de prototipação. Na **Figura 1** é ilustrado esse paradigma caracterizado por ser um processo com ciclos e composto por etapas. Nesse trabalho, cada etapa foi realizada da seguinte forma: (a) a comunicação com o cliente deu-se de forma online e com o auxílio do Google Drive, detalhado na subseção 6.1; (b) a modelagem e projeto rápido são descritos na seção 6.2; (c) o projeto rápido é esmiuçado na seção 6.3; (d) a entrega contínua e a retroalimentação (*feedback*) com participação do cliente está apresentada na seção 6.4. Detalhes adicionais do uso dessa metodologia estão presentes nas seções subsequentes desse documento.

**Figura 1 - O paradigma da prototipação**




**Fonte: Pressman (2011).**

# 5 TRABALHOS RELACIONADOS

Foram realizadas pesquisas relacionadas ao desenvolvimento de sistemas *web* para o gerenciamento de restaurantes universitários em repositórios de Trabalhos de Conclusão de Curso de Universidades e também no Google Acadêmico.

Dentre os trabalhos apreciados, destacou-se o de Batistella (2017), que apresentou o processo de implementação de um sistema *web* para controle de créditos em Restaurantes Universitários (RU). O sistema proposto tinha em seu escopo principal o controle de créditos financeiros por meio da compra desses créditos em períodos diferentes do ato da pré-refeição. Sendo assim, o aluno mantém em sua carteira virtual o saldo utilizado no processo e evita formação de filas nos horários do fornecimento das refeições pelo RU.

As tecnologias utilizadas para o desenvolvimento do sistema de Batistella (2017) foram a linguagem de programação Java conjugada com várias ferramentas do ecossistema *Spring*, incluindo o *Spring Boot* e *Spring MVC* para desenvolvimento *back-end* da aplicação, e *HTML, CSS*, *JavaScript* e *Angular* para desenvolvimento no *front-end*.

Apesar do escopo do Web RU ser diferente do que propõe em (BATISTELLA, 2017), é possível perceber similaridade quanto ao uso das tecnologias e conceitos apresentados.

Batistella (2017) não menciona a integração com outros sistemas acadêmicos. Esses outros sistemas normalmente mantém informações que podem ser utilizadas em consultas a determinadas funcionalidades ou situações cadastrais dos usuários, e essa integração poderia evitar duplicidade, divergência e inconsistências de dados em bases de dados distintas.

Apesar das similaridades, os resultados obtidos por Batistella (2017) destoam do escopo principal do trabalho apresentado nesse documento, tendo em vista que foram trabalhos que tratam da implementação de sistemas *web* de gerenciamento de créditos financeiros e controle de saldos.

# 6 MATERIAIS E MÉTODOS

Neste capítulo são apresentados os materiais e métodos utilizados para desenvolver a solução de *software* proposta neste trabalho. Na subseção 6.1 é descrita a metodologia utilizada para o levantamento de requisitos funcionais para o Web RU. Em 6.2 são apresentados os requisitos funcionais levantados e quais fontes de requisitos foram utilizadas para essa atividade. Na subseção 6.3 é explicado como se deu o desenvolvimento do protótipo do sistema desenvolvido. Em 6.4 são relatadas como as técnicas de integração contínua e implantação contínua foram utilizadas no processo de desenvolvimento até a entrega do sistema Web RU.

## 6.1 Metodologia do levantamento e validação dos requisitos

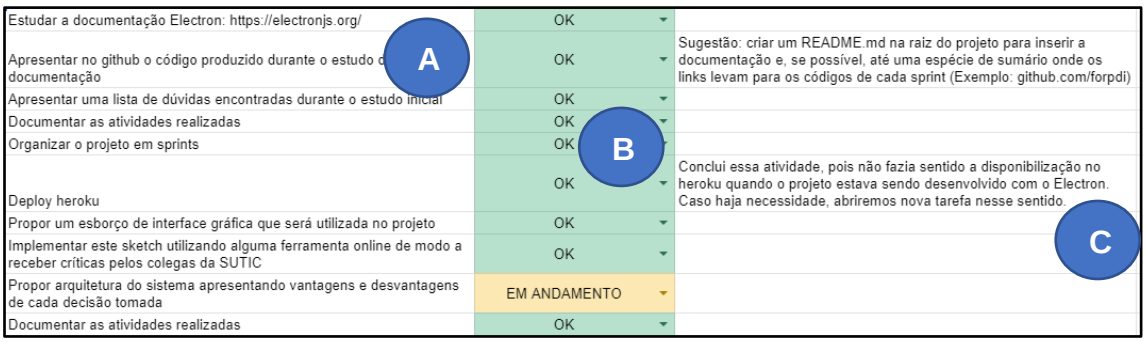
A princípio o desenvolvimento da solução de *software* apresentada neste trabalho se deu com a reunião *kick-off* com o cliente e o coordenador do projeto TCC-SIGAA[2]. Nessa reunião, foram abordados os objetivos, descrição do projeto de *software*, cronograma e tecnologias a serem adotadas.

No decorrer do trabalho, o cliente anotou as demandas a serem atendidas em uma planilha (excerto apresentado na **Figura 2**) compartilhada entre os membros da equipe: funcionários da SUTIC, representando os clientes, o coordenador do projeto e o desenvolvedor.

Na **Figura 2,** (A) está delimitando a demanda a ser cumprida; em (B) está o estado atual da demanda, se ela foi atendida plenamente (OK), se está sendo cumprida (EM ANDAMENTO) ou se ainda será vista; e em (C) eram inseridas anotações que facilitavam a comunicação entre a equipe e o cliente. Essa planilha foi utilizada no decorrer de todo o trabalho de conclusão de curso I (entre agosto de 2019 a dezembro de 2019).

[2] www.zegildo.com/tcc-sigaa/

**Figura 2 - Captura de tela com as atividades listadas pelo cliente.**



| | | |
|---|---|---|
| Estudar a documentação Electron: https://electronjs.org/ | OK | |
| Apresentar no github o código produzido durante o estudo d documentação | OK | Sugestão: criar um README.md na raiz do projeto para inserir a documentação e, se possível, até uma espécie de sumário onde os links levam para os códigos de cada sprint (Exemplo: github.com/forpdi) |
| Apresentar uma lista de dúvidas encontradas durante o estudo inicial | OK | |
| Documentar as atividades realizadas | OK | |
| Organizar o projeto em sprints | OK | |
| Deploy heroku | OK | Conclui essa atividade, pois não fazia sentido a disponibilização no heroku quando o projeto estava sendo desenvolvido com o Electron. Caso haja necessidade, abriremos nova tarefa nesse sentido. |
| Propor um esborço de interface gráfica que será utilizada no projeto | OK | |
| Implementar este sketch utilizando alguma ferramenta online de modo a receber críticas pelos colegas da SUTIC | OK | |
| Propor arquitetura do sistema apresentando vantagens e desvantagens de cada decisão tomada | EM ANDAMENTO | |
| Documentar as atividades realizadas | OK | |



**Fonte: SUTIC (2018).**

O desenvolvimento das atividades propostas era apresentado ao cliente a cada 15 (quinze) dias, onde as funcionalidades desenvolvidas eram avaliadas quanto à completude. Em seguida, novas atividades eram propostas e descritas na planilha.

O primeiro ciclo de atividades proposto consistiu do estudo da tecnologia proposta pelo cliente para desenvolver o *software*. Detalhes sobre a escolha das tecnologias são descritas no Capítulo 7.

Foi sugerido o estudo do *Framework Electron*[3], em seguida, o código produzido deveria ser apresentado e documentado em um repositório no *GitHub*[4]. Posteriormente ao primeiro ciclo, foi sugerido recomendar um esboço de interface gráfica que seria utilizada no projeto. Além disso, foi implementado um protótipo do projeto que deveria ser posto *online* para receber críticas e sugestões da SUTIC. Nessa etapa do projeto, foi possível elicitar os requisitos do sistema por meio da observação do sistema utilizado atualmente e discussão das restrições atuais.

Na terceira etapa, foram realizadas correções no protótipo, sugeridas pela SUTIC. Ademais foi proposto pela SUTIC a implementação de um *CRUD* (*Create, Read, Update, Delete*) utilizando o *Electron*. No entanto o *Electron* foi substituído por outra tecnologia, pois apresentou alguns problemas que não são desejáveis ao desenvolvimento do *software* proposto. Os problemas foram relativos à lentidão no *software* desenvolvido com a tecnologia e a pouca produtividade na escrita do código fonte da aplicação.

[3] www.electronjs.org/
[4] https://github.com/vitorcardoso98/WebRU-UFERSA

A quarta e quinta etapa tiveram como atividades sugeridas a implementação de um *CRUD* com a nova tecnologia sugerida: a linguagem de programação Java e o *Framework MVC Vraptor*.

Além da implementação do *CRUD*, foi proposto pelo cliente a criação de um Modelo Entidade Relacionamento (MOR) com base nos requisitos do sistema e a realização da implantação do protótipo (*deployment)* da aplicação no *Heroku*.

A partir dessa etapa, o fluxo de atividades consistiu da implementação das funcionalidades do sistema, visto que as fases de elicitação de requisitos e projeto já haviam sido completadas e o desenvolvedor já estava proficiente no uso da tecnologia proposta.

## 6.2 Especificação e validação de requisitos para o Web RU

Espera-se que um software que teve as demandas levantadas em conjunto com o cliente atenda às exigências previstas, solucionando o problema de forma satisfatória. Dessa forma, desenvolver um *software* requer um projeto que minimize ao máximo os riscos e custos de desenvolvimento e que seja implementada uma solução de qualidade que atenda aos requisitos dos potenciais usuários.

No contexto do desenvolvimento do sistema Web RU, o papel do cliente foi desempenhado pela SUTIC. Atualmente a SUTIC possui a demanda de atualizar o *software* utilizado no gerenciamento do RU, mantendo as regras de negócios atuais. Os requisitos para o Web RU foram levantados por meio da observação do sistema existente e discussões acerca dos requisitos existentes no sistema atual. Sendo assim, foi possível listar os requisitos que devem ser contemplados no sistema em desenvolvimento e sugerir requisitos que suprem novas demandas da UFERSA. Os requisitos funcionais para o Web RU são apresentados no **Quadro 1**.

**Quadro 1 - Requisitos funcionais do Web RU**

| Código | Descrição |
|---|---|

| | |
|---|---|
| RF01 | Cadastro, inativação e exclusão do aluno bolsista PROAE. |
| RF02 | Cadastro, inativação e exclusão do aluno UFERSA. |
| RF03 | Cadastro, inativação e exclusão dos gestores do sistema. |
| RF04 | Controle de fluxo de refeição. |
| RF05 | Cadastro, visualização, exclusão e alteração de parâmetros do sistema. |
| RF06 | Geração do relatório de histórico de consumo. |
| RF07 | Geração do relatório de bolsistas cadastrados. |
| RF08 | Cadastro e visualização de avaliações. |

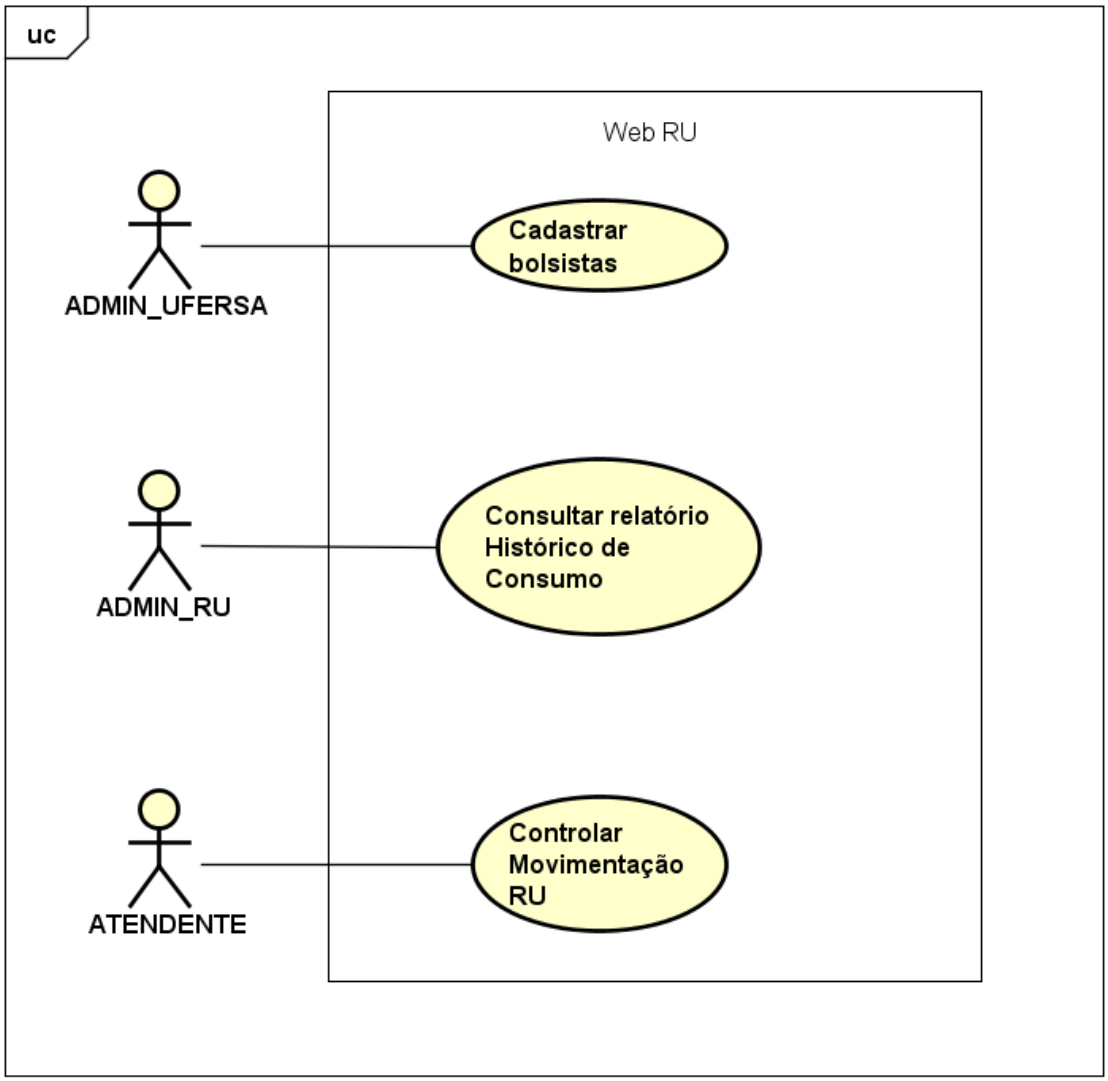
**Fonte: Autoria própria (2019).**

Os requisitos **RF01** e **RF02** são referentes às operações de gerenciamento dos alunos bolsistas PROAE e UFERSA, respectivamente. Os alunos bolsistas são aqueles cujo valor das refeições (almoço e jantar) são custeados/subsidiados integralmente pela UFERSA e são assistidos pela unidade PROAE. Já os alunos UFERSA (**RF02**) têm apenas uma parte da sua refeição custeada, sendo a outra parte paga pelo aluno.

A funcionalidade de gerenciamento de gestores, requisito **RF03**, é responsável por manter os dados dos gestores no sistema. Os gestores podem assumir três papéis diferentes no contexto do Web RU, cada qual com acesso à recursos específicos do sistema. Por meio do diagrama de casos de uso ilustrado na **Figura 3**, é possível perceber que diferentes usuários são responsáveis por desempenhar funções distintas no sistema.

O usuário com o perfil de **ADMIN_UFERSA** é um funcionário lotado em alguma unidade na UFERSA que tem relação com o RU. Como exemplo, um funcionário da PROAE pode cadastrar alunos (**RF01 e RF02**), mas os usuários **ADMIN_RU** e **ATENDENTE** não devem possuir esse nível de acesso. Da mesma forma que um **ATENDENTE** não deve ter acesso ao histórico de consumo, por exemplo, ficando apenas permitindo-lhe cadastrar uma movimentação de uma refeição.

**Figura 3 - Níveis de acesso ao sistema**




**Fonte: Autoria própria (2019).**

O controle de movimentação (**RF04**) é uma das funcionalidades mais importantes do sistema. Essa funcionalidade registra o fluxo de refeições vendidas no RU. Dessa forma, no momento da refeição, ao aluno apresentar sua identificação ao atendente, por meio de uma carteira com um código de barras, o sistema confere se o aluno é bolsista ou não e registra os dados da operação, como data/hora e valor pago pela UFERSA. Dessa forma, essa operação é lançada no histórico de consumo (**RF06**) que, por sua vez serve para fins de contagem do número de refeições vendidas e o valor em dinheiro a ser pago pela UFERSA à empresa que presta seus serviços ao RU.

Atualmente, a opção de geração de relatórios com os registros dos valores das refeições não está contemplada no sistema atual, sendo assim, foi solicitado por parte da SUTIC que essa funcionalidade fosse implementada.

O requisito **RF05** diz respeito a funcionalidade de gerenciamento (cadastro, exclusão, visualização e alteração) de parâmetros utilizados pelo sistema, como exemplo, na tela de cadastro de movimentação, os horários de início das refeições e os valores padrões das mesmas. Esses parâmetros são recuperados para fins de cálculo e controle pelo sistema e, para fins de exibição ao atendente.

O requisito **RF07** trata da geração de um relatório com os nomes e matrículas dos bolsistas ativos e inativos presentes no sistema. Já o requisito **RF08** diz respeito ao gerenciamento das avaliações que devem ser feitas pelos usuários do RU para que haja melhoria do serviço prestado.

## 6.3 Desenvolvimento do protótipo da aplicação do Web RU

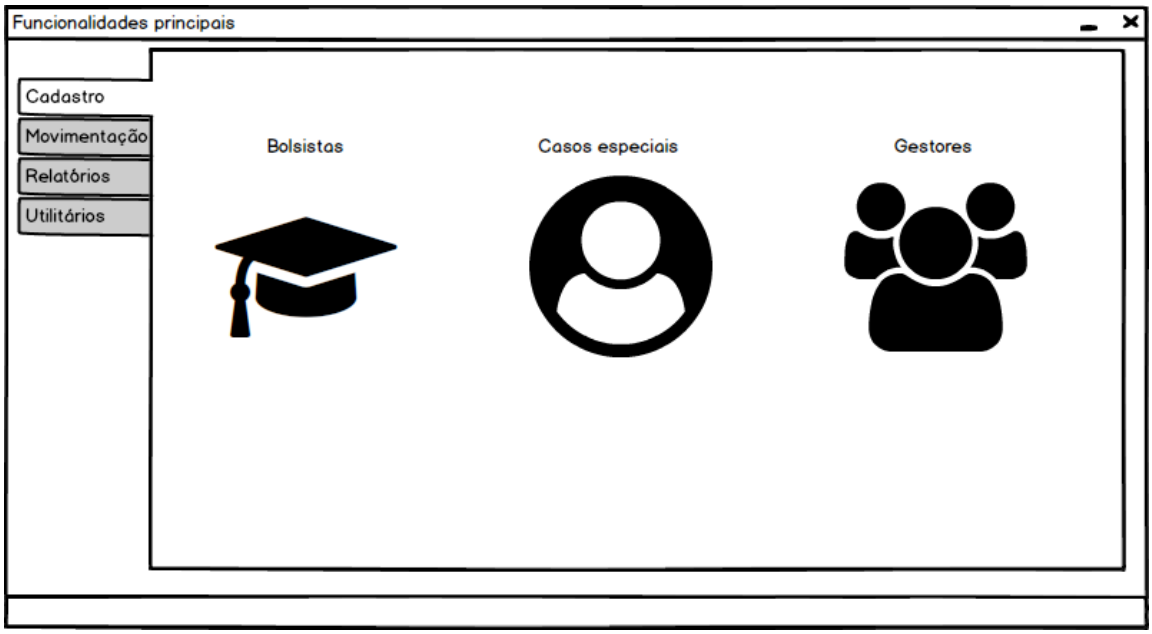
De posse dos requisitos funcionais descritos anteriormente, foi possível pensar um projeto de *software* que atendesse as necessidades e restrições que o sistema deveria contemplar para suprir as demandas do cliente. Dessa forma, foi possível desenvolver algumas telas com o objetivo de construir um protótipo de *software* para validar os requisitos levantados e apresentar um projeto de interface com o usuário.

Inicialmente, o servidor da SUTIC apresentou capturas de telas explicando o fluxo das interações do sistema atual, com o intuito de balizar as discussões e possíveis alterações no fluxo atual por meio de melhorias ou o desenvolvimento de novas funcionalidades.

As etapas referentes a projeto rápido, modelagem do projeto rápido e construção do protótipo foram implementadas com as ferramentas *Balsamiq Wireframes*, uma ferramenta rápida de *wireframes* de baixa fidelidade que reproduz a experiência de desenhar em um bloco de notas ou quadro branco, mas usando um computador; e *Invision App*, um *software web* que permite criar protótipos interativos com base em imagens estáticas. Na **Figura 4** está

exemplificado o desenvolvimento de uma tela do Web RU utilizando a ferramenta *Balsamiq Wireframes*.

**Figura 4 - Design da tela inicial desenvolvida com o Balsamiq**




**Fonte: Autoria própria (2019).**

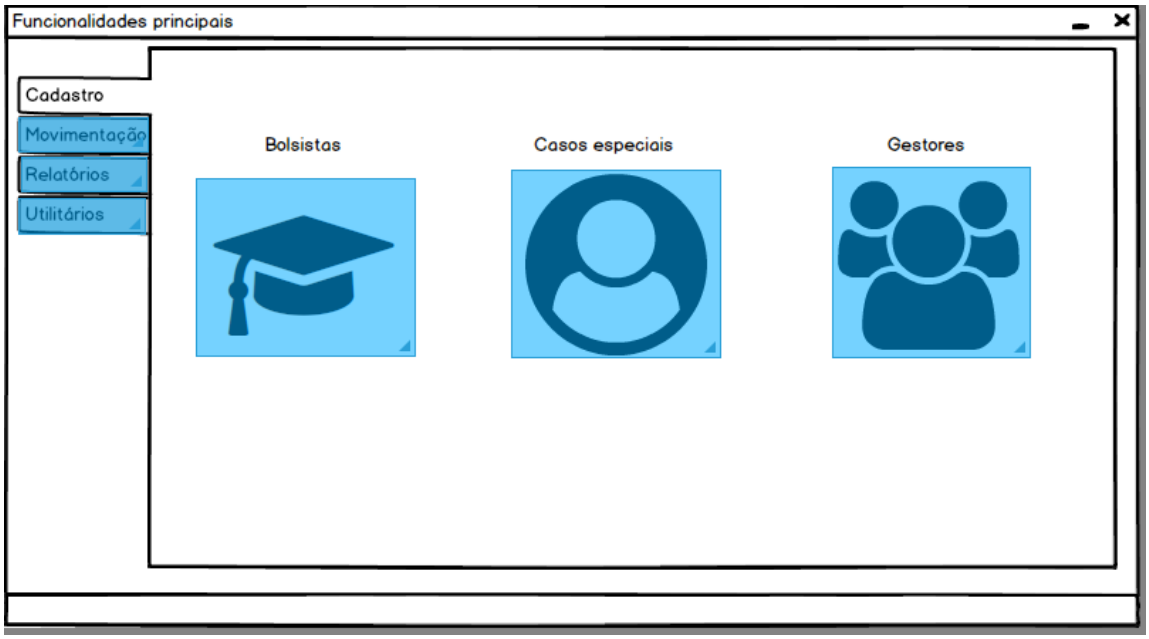
Segundo Pressman (2011), esse tipo de protótipo é considerado descartável, pois se trata apenas de uma forma de identificar os requisitos do sistema e, neste caso, utilizado como mecanismo de validação.

> "Uma iteração de prototipação é planejada rapidamente e ocorre a modelagem (na forma de um "projeto rápido"). Um projeto rápido se concentra em uma representação daqueles aspectos do software que serão visíveis aos usuários finais (por exemplo, o layout da interface com o usuário ou os formatos de exibição na tela)." (PRESSMAN, 2011, p.63).

Apesar de se tratar de um protótipo de baixa fidelidade e descartável, com o uso de técnicas e abordagens apropriadas é possível desenvolver um protótipo confiável e que atenda o objetivo de validar os requisitos. Sommerville (2007) aponta duas dessas técnicas que podem enriquecer o projeto de prototipação. "Você pode usar uma técnica de *storyboarding* para apresentar o projeto de interface. Um *storyboard* é uma série de esboços que ilustram uma sequência de interações." (SOMMERVILLE, 2007, p.253).

Na **Figura 5** e **Figura 6** são exemplificadas as interações que podem ser desenvolvidas utilizando o *Invision App*. As marcações em azul da **Figura 5** são os fluxos que o usuário do sistema pode optar. A **Figura 6** é o resultado do fluxo escolhido pelo usuário. Nesse caso, optou-se por clicar na opção Movimentação, do menu vertical localizado a esquerda da tela e como resultado se tem a exibição da tela referente à opção escolhida.

**Figura 5 - Exemplo 1 de interação alcançada com o Invision App**

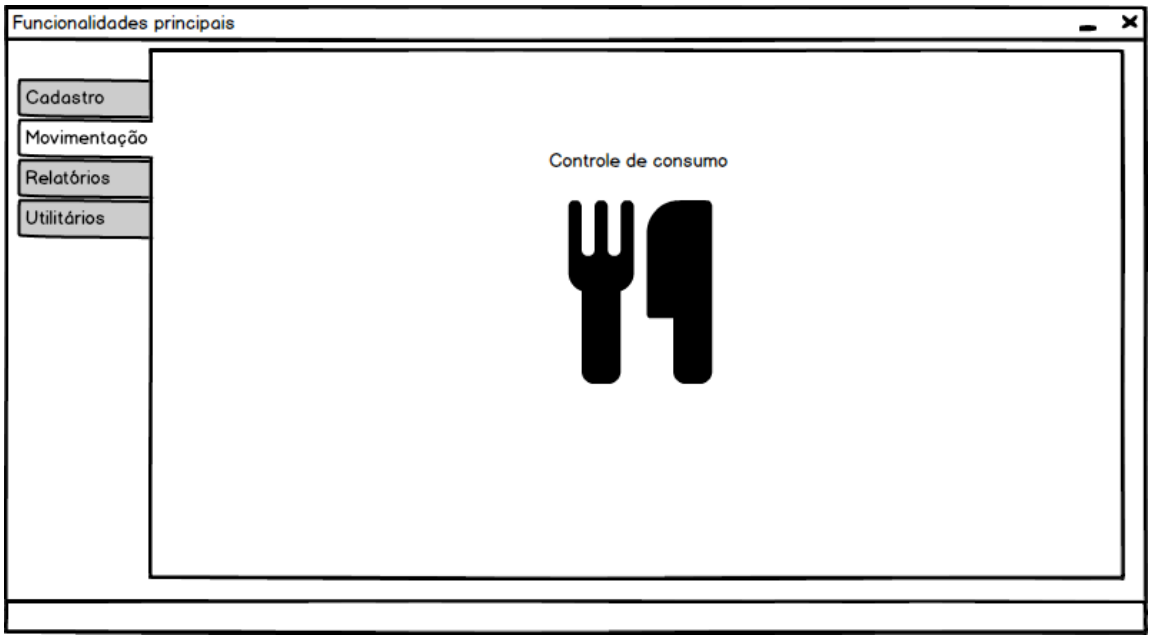



**Fonte: Autoria própria (2019).**

Essa abordagem utilizada na modelagem e construção do protótipo é definida por Sommerville como ‘Mágico de Oz’. Nessa abordagem, os usuários interagem com o que parece ser um sistema de computador, mas suas entradas são na verdade direcionadas a uma pessoa oculta que simula as respostas do sistema. (SOMMERVILLE, 2007).

Ainda de acordo com o paradigma de Pressman, a etapa de emprego, entrega e retroalimentação fecham o ciclo incremental do projeto de prototipação. O *Invision App* possui funcionalidades que permitem que colaboradores, sejam usuários do sistema, aprovem ou não as telas, esse retorno (*feedback)* retroalimenta o ciclo de interações entre o cliente e o desenvolvedor do sistema.

**Figura 6 - Exemplo 2 de interação alcançada com o Invision App**

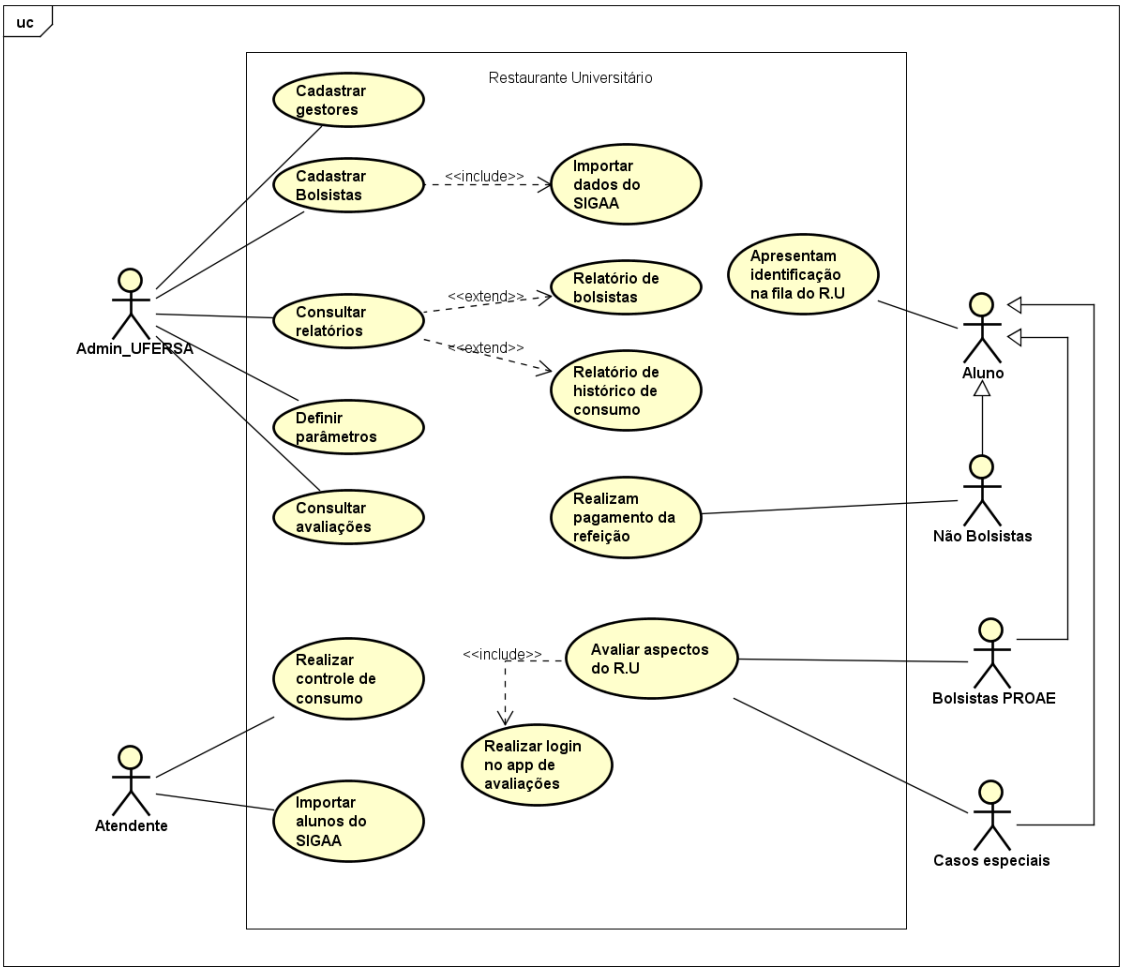



**Fonte: Autoria própria (2019).**

Na **Figura 7** apresenta-se o diagrama de caso de uso proposto para a nova versão do sistema de RU. O objetivo principal foi o de conceber um arcabouço de sistema de *software* limpo, que exigisse do responsável pelo controle e fluxo de usuários do RU o mínimo de cliques ou digitação de dados de modo a simplificar suas atividades. Dessa forma, minimiza-se as chances de erros humanos, tornando mais célere todas as etapas de utilização do sistema.

Ao cadastrar os usuários do RU, no sistema atual, o administrador pode equivocar-se e digitar matrículas inexistentes ou pertencentes a usuários diferentes. Essa necessidade de povoar manualmente o sistema do RU gera uma série de conflitos oriundos de inconsistências ou faltas que podem ser sanadas obtendo-se as informações já presentes no Sistema Integrado de Gestão de Atividades Acadêmicas (SIGAA) (OLIVEIRA; FERREIRA, 2015; LOPES, 2018).

**Figura 7 - Diagrama de casos de uso do Web RU**




**Fonte: Autoria própria (2019).**

Nessa perspectiva, para automatizar o cadastro inicial e evitar erros de inserção de dados foi proposto também uma integração do novo sistema do RU com o SIGAA de modo a importar as informações já consolidadas. Dessa forma, o sistema proposto tem a capacidade de importar as informações do SIGAA para povoar o cadastro, eliminando a necessidade de cadastro manual.

## 6.4 Integração contínua e implantação contínua

Segundo Barbosa (2018, p. 14), consoante com Fowler (2006), “os conceitos de Integração contínua (*CI - Continuous Integration*) e implantação contínua (*CD – Continuous Deployment*) visam deixar automáticos algumas fases do processo de desenvolvimento de *software* do início até a entrega do

*software* em produção, inclusive durante a manutenção de *software*, garantindo versatilidade de mudanças e entregas rápidas de novas funcionalidades".

De acordo com Pessôa (2017, p. 20), conforme citado por Rahman et al. (2015) "há 11 práticas que podem ser identificadas na entrega contínua de *software"*. Nessas práticas é possível destacar o uso do repositório de código fonte como alternativa ao versionamento para produzir e manter códigos referentes às funcionalidades do sistema. Aliado ao versionamento está a possibilidade de se utilizar a implantação automatizada da aplicação a partir de ramificações (*branches*) do repositório.

Na fase de especificação e análise dos requisitos do Web RU, foram identificados alguns problemas que limitaram a evolução do *software* utilizado atualmente no RU da UFERSA. Dentre esses problemas, encontram-se o versionamento do código fonte da aplicação e a consequente distribuição e a disponibilização do *software* após uma atualização.

No escopo da proposta de *software* concebida neste trabalho, há a utilização de ambas as práticas citadas anteriormente, com o objetivo de facilitar e deixar automáticas algumas partes do processo de desenvolvimento de *software* atual do RU. Para isso foi utilizada a ferramenta de versionamento *Git*[5], com a utilização do *GitHub* para disponibilização em nuvem do código fonte da aplicação. Para implantação, utilizou-se o *Heroku*[6] – um serviço *web*, baseado em nuvem, que permite implantar e gerenciar aplicações.

Durante a implementação do sistema Web RU, sentiu-se a necessidade de utilizar-se das práticas de integração contínua e implantação contínua com o objetivo de prover a organização do código implementado e a disponibilização da aplicação em um ambiente *web*, mimetizando o ambiente que a aplicação poderia ser disponibilizada na etapa de uso pelos frequentadores do RU. Para isso, foi estabelecido um fluxo de trabalho com ferramentas de versionamento e implantação de *software*.

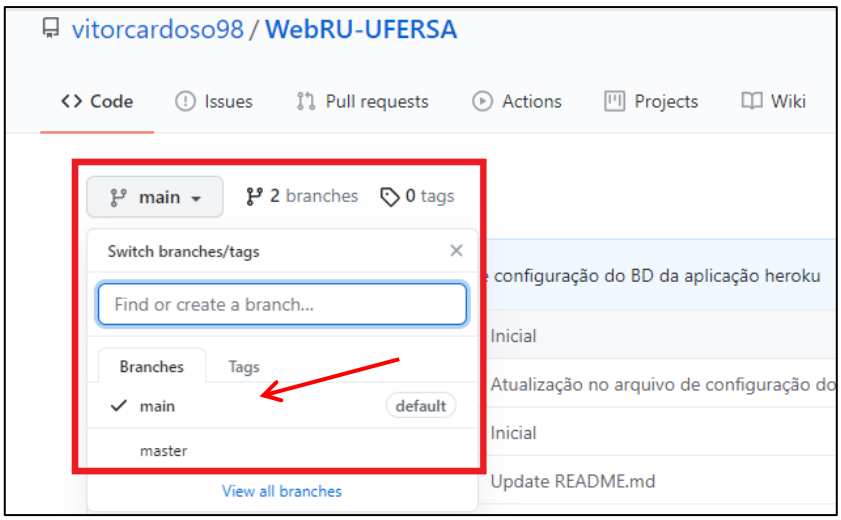
Atualmente é possível aproveitar modelos de fluxos de trabalho como o *GitFlow*[7] ou *GitHub Flow*[8]. Esses modelos de fluxos de trabalhos facilitam o

[5] https://git-scm.com/
[6] https://devcenter.heroku.com/categories/reference
[7] https://www.atlassian.com/br/git/tutorials/comparing-workflows/gitflow-workflow
[8] https://guides.github.com/introduction/flow/

gerenciamento de ramificações (*branches*) do projeto. A ideia de dispor de múltiplas ramificações em um repositório está na necessidade de múltiplos desenvolvedores trabalharem em conjunto ou da divisão das funcionalidades da aplicação, sendo possível, por exemplo, manter uma ramificação (*branch*) com o código da aplicação sempre estável e outras ramificações (*branches*) com códigos ainda não testados ou em desenvolvimento.

Dada a complexidade e robustez do *GitFlow*, optou-se por recorrer ao modelo de gerenciamento de ramificações (*branches*) baseado no *GitHub Flow*, por se tratar de um fluxo mais enxuto e mais adequado ao tamanho do projeto Web RU, aliado ao fato de não haver, nesse trabalho, a atuação concorrente de múltiplos desenvolvedores. Assim, o repositório do *GitHub* do Web RU consiste de duas ramificações, denominadas de *main* e *master*. Na *main* são mantidos os códigos de funcionalidades que ainda estão em desenvolvimento e na *master*, há consolidação da ramificação *main*, com a versão mais estável possível do código. Na **Figura 8** são exibidas as ramificações (*branches*) do projeto.

**Figura 8 - Ramificações (*branches*) do repositório do Web RU no GitHub**




**Fonte: Autoria Própria (2021).**

Para implantar o projeto de forma online, foi utilizado o *Heroku*. O processo de implantação consiste da criação do contêiner da aplicação no *Heroku*, configuração do banco de dados online e a implantação (*deployment*).

Segundo Barbosa (2018, p. 25) conforme citado por Azeri (2017), a implantação contínua (*CD*) pode ser feita de forma automatizada quando há uma integração contínua bem estruturada. Da maneira que o fluxo de trabalho do Web RU foi estruturado é possível alcançar essa automatização da disponibilização das versões do projeto. Sendo assim, no contêiner da aplicação no *Heroku* foi configurado uma funcionalidade de implantar o código a partir de uma ramificação do repositório no *GitHub*.

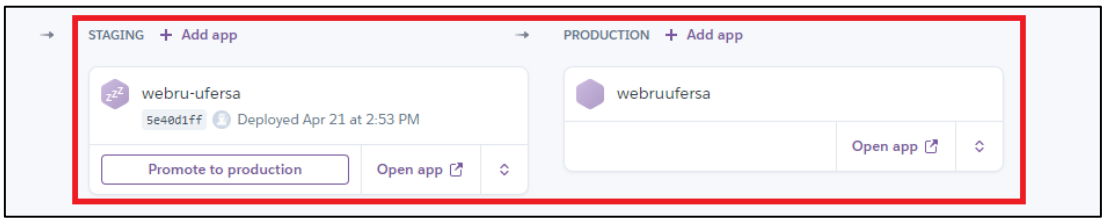
Com isso, de forma integrada e automática, é possível disponibilizar uma nova versão da aplicação sempre que há uma nova versão do código-fonte da aplicação na ramificação *main* do repositório do Web RU no *GitHub*. Após cada modificação no *GitHub*, a plataforma do Heroku dispara um evento automático de compilação, execução de testes automáticos (se disponíveis), empacotamento de aplicação e disponibilização da última versão da aplicação em uma *URL* pré-definida para o projeto.

Idealmente, necessita-se que a aplicação que está implantada esteja sempre estável e disponível para uso pelo usuário final. Porém, nem sempre é possível entregar uma funcionalidade completa que não dependa de outra. Nessa situação, pode ocorrer que uma funcionalidade implantada ocasione falhas na versão da aplicação que já está implantada. Surge, então, a necessidade de se criar diferentes ambientes para sempre manter a aplicação o mais estável possível e que não interfira no desenvolvimento de outras funcionalidades.

Para o Web RU, no *Heroku*, adotou-se a criação de estágios de trabalho (*pipelines*[9]). O fluxo de entrega contínua sugerida pelo *Heroku* propõe quatro estágios de trabalho, a saber: *Development, Review, Staging* e *Production*. Novamente, dado o escopo menor do Web RU e o desenvolvimento do código por apenas uma pessoa, foram utilizados dois dos quatro estágios, como mostrado a **Figura 9**.

[9] https://devcenter.heroku.com/articles/pipelines

**Figura 9 - Estágios de implantação da aplicação Web RU no Heroku**




**Fonte: Captura de tela Heroku (2021).**

No estágio de encenação (*staging)* é implantada a aplicação automaticamente através da integração com a ramificação *main* do *GitHub*. A aplicação é testada manualmente pelo desenvolvedor a fim de identificar possíveis erros na aplicação, e caso não apresente erros, a aplicação é promovida para o estágio de produção (*production)*, sinalizando que essa é a versão mais estável e será utilizada pelos usuários finais.

Caso a aplicação apresente falhas, o código implementado passa pela correção e o processo de implantação é novamente iniciado. A utilização de uma ferramenta de integração contínua nesse processo garante a disponibilização de versões funcionais do *software* em espaços de tempo mais curtos.

# 7 APLICAÇÃO DESENVOLVIDA

A UFERSA utiliza atualmente um sistema *desktop* para gerenciar a venda de refeições do RU. Porém, ao longo do tempo esse sistema apresentou limitações que inviabilizaram sua atualização. Dentre as limitações identificadas, destacaram-se: a impossibilidade de geração de relatórios condizentes com as demandas atuais do RU; a atualização da base de dados dos discentes, que implica em trabalho manual excessivo para atualizar o registro de alunos ingressantes na UFERSA a cada semestre letivo; dificuldade de disponibilização e distribuição do sistema para outros campi, por se tratar de um sistema *desktop* há dificuldade no empacotamento do *software* a cada nova implantação.

Diante das limitações apresentadas, optou-se nesse trabalho por implementar um sistema *web* que mantivesse as regras de negócio atuais e a adição de novas funcionalidades e recursos que viabilizasse a correção das limitações identificadas. O Web RU foi desenvolvido utilizando uma arquitetura dividida em *back-end* e *front-end*, consistindo num sistema de múltiplas camadas.

Como base para o *back-end*, foram utilizadas a linguagem de programação Java e o ecossistema de tecnologias *Spring* (*Spring Boot, Spring Data JPA, Spring Security*), com SGBD PostgreSQL. Inicialmente, fora adotado o *Framework Vraptor* para o *back-end*, no entanto o repositório fonte do projeto no *GitHub* passou a não receber mais atualizações desde 5 de maio de 2015, o que pode denotar a maturidade do projeto ou o abandono por parte dos mantenedores. Achou-se que esse fator não seria desejável para a disponibilização de uma aplicação em ambiente de produção, portanto, o *Framework Vraptor* foi abandonado.

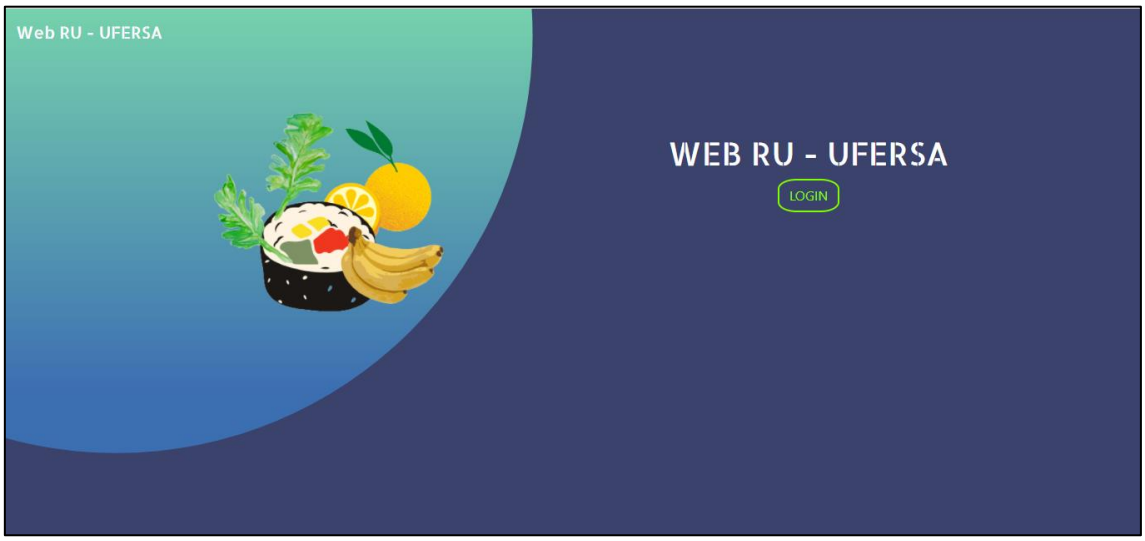
No *front-end* da aplicação utilizou-se as tecnologias para o lado do cliente (*client-side*) *HTML, CSS e JavaScript,* ambas empacotadas por meio do framework *Bootstrap*. O *Bootstrap*[10] permite construir páginas *web* utilizando componentes e *plugins* prontos, provendo responsividade (adaptabilidade da aplicação a diferentes tamanhos de telas) às páginas construídas, sem

[10] https://getbootstrap.com.br/docs/4.1/getting-started/introduction/

demandar muito esforço na escrita do código fonte das funcionalidades de interação com o usuário.

Na **Figura 10** é apresentada a página inicial do Web RU, onde está presente o botão de chamada para a página de *login*.

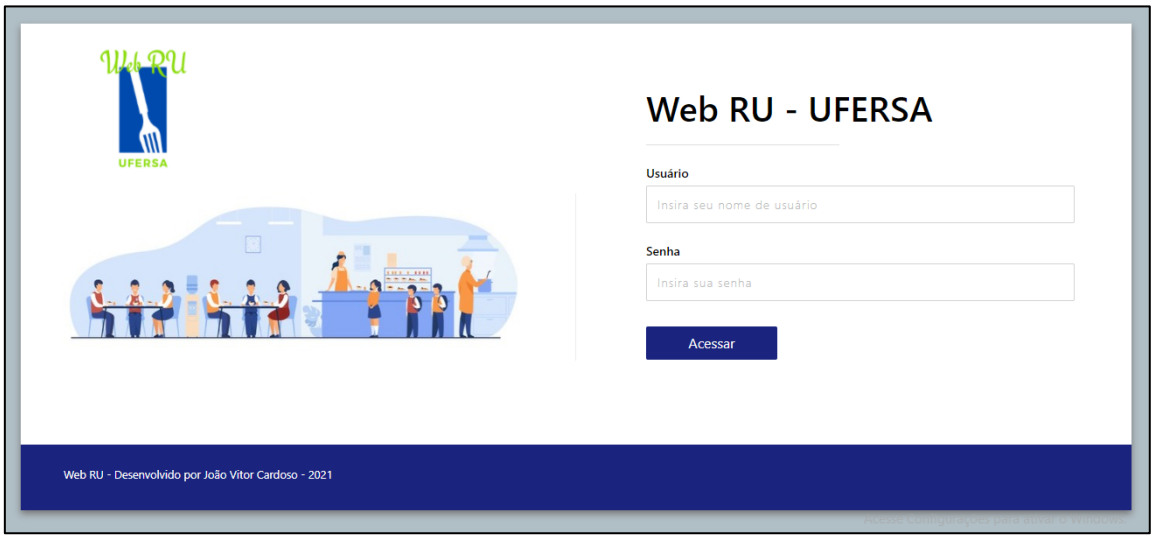
**Figura 10 - Página Inicial do Web RU**




**Fonte: Autoria própria (2021).**

Ao clicar no botão de *login* o usuário é direcionado para a página apresentada na **Figura 11**, onde são apresentados os campos para inserção das credenciais do usuário.

**Figura 11 - Página de *login* do Web RU**




**Fonte: Autoria própria (2021).**

Após inserir as informações e clicar no botão *Acessar*, o sistema verifica as credenciais informadas. Caso as credenciais estejam válidas, o usuário é redirecionado para página de acesso às funcionalidades do sistema, conforme mostrado na **Figura 12**. Ressalta-se que, para o usuário finalizar a sessão atual, basta clicar no menu do sistema (link *Sair*), presente no canto inferior esquerdo do menu da aplicação.

**Figura 12 - Página de acesso às funcionalidades do Web RU**


**Fonte: Autoria própria (2021).**

A cada semestre, novos alunos ingressam na UFERSA e requerem o uso do cadastro no RU. Atualmente, o fluxo de cadastro de alunos, por parte do sistema utilizado, exige que os dados desses alunos sejam cadastrados manualmente na base de dados. No Web RU foi implementado uma integração com uma *API* que trafega dados no formato *JSON* (*JavaScript Object Notation*).

O intuito dessa funcionalidade parte do princípio que os dados dos alunos estão consolidados no SIGAA. No entanto, como o SIGAA da UFERSA não disponibiliza recursos via API externa (utilizando arquitetura REST, por exemplo), para que a aplicação pudesse ser testada antes que a SUTIC fizesse a disponibilização do ponto de conexão (*endpoint)* que provê o *JSON* com os dados do cadastro de alunos, foi utilizada uma *API* temporária, criada utilizando o *JSONPlaceholder,* uma aplicação *web* que disponibiliza API para prototipagem rápida de sistemas.

A base de dados dessa API é um arquivo *JSON* criado com os dados que refletem os atributos das classes da camada de modelo da aplicação em desenvolvimento, e hospedado em um repositório no *GitHub*. O

*JSONPlaceholder* fornece um padrão de rotas para acesso à esses dados. Na **Figura 13** é apresentada uma captura de tela com os dados da base de dados da *API*. Na aplicação do Web RU, é feita uma chamada a essa *API* usando a URI my-json-server.typicode.com/vitorcardoso98/fakeapi-ufersaru/aluno. Os dados de nomes utilizados para popular o arquivo *JSON* são fictícios e foram obtidos por meio do 4Devs[11] - um gerador de nomes aleatórios disponível na Internet.

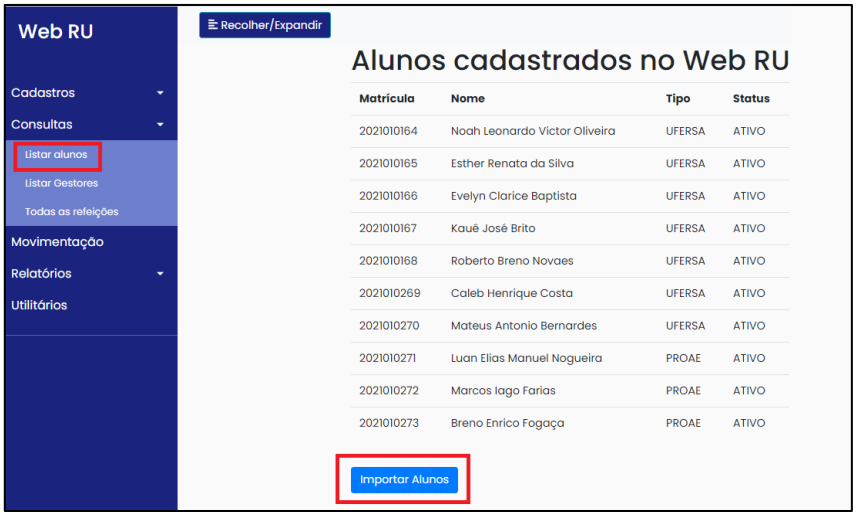
**Figura 13 - Arquivo JSON da API temporária com os dados fictícios**


**Fonte: Captura de tela do GitHub (2021).**

No Web RU essa funcionalidade de importar os alunos de uma API externa atende o requisito RF01 e RF02, no que tange ao cadastro dos alunos UFERSA e bolsistas PROAE. Essa funcionalidade é executada pelo usuário com perfil de administrador UFERSA (ADMIN_UFERSA), acessando a listagem de alunos e clicando no botão *Importar Alunos*, conforme mostrado na **Figura 14**. Os alunos importados são persistidos na banco de dados da aplicação.

[11] https://www.4devs.com.br/gerador_de_pessoas

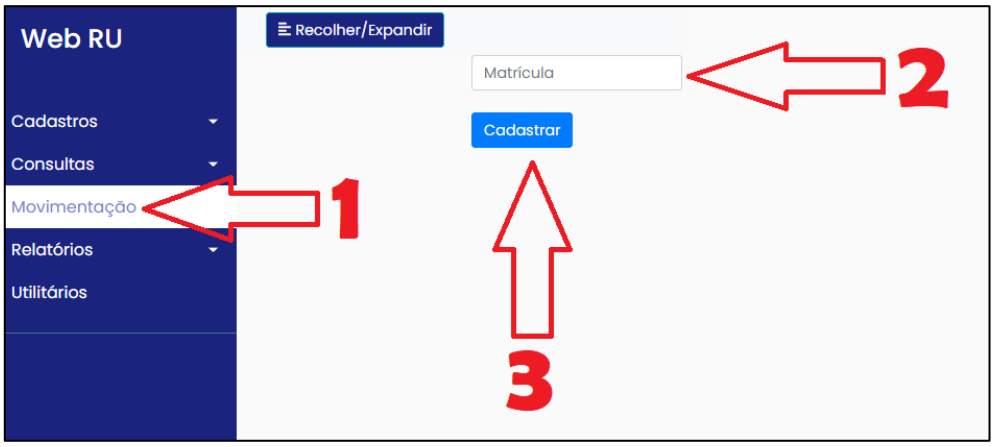
**Figura 14 - Tela com alunos cadastrados no Web RU**




**Fonte: Autoria própria (2021).**

Após o cadastro dos alunos ser finalizado, é possível continuar o fluxo operacional do RU. Uma das principais funcionalidades refere-se às vendas de refeições, que no sistema são tratadas como movimentações. Na **Figura 15** é apresentado esse fluxo, detalhado a seguir**.**

**Figura 15 - Tela com a funcionalidade de Movimentação**




**Fonte: Autoria própria (2021).**

- Passo **1**: ao clicar no menu do sistema (link *Movimentações*), é aberta a tela de venda de refeições.

- Passo **2**: o usuário com perfil de *ATENDENTE* faz a leitura do código de barras presente na carteira de identificação do aluno. O dado lido alimenta o campo de matrícula presente na tela (o campo de formulário *Matrícula*).

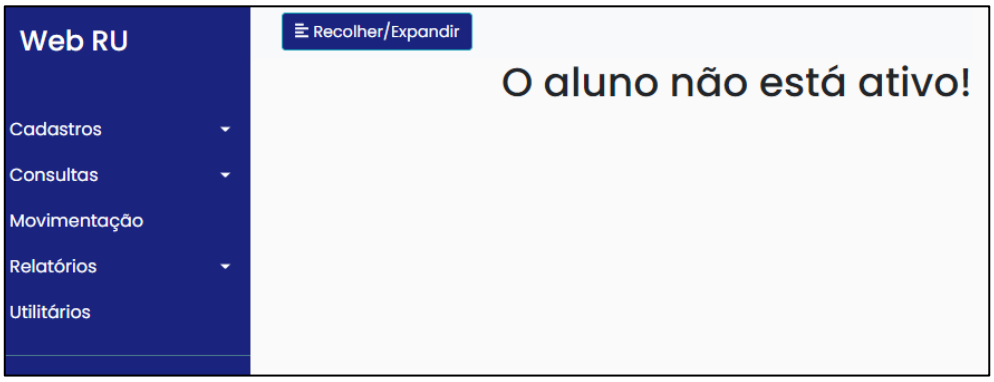
- Passo **3**: ao clicar no botão, o sistema verifica se o aluno está com *status* ATIVO e é exibida a tela com detalhes da movimentação (**Figura 16**).

**Figura 16 - Informações de uma refeição vendida**


**Fonte: Autoria própria (2021).**

O fluxo alternativo resultante da verificação do sistema ao *status* do aluno é apresentado na **Figura 17** quando identificado que o cadastro encontra-se INATIVO. A inativação do cadastro do aluno parte do princípio que o mesmo não está frequentando as aulas com assiduidade e dessa forma a API disponibilizaria a situação do aluno como inativo, para que o Web RU não permita a venda de uma refeição nesse caso.

**Figura 17 - Tela informando que o aluno está inativo no sistema**

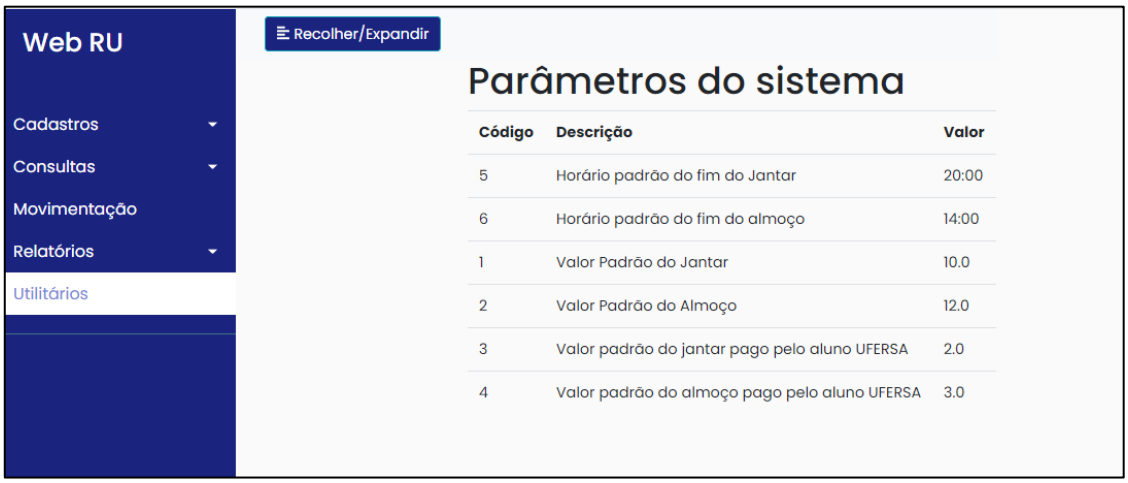



**Fonte: Autoria própria (2021).**

A funcionalidade de controle de movimentação utiliza alguns parâmetros para registrar vendas de refeições. Esses parâmetros são referentes aos **horários** de fim do almoço/jantar e **valores** das refeições.

A realização do cadastro dos parâmetros utilizando a aplicação remove a complexidade de inserções e alterações via manipualação direta do banco de dados. A parametrização é utilizada para fins de cálculos dos valores do quantitativo das refeições vendidas e para a automatização da definição dos horários do almoço e jantar. Na **Figura 18** são apresentados os parâmetros necessários para a operação de venda de refeições.

**Figura 18 - Tela com os parâmetros cadastrados no sistema**




**Fonte: Autoria própria (2021).**

A SUTIC precisa que o sistema gere relatórios com o quantitativo das refeições e os valores a serem pagos à empresa contratada para fornecer o serviço. Porém, o sistema atual não contempla a opção de gerar o relatório de histórico de consumo com os valores totais das refeições, gerando um relatório limitado ao quantitativo de refeições vendidas.

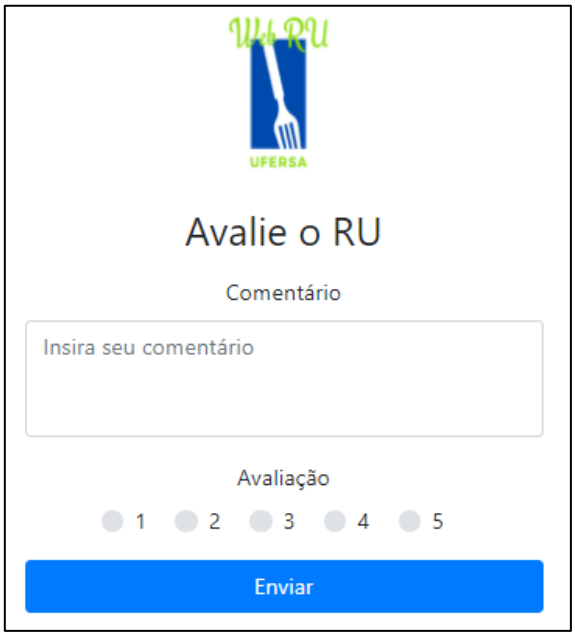
Dessa forma, no Web RU foi implementada a melhoria nesse relatório (**Figura 19**), gerando o histórico de consumo com os valores totais a serem pagos, discriminados de acordo o tipo de refeição (almoço ou jantar) e o tipo de subsídio (integral ou parcial).

**Figura 19 - Relatório histórico de consumo**


**Fonte: Autoria própria (2021).**

Outra funcionalidade levantada pela SUTIC para atender demandas atuais no tocante a administração do espaço, refere-se à possibilidade dos alunos avaliarem o serviço fornecido pelo RU. Dessa forma, foi implementada uma funcionalidade, onde os alunos podem fornecer respostas aos administradores do RU, sobre o serviço que está sendo utilizado. Na **Figura 20** apresenta-se a tela com o formulário de avaliação que deve ser preenchido pelo aluno ao submeter uma avaliação. O formulário de avaliação ainda carece de validação por parte da SUTIC e ainda será integrado ao menu principal do sistema.

**Figura 20 - Formulário de avaliação**




**Fonte: Autoria própria (2021).**

# 8 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

Nesta seção são apresentados o resumo dos resultados obtidos e propostas de trabalhos futuros.

Na subseção 8.1 são apresentados os resultados obtidos com a pesquisa desenvolvida neste trabalho, são descritos os resultados encontrados a partir dos objetivos e apresentada a relevância do trabalho para o tema de estudo proposto. Na subseção 8.2 são propostos trabalhos futuros.

## 8.1 Resultados obtidos

O objetivo geral deste trabalho foi atingido com sucesso, visto que o sistema *web* proposto foi desenvolvido e corrigiu as limitações identificadas no sistema *desktop* utilizado pela UFERSA no RU atualmente.

O Web RU ainda carece de validação em ambiente de produção. Não foi possível validar a funcionalidade de automatização de cadastros com base no consumo de dados vindos do SIGAA devido ao sistema não fornecer uma API externa e nenhum *endpoint* ter sido disponibilizado por parte da SUTIC. No entanto o sistema está apto a consumir arquivos provisionados de recursos externos tal como foi apresentado no Capítulo 7.

A disponibilização da funcionalidade de avaliação ainda carece de validação por parte da SUTIC.

A integração do Web RU com sistemas externos admite a apreciação de dados que permitem gerar relatórios mais precisos e relevantes para o fluxo atual das atividades do RU, sendo possível inativar benefícios concedidos à discentes que não estão frequentando aulas assiduamente.

A arquitetura e os métodos contemplados no Web RU permite a manutenção, distribuição e versionamento do sistema com mais facilidade, dado que as tecnologias utilizadas são consolidadas no mercado, recebem frequentes correções pontuais e são comuns à vários sistemas desenvolvidos atualmente.

O trabalho apresentado se demonstra relevante para o campo de estudo abordado, sendo aceito para publicação na revista *Brazilian Journal of*

*Development*[12], com avaliação Qualis Capes B2. Além disso, foi possível publicar um artigo científico (CARDOSO e ARAÚJO JÚNIOR, 2019), resultante desse trabalho, no Encontro de Computação do Oeste Potiguar (ECOP/UFERSA)[13].

## 8.2 Trabalhos futuros

As seguintes atividades são propostas como trabalhos futuros, visando a melhoria do Web RU:

Em função da indisponibilidade de testar o Web RU em ambiente de produção durante o período do desenvolvimento do trabalho de conclusão de curso II, se faz necessária a validação do sistema com usuários do RU nas 3 categorias (administrador, atendente e aluno) para que verifique-se se as demandas levantas pela SUTIC e implementadas nesse trabalho estão adequadas ao uso por parte da comunidade acadêmica.

Deseja-se integrar o aplicativo da UFERSA ao Web RU para que os alunos possam avaliar, a partir deste, aspectos referentes ao modo como o serviço está sendo prestado.

Com o objetivo de diminuir eventuais filas que possam se formar durante os horários de venda das refeições no RU, dado que os alunos precisam apresentar sua carteira de identificação para que sejam feitas algumas verificações por parte do sistema, é desejável desenvolver a funcionalidade de controle de créditos, sendo possível aos discentes comprarem esses créditos e com isso manterem um saldo virtual para utilizar na compra das refeições.

Com relação a funcionalidade de geração do relatório de histórico de consumo, pretende-se disponibilizar os resultados do quantitativo das refeições e valores totais a serem pagos pela UFERSA à empresa contratada em formato .CSV (*comma-separated values*), facilitando assim a auditoria na fiscalização dos contratos relacionados ao RU.

Por fim, pretende-se avaliar a segurança da aplicação em ambiente de produção, por se tratar de um contexto que lida com dados sensíveis dos discentes e valores monetários oriundos da venda das refeições e que

[12] http://brazilianjournals.com/index.php/BRJD
[13] https://periodicos.ufersa.edu.br/index.php/ecop/issue/view/218

precisam ser auditados devido a contratos firmados entre a UFERSA e a empresa contratada para o fornecimento do serviço da venda de refeições.

# 9 REFERÊNCIAS

BATISTELLA, Rafael. **Sistema web para controle de créditos em um restaurante universitário**. 2017. 54f. Monografia (Trabalho de especialização) – Departamento Acadêmico de Informática, Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. Pato Branco, 2017.

BARBOSA, Hefrayn Antero Barros. **ANÁLISE DAS VANTAGENS DO USO DE CI/CD NO DESENVOLVIMENTO DE UM PROJETO JAVA WEB.** 2018. 64 f. TCC (Graduação) - Curso de Engenharia de Computação, Centro Universitário de Anápolis - Unievangélica, Anápolis, 2018. Disponível em: http://repositorio.aee.edu.br/jspui/handle/aee/1097. Acesso em: 12 abr. 2021.

CARDOSO, J. V. C. e ARAÚJO. JÚNIOR, J. G. **Web RU: Uma nova proposta de Administração, Gestão e Controle de Restaurante Universitário.** *In*: ENCONTRO DE COMPUTAÇÃO DO OESTE POTIGUAR ECOP/UFERSA, 3., 2019, Pau dos Ferros. **Anais eletrônicos**... Pau dos Ferros: UFERSA, 2019. p. 250 – 257.

LOPES, N. M. C., GOMES, T. N., MENDES, R. F., CANTANO, M. M. R., e de Sousa, A. S. (2018). **Avaliação da eficácia e utilização do sigaa na formação discente**. *CIET*: EnPED.

OLIVEIRA, R. B. d. e FERREIRA, D. C. S. (2015). **Sistema de gestão e aplicação de avaliações sigaa**.

PESSÔA, Sabryna de Sousa. **Entrega contínua de software**: um estudo de caso. 2018. 57 f., il. Trabalho de Conclusão de Curso (Bacharelado em Engenharia de Software) - Universidade de Brasília, Brasília, 2018.

PRESSMAN, Roger S. **Engenharia de Software: uma abordagem profissional**. 7. ed. São Paulo: Pearson Makron Books, 2011.

SOMMERVILLE, Ian. **Engenharia de Software**. 8. ed. São Paulo: Pearson Addison Wesley, 2007.

POHL, Klaus; RUPP, Chris. **Fundamentos da Engenharia de Requisitos** - Um guia para o exame CPRE-FL em conformidade com o padrão IREB.