Web RU: Uma nova proposta de Administração, Gestão e Controle de Restaurante Universitário

João Vitor Costa Cardoso[1], José Gildo de Araújo Júnior[2]

[1,2]Universidade Federal Rural do Semi-Árido (UFERSA/Angicos)

{joaovitorcardoso31@gmail.com, jose.araujo@ufersa.edu.br}

Abstract. *The desktop software currently used to manage the activities of the University restaurant at (omitted due blind submission) presents a series of limitations regarding maintenance, distribution and versioning, which imply both the difficulty of implementing new functionalities and maintaining the functionalities existing. Based on technologies that favor productivity and interoperability, this work presents a web system proposal that solve the problems faced guaranteeing better management and control of the university restaurant by the competent team.*

Resumo. *O software desktop utilizado atualmente para gerenciar as atividades do Restaurante Universitário (RU) apresenta uma série de limitações referentes à manutenibilidade, distribuição e versionamento, que implicam tanto na dificuldade de implementar novas funcionalidades quanto de manter as funcionalidades existentes. Baseado em tecnologias que prezam pela produtividade e interoperabilidade, este trabalho apresenta uma proposta de sistema web que se propõe a solucionar os problemas enfrentados, garantindo assim, melhor administração e controle do restaurante universitário por parte da equipe competente.*

1. Introdução

O Restaurante Universitário (RU) da Universidade Federal Rural do Semi-Árido (UFERSA) tem o objetivo de oferecer refeições ao público acadêmico com qualidade e agilidade. Para isso, dispõe de recursos humanos e computacionais aptos a garantirem o melhor atendimento. Dentre os recursos computacionais utilizados, tem-se atualmente, o sistema desenvolvido pela própria instituição para gerenciar as atividades do RU.

O sistema utilizado atualmente foi concebido para uma arquitetura desktop e ao longo do tempo, apresentou uma série de limitações que inviabilizam sua manutenção, distribuição e versionamento. No contexto da Superintendência de Tecnologia da Informação e Comunicação (SUTIC) o ciclo de desenvolvimento desse sistema em específico impõe inúmeros desafios à equipe. No que concerne especificamente à manutenção, emergem inúmeras dificuldades. Inicialmente o software do RU foi concebido em linguagem *Delphi* que, por se tratar de um código legado não é familiar para a maioria dos servidores atuais o que por si só já compromete a manutenção. Com relação a distribuição e versionamento, cada nova atualização do sistema, implica em novo empacotamento e distribuição para todos os *campi*, o que sobrecarrega e inibe a eficiência de todo o processo. Por fim, a lógica do sistema, encontra-se atualmente dissociada da condição real dos alunos o que impede um melhor gerenciamento e controle. Um aluno pode, por exemplo, estar sendo beneficiado com o custeio da alimentação, mesmo não frequentando as aulas de forma assídua. Dessa forma, a Pró-reitora de Assuntos estudantis (PROAE) não toma conhecimento e não consegue notificar esses alunos nesse perfil para saber o motivo das faltas, ou caso necessário, inativar o

benefício junto ao RU. Essas e outras limitações fazem com que o fluxo de atividades gerenciais do RU percam celeridade, dificultando a gestão do sistema e o funcionamento ideal do serviço.

Isto posto, este trabalho tem o objetivo de apresentar uma solução de sistema *WEB* (Web RU) 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. Neste trabalho será abordada a validação do sistema, mas a concepção do projeto e as novas estruturas propostas.

2. Descrição do problema

O sistema utilizado atualmente, concebido para a arquitetura *desktop* e implementado na linguagem *Delphi* apresenta limitações que inviabilizam as atividades gerenciais do RU, entre elas:

- **Manutenção:** Questões referentes à orientação a objetos (encapsulamento, herança, polimorfismo), e *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 investir-se muitos recursos (tempo, capital humano, estrutura física) no processo de manutenção do software: localizar, corrigir e testar as falhas levantadas. Quanto mais o sistema cresce (implementação de novas demandas) mais complexo torna-se mantê-lo. Soma-se a isso, ainda, a carência de mão de obra apta a trabalhar com a linguagem *Delphi* no contexto atual da SUTIC;
- **Geração de Relatório:** Limitações referente à geração de relatórios condizentes com a necessidade da universidade. Atualmente, o sistema utilizado não gera o relatório de consumo (número de refeições vendidas) com os respectivos valores, sendo necessário realizar o cálculo do valor a ser pago para a prestadora de serviços do RU, manualmente. Além de tudo, o sistema não permite que a PROAE identifique alunos que estão sendo beneficiados com o custeio integral das refeições, mas que não estão frequentando as aulas de forma assídua.
- **Atualização de Dados:** Limitações referentes à importação de dados, que obriga a cada semestre a realização de cadastro manual para todos os ingressantes 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 em que o RU vai sendo estabelecido em outros *campi* da mesma universidade, a manutenção de um sistema *desktop* potencializa e inviabiliza a capacidade de manutenção por parte da SUTIC, uma vez que cada *campus* estará livre para gerir sua própria versão do sistema e executar localmente as atividades de RU;
- **Versionamento**: Dificuldade para produzir, manter e gerir novas versões do sistema do RU em decorrência dos problemas de manutenção, distribuição e disponibilização apresentados.

Diante disso, foi elaborada a proposta de adoção 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. Metodologia

A princípio o desenvolvimento da solução apresentada por este trabalho se deu por meio de uma reunião *kick-off*[1] com a SUTIC. Nessa reunião, foram discutidos os objetivos, descrição do projeto de software, cronograma, tecnologias a serem adotadas, entre outros assuntos. O fluxo de trabalho seguiu a metodologia *scrum* onde a SUTIC anotava as demandas a serem atendidas por ordem de prioridade em uma planilha de livre edição compartilhada entre todos os membros da equipe: funcionários da SUTIC, representando os clientes, o coordenador do projeto e o desenvolvedor conforme é apresentado pela **Figura 1**.

Figura 1. Roteiro de atividades listadas pelo cliente

Atividades	Status	Comentários
Estudar a documentação Electron: https://electronjs.org/	OK	
Apresentar no github o código produzido durante o estudo da 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 esboço 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 foram apresentadas ao cliente a cada 15 (quinze) dias. Nestes momentos as atividades eram apresentadas e o *feedback* fornecido era utilizado para conclusão das atividades.

Inicialmente foi implementado um protótipo do projeto que deveria ser disponibilizado *online* para receber críticas da SUTIC. Nessa etapa, foi possível elicitar os requisitos do sistema por meio da observação das formas de uso tomando-se em conta as restrições atuais.

Por possuir maior familiaridade entre os membros da SUTIC, e assim, possibilitar que a manutenção do sistema pudesse ser realizada por quaisquer de seus membros o novo sistema foi proposto em linguagem Java[2], por meio do *framework Vraptor*[3]. Por fim, foi proposto pela SUTIC, que a realização do *deploy* da aplicação ocorresse via *Heroku*[4] de modo a possibilitar testes de usuário diretamente sob as funcionalidades. Visto que os desenvolvedores já estavam proficientes na tecnologia proposta e que as fases de elicitação de requisitos e projeto já haviam sido completadas, a próxima etapa,

[1] Reunião feita no início do projeto para definir vários assuntos referentes ao desenvolvimento e implantação do software.
[2] https://www.java.com/pt_BR/download/
[3] http://www.vraptor.org/pt/
[4] https://www.heroku.com/

consistiu em desenvolver o fluxo de atividades e implementação das funcionalidades do sistema.

5. Web RU

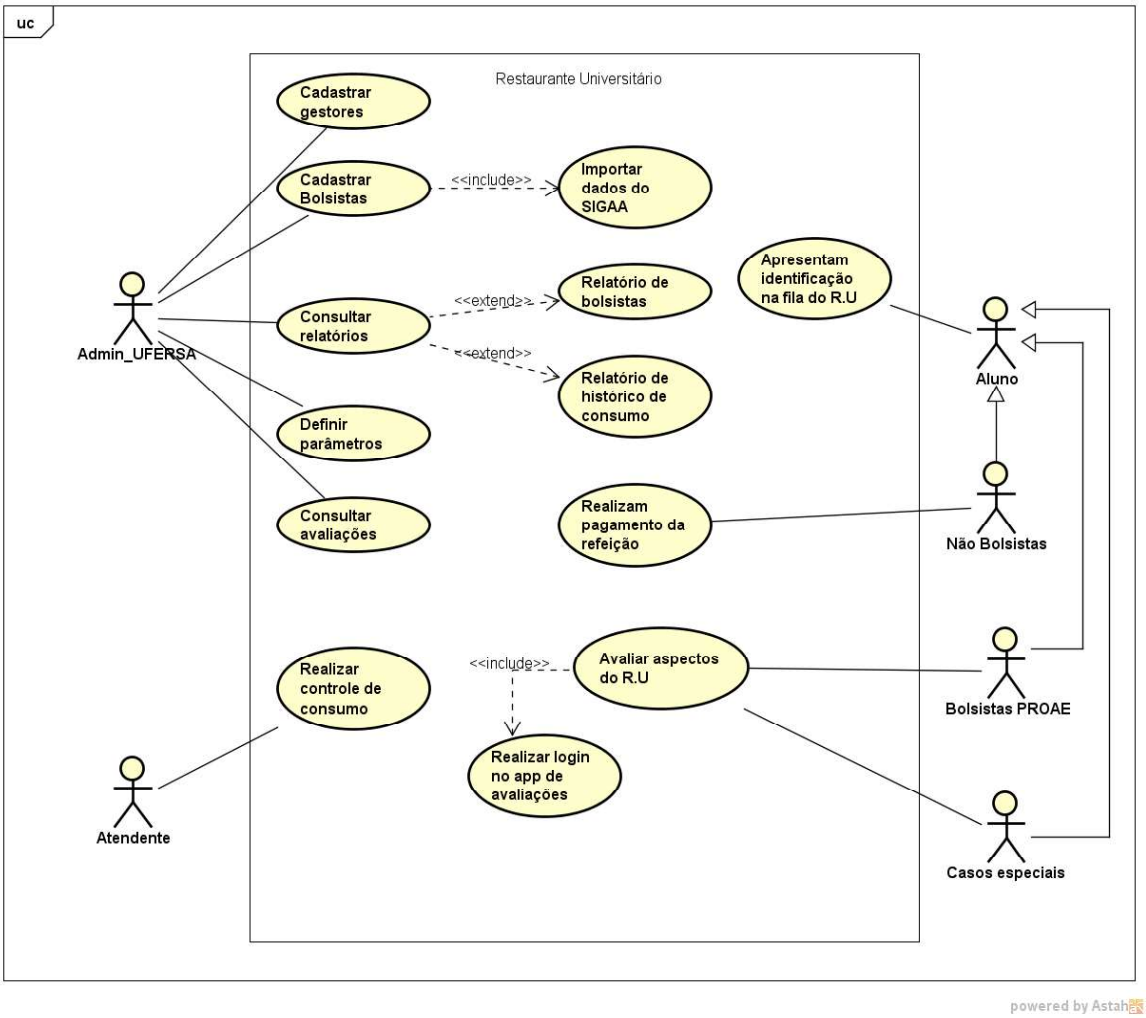
A **Figura 2** apresenta o diagrama de caso de uso proposto para a nova versão do sistema de RU. A utilização do diagrama facilita a comunicação entre o cliente e o desenvolvedor, pois é uma descrição visual de alto nível das funcionalidades que o cliente necessita. Dessa forma é possível extrair dos casos de uso as funcionalidades que devem ser implementadas.

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 falhas que podem ser sanadas obtendo-se as informações já presentes no Sistema Integrado de Gestão de Atividades Acadêmicas (SIGAA) (Oliveira and Ferreira 2015, Lopes et al. 2018). Nessa perspectiva, para automatizar a coleta de dados foi proposto, a integração do novo sistema do RU diretamente com o SIGAA de modo a importar as informações já corretas e consolidadas. Essa integração é realizada por meio da utilização dos serviços *REST* (*Representational State Transfer*) que a API (*Application Programming Interface*) do SIGAA disponibiliza. Dessa forma, o sistema proposto importa as informações do SIGAA para povoar o cadastro e eliminar a necessidade de povoamento manual.

Ainda com relação à integração entre os sistemas do RU e o SIGAA, a solução proposta permite ao administrador acessar relatórios sobre o *status* da matrícula do aluno na instituição, isto é, uma vez que o aluno deixe de possuir vínculo com a instituição será possível inativar seu benefício referente ao auxílio alimentação junto ao RU. Outra vantagem deste relatório gerencial é a possibilidade de verificar a assiduidade do aluno nas aulas e dessa forma notificá-lo, caso este esteja utilizando o benefício do auxílio alimentação, mas que não esteja frequentando as aulas regularmente.

Por fim, a nova versão do sistema de RU se propõe a implantar a avaliação dos serviços prestados pelo restaurante. Aspectos como a qualidade da comida, limpeza do ambiente, entre outros fatores passíveis de serem avaliados por todos os usuários do restaurante. O *feedback* obtido é enviado para os responsáveis de modo a utilizar a retroalimentação do serviço para melhoria de todos os serviços sendo prestados.

Figura 2. Diagrama de caso de uso do Web RU



Fonte: Autoria Própria (2018)

As funcionalidades do Web RU apresentadas no diagrama de casos de uso estão distribuídas entre os módulos conforme ilustra a **Tabela 1**.

Tabela 1. Distribuição das funcionalidades do Web RU entre os módulos

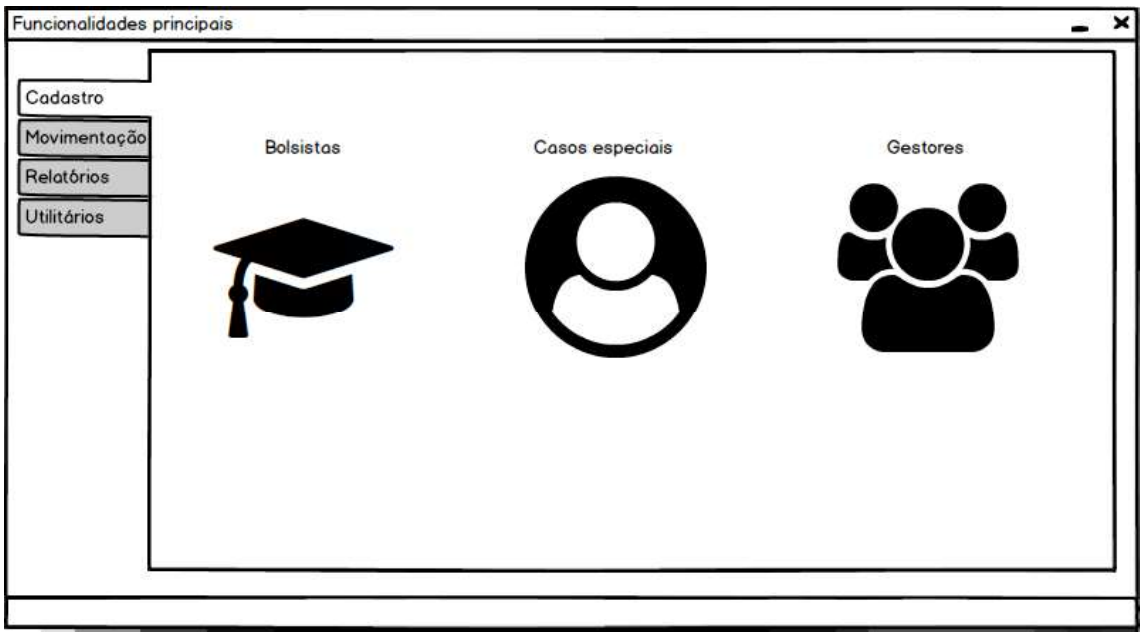
Módulo	Funcionalidades
Cadastro	Cadastro de alunos, gestores e parâmetros do sistema.
Consulta	Relatório de histórico de consumo, bolsistas.
Movimentação	Gerenciamento de controle de consumo.

Fonte: Autoria Própria (2019)

6. Prototipação do Web RU

A **Figura 3** apresenta uma imagem do protótipo funcional desenvolvido. A ideia central foi desenvolver uma interface que minimizasse o erro do operador do sistema, com o menor número de cliques e o menor esforço por buscar informações deixando toda a complexidade envolvida dentro do código fonte.

Figura 3. Protótipo funcional do Web RU



Fonte: Autoria Própria (2018)

Ainda com relação a **Figura 3** é possível visualizar o protótipo da interface inicial do sistema. Nela estão exemplificados por meio da aba **Cadastro** as funcionalidades referentes às operações de gerenciamento dos alunos bolsistas PROAE, casos especiais e gestores. Os alunos bolsistas são aqueles cujo o valor das refeições (almoço e jantar) são custeados integralmente pela UFERSA e são assistidos pela unidade PROAE. Já os alunos *casos especiais*, têm apenas uma parte da sua refeição custeada, sendo a outra parte paga pelo aluno.

A funcionalidade de gerenciamento de gestores, é responsável por manter os dados dos gestores no sistema. Os gestores podem assumir três papéis diferentes no contexto do Web RU: o usuário com o perfil de **ADMIN_UFERSA** é um funcionário lotado em alguma unidade na UFERSA que tem relação com o RU. Como, por exemplo, um funcionário da PROAE pode cadastrar alunos, 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 permitido-lhe cadastrar uma movimentação de uma refeição.

A aba **Movimentação** dá acesso à funcionalidade de controle de movimentação. Essa funcionalidade registra o fluxo de refeições vendidas no RU. Dessa forma, ao passar um aluno pelo o atendente e apresentar sua identificação 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 (relatório) que, por sua vez serve para fins de contagem

do número de refeições vendidas e o valor em dinheiro a ser pago pela UFERSA a empresa que presta seus serviços ao RU.

Na aba **Utilitários** é possível ter acesso aos parâmetros do sistema. Esses parâmetros são gerenciados (cadastro, exclusão, visualização e alteração) e servem para, por exemplo, na tela de cadastro de movimentação, os horários de início das refeições e os valores padrões das mesmas, são recuperados para fins de cálculo e controle pelo sistema e, para fins de exibição ao atendente.

Na aba de **Relatórios**, é possível ter acesso aos relatórios que descrevem o quantitativo de refeições e valores que devem ser pagos pela UFERSA, relação de bolsistas, avaliações do usuários e assiduidade dos bolsistas.

O protótipo do Web RU foi desenvolvido em dois níveis de prototipação: baixa e média fidelidade. O protótipo de baixa finalidade foi desenvolvido utilizando a ferramenta Balsamiq Wireframes[5], onde foi possível desenhar as telas do sistema em preto e branco. O protótipo de média fidelidade foi elaborado utilizando a ferramenta *online Invision*[6]. Com essa ferramenta foi possível criar interações de cliques a partir das imagens produzidas no protótipo de baixa fidelidade. Essas interações permitiram ao cliente navegar entre as telas do sistema e fornecer comentários sobre as mesmas na própria ferramenta. Dessa forma, foi possível validar as funcionalidades propostas, pois foi possibilitado ao cliente testar as interações referentes às funcionalidades do sistema.

7. Conclusão

Este trabalho se propôs a apresentar uma proposta de *software* que resolvesse os problemas do sistema que vem sendo utilizado atualmente pela SUTIC para gerenciar as atividades do RU. A nova abordagem sendo desenvolvida, além de resolver questões referentes à manutenção, disponibilização, versionamento, atualização, e geração de relatórios, estabelece integração direta com o SIGAA de modo a conceber duas atividades principais: automatização e controle. Da automatização, elimina-se a necessidade de povoamento manual de dados a cada novo usuário, minimizando, assim problemas oriundos de erros humanos e laboriosos trabalhos manuais. Do controle, ampliam-se as possibilidades de auditoria sobre a concessão de benefícios, bloqueando ou notificando alunos não matriculados ou faltosos quanto sua condição e pendências. Por fim, o módulo avaliação dos serviços do RU, permitirá que os usuários possam criticar e elogiar os serviços prestados e, dessa forma, contribuir para o aperfeiçoamento do serviço.

O desenvolvimento deste trabalho está integrado ao projeto TCC-SIGAA, que oportuniza alunos de graduação à possibilidade de desenvolverem soluções de *software* que sejam relevantes à comunidade acadêmica em seu entorno.

8. Reprodutibilidade

Todo código utilizado por este trabalho pode ser acessado e encontra-se disponível em https://github.com/vitorcardoso98/TCC-SIGAA.

[5] https://balsamiq.com/wireframes/

[6] https://www.invisionapp.com/

Referências

Oliveira, R. B. d. and Ferreira, D. C. S. (2015). Sistema de gestão e aplicação de avaliações sigaa.

Disponível em: < https://www.java.com/pt_BR/download/ >. Acesso em: 15 maio 2019.

Disponível em: < https://balsamiq.com/wireframes/ >. Acesso em 15 maio 2019.

Disponível em: < http://www.vraptor.org/pt/ >. Acesso em: 15 maio 2019.

Disponível em: < https://www.invisionapp.com/ >. Acesso em 15 maio 2019.

Disponível em: < https://www.heroku.com/ >. Acesso em: 15 maio 2019.