

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO
RIO GRANDE DO SUL
CAMPUS CANOAS
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE
SISTEMAS

AUGUSTO RAFAEL SANTOS DA SILVA

**YAFMA - Aplicação web para auxiliar no
planejamento financeiro pessoal**

Canoas, 25 de julho de 2022.

AUGUSTO RAFAEL SANTOS DA SILVA

**YAFMA - Aplicação web para auxiliar no
planejamento financeiro pessoal**

Trabalho de Conclusão de Curso
apresentada como requisito parcial para
obtenção do grau de Tecnólogo em Análise
e Desenvolvimento de Sistemas pelo
Instituto Federal de Educação, Ciência e
Tecnologia do Rio Grande do Sul –
Campus Canoas.

Profa. Dra. Carla Odete Balestro Silva



Ministério da Educação
Secretaria de Educação Profissional, Científica e Tecnológica
Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul
Campus Canoas

ATA DE DEFESA PÚBLICA DO TRABALHO DE CONCLUSÃO DE CURSO

Ao primeiro dia do mês de agosto do ano de 2022, às 14 horas, em sessão pública na sala virtual do Google Meet (<https://meet.google.com/ttz-dafh-mwx>) do Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul, Campus Canoas, na presença da Banca Examinadora presidida pelo(a) Professor(a):

Dra. Carla Odete Balestro Silva e composta pelos examinadores:

1. Prof. Dra Patricia Nogueira Hubler;
2. Prof. Dr. Rafael Coimbra Pinto;

o(a) aluno(a) AUGUSTO RAFAEL SANTOS DA SILVA apresentou o Trabalho de Conclusão de Curso intitulado: "YAFMA - Aplicação web para auxiliar no planejamento financeiro pessoal" como requisito curricular indispensável para a integralização do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas. Após reunião em sessão reservada, a Banca Examinadora deliberou e decidiu pela APROVAÇÃO do referido trabalho, divulgando o resultado formalmente ao aluno e demais presentes e eu, na qualidade de Presidente da Banca, lavrei a presente ata que será assinada por mim, pelos demais examinadores e pelo aluno.

Carla Odete Balestro Silva

PATRICIA NOGUEIRA Assinado de forma digital por
PATRICIA NOGUEIRA
HUBLER:710638890
49 Data: 2022.08.01 15:27:55
-03'00'

Patricia Nogueira Hubler

Rafael
Coimbra
Pinto

Rafael Coimbra Pinto

Augusto Rafael Santos da Silva

RESUMO

Educação financeira é um tópico muito importante para a vida de qualquer adulto na nossa sociedade. Este trabalho implementou um sistema para ajudar indivíduos realizar o planejamento financeiro individual ou familiar. O levantamento dos requisitos foi feito com base no problema central: planejamento financeiro e na análise de aplicativos já existentes no mercado. O desenvolvimento do sistema, no lado do cliente foi feito com HTML, CSS, Javascript com a biblioteca React. No servidor, também foi utilizado Javascript com Node.js em conjunto com um banco de dados PostgreSQL e a *engine* Hasura para gerar APIs GraphQL. Os objetivos dessa pesquisa foram validados com usuários e, após a utilização do sistema, responderam um questionário de pesquisa para avaliação.

Palavras-Chave: Educação Financeira. Planejamento financeiro. Aplicativo web. Tecnologia da informação.

ABSTRACT

Financial literacy is a topic really important in the life on any adult in our society. This paper implemented a software to help individuals realize the financial planning individual or for the entire family. The requirements for this software were gathered over the main problem: financial planning and analyzing existing apps already established in the market. The development of this system, on the client side, was made with HTML, CSS and Javascript with the React library. On the server, was also made use of javascript with Node.js combine with the PostgreSQL database and Hasura engine to generate GraphQL APIs. The objectives for this research were validated by users and, after testing the system, they answered a form to evaluate the software.

Keywords: Financial literacy. Financial Planning. Web application. Information Technology.

LISTA DE FIGURAS

Figura 1: Realizam algum tipo de controle financeiro	13
Figura 2: Formas de organizar as finanças	14
Figura 3: Pessoas interessadas em uma aplicação web para planejamento financeiro.....	14
Figura 4: Aplicativo Bills Reminder	17
Figura 5: Aplicativo Timely Bills.....	19
Figura 6: Aplicativo Olivia AI	20
Figura 7: Aplicativo Mobills	22
Figura 8: Diagrama de caso de uso	34
Figura 9: Diagrama de Atividade da criação de conta	36
Figura 10: Diagrama Entidade-Relacionamento.....	37
Figura 11: Tela inicial do sistema.....	43
Figura 12: Tela de cadastro do usuário.....	44
Figura 13: Tela principal do sistema.....	45
Figura 14: Modal de criação de conta	46
Figura 15: Opção de repetições de conta.....	47
Figura 16: Opção de vincular conta com Google Calendar	48
Figura 17: Janela de login Google.....	49
Figura 18: Tela de aviso de aplicativo não validado	50
Figura 19: Tela de fornecer permissão ao calendário.....	51
Figura 20: Aba próximas contas.....	52
Figura 21: Modal de pagamento	53
Figura 22: Modal edição de conta	54
Figura 23: Modal de confirmação de exclusão	55
Figura 24: Aba de contas atrasadas.....	56
Figura 25: Aba de contas não pagas.....	57
Figura 26: Aba de contas pagas	58
Figura 27: Modal de cadastro de salário	59
Figura 28: Modal para adicionar renda extra.....	60
Figura 29: Mensagem informando que o usuário pode realizar determinada compra.....	61
Figura 30: Mensagem informando que o usuário não pode realizar determinada compra...	61
Figura 31: Gráfico de Gastos do mês atual	62
Figura 32: Gráfico de previsão de gastos dos últimos seis meses	63
Figura 33: Gráfico de previsão de gastos por categoria de gastos	64
Figura 34: Gráfico de histórico de gastos	65
Figura 35: Interface do sistema em dispositivos mobiles.....	66
Figura 36: Testes unitários no <i>front-end</i>	67
Figura 37: Testes unitários no <i>back-end</i>	68
Figura 38: Testes <i>end-to-end</i> com Cypress	69
Figura 39: Gênero dos participantes	81
Figura 40: Idade dos participantes	81
Figura 41: Nível de escolaridade.....	82
Figura 42: Realizam alguma atividade remunerada	82
Figura 43: Realizam controle financeiro	82
Figura 44: Utilizam algum aplicativo ou site para organização	83
Figura 45: Costumam planejar em poupar dinheiro.....	84
Figura 46: Participantes endividados	84

Figura 47: Tipos de despesas	85
Figura 48: Funcionalidades essenciais para um sistema de finanças pessoais.....	85
Figura 49: Participantes interessados em um site de planejamento financeiro.....	86
Figura 50: Considerações finais.....	86
Figura 51: Gênero dos participantes	87
Figura 52: Idade dos participantes	87
Figura 53: Escolaridade dos participantes.....	88
Figura 54: Grau de facilidade para se cadastrar.....	88
Figura 55: Grau de facilidade para criar uma conta.....	89
Figura 56: Grau de facilidade para visualizar dados financeiros.....	89
Figura 57: Utilização do Google Calendar.....	90
Figura 58: A aplicação ajudou no planejamento financeiro?	90
Figura 59: Observações finais.....	91

LISTA DE QUADROS

Quadro 1: Comparativo entre as funcionalidades dos aplicativos.....	23
Quadro 2: Especificação dos casos de usos.....	77

LISTA DE ABREVIATURAS E SIGLAS

API	<i>Application Programming Interface</i>
AWS	<i>Amazon Web Services</i>
CdU	Caso de Uso
CSS	Cascade StyleSheet
JWT	<i>JSON Web Token</i>
HTML	HyperText Markup Language
HTTP	<i>Hypertext Transfer Protocol</i>
SGBD	Sistema gerenciador de banco de dados
SPA	<i>Single Page Application</i>
UML	Unified Modeling Language
YAML	<i>YAML Ain't Markup Language</i>

SUMÁRIO

1	INTRODUÇÃO.....	11
1.1	MOTIVAÇÃO	12
1.2	OBJETIVOS.....	15
1.2.1	Objetivo Geral.....	15
1.2.2	Objetivos Específicos.....	15
2	ESTADO DA ARTE.....	16
2.1	Trabalhos acadêmicos.....	16
2.2	Aplicações comerciais	17
2.2.1	Bills Reminder.....	17
2.2.2	TimelyBills	18
2.2.3	Olivia AI	20
2.2.4	Mobills	21
2.2.5	Quadro comparativo	23
3	REVISÃO BIBLIOGRÁFICA.....	25
3.1	Educação Financeira	25
3.2	Regra 50-30-20.....	27
3.3	Engenharia de Software	28
3.4	UML.....	29
3.5	Arquitetura Cliente-Servidor.....	30
4	METODOLOGIA	32
5	DESENVOLVIMENTO	34
5.1	Modelagem.....	34
5.2	Tecnologia	39
5.3	Aplicação	42
5.4	Testes.....	67
6	CONSIDERAÇÕES FINAIS	71
6.1	Trabalhos Futuros.....	71
7	REFERÊNCIAS	73
	APÊNDICE A – Especificação dos Casos de Usos.....	77
	APÊNDICE B – Questionário sobre educação na íntegra	81
	APÊNDICE C – Questionário para a avaliação da aplicação na íntegra.....	87

1 INTRODUÇÃO

O objeto dessa pesquisa foi desenvolver o trabalho de conclusão do curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas. O propósito principal era auxiliar a resolver alguns problemas relacionados à organização financeira pessoal ou familiar, mais especificamente, o planejamento financeiro. Para isso, o objetivo foi desenvolver um sistema web para auxiliar o planejamento financeiro do usuário.

A principal motivação para o desenvolvimento de uma aplicação para auxiliar no planejamento financeiro é o baixo nível de educação financeira da população brasileira. Uma má gestão das próprias finanças pode acarretar diversos problemas na vida das pessoas. Grande parte dos brasileiros possui algum tipo de endividamento. Segundo a reportagem da CNN, o endividamento das famílias brasileiras chegou a 77,7% no mês de abril de 2022 (PUENTE, JANONE, 2022).

Pensando nisso, para tentar auxiliar na mitigação desse problema, foi desenvolvida uma aplicação web cuja finalidade é centralizar as contas a pagar que uma pessoa ou família tenha, notificar quando a data de vencimento estiver próxima e exibir alguns gráficos para informar ao usuário dados relacionados às suas finanças.

A pesquisa aqui apresentada é de abordagem qualitativa e natureza aplicada, com o produto sendo uma aplicação web. O sistema foi desenvolvido na linguagem Javascript no *front-end* em conjunto com a biblioteca React. No *back-end* foi utilizado um servidor GraphQL, o Hasura e Node.js para algumas APIs REST e os dados armazenados em um banco de dados Postgresql¹.

Depois da aplicação concluída e hospedada em serviço gratuito, foram realizados testes unitários com o Jest, *end-to-end* com o Cypress e testes de validação com o usuário final que atestaram que a aplicação era funcional e atingia os objetivos propostos.

Neste trabalho, temos os seguintes capítulos: a motivação, onde será explicado com mais detalhes porque essa pesquisa será desenvolvida. Seguido dos objetivos gerais e específicos que este trabalho buscará atingir; da metodologia para

¹ Detalhes sobre as tecnologias e ferramentas serão apresentados nos capítulos subsequentes.

explicar as etapas da pesquisa; do desenvolvimento da aplicação, seus testes e, por fim, as referências.

1.1 MOTIVAÇÃO

No Brasil, existe um alto índice de endividamento e um baixo nível de educação financeira da população brasileira. Segundo a pesquisa S&P Ratings Services Global Financial Literacy Survey² feita com 174 países do mundo, o Brasil ocupa a 74^o colocação, ficando atrás de alguns países do mundo considerados menos desenvolvidos, como Zimbábue (KLAPPER, LUSARDI, OUDHEUSDEN, 2015).

Isso, combinado com o nível de endividamento da população é extremamente elevado, chegando aos 66,5% no final de 2020, de acordo com a reportagem feita pela Agência Brasil (ABDALA, 2021). Por consequência, isso impacta diretamente na qualidade de vida das famílias brasileiras, podendo até mesmo desestruturar o núcleo familiar (ABDALA, 2021).

Porém, sabe-se que a pouca educação financeira não é o único fator para o endividamento da população, em especial diante da crise financeira do Brasil. Contudo, nesse trabalho não se discutirá os demais aspectos relacionados ao endividamento e será abordado somente o aspecto dos impactos do planejamento financeiro.

De acordo com o Nubank (2020), um planejamento financeiro é uma forma de começar a ter mais controle sobre o orçamento, servindo como um guia para a organização correta do próprio dinheiro. Pensando nisso, ferramentas que auxiliam na organização financeira são essenciais para facilitar a vida financeira das pessoas (NUBANK, 2020).

A fim de obter mais informações sobre o comportamento das pessoas em relação ao planejamento financeiro, foi realizada uma pesquisa³ com a utilização de questionários com perguntas fechadas a respeito do assunto. Os questionários foram criados na plataforma Google Forms⁴ e distribuídos através das redes sociais

² Pesquisa Global de Educação Financeira da divisão de ratings e pesquisas da Standard & Poor's, traduzido para a Língua Portuguesa.

³ Formulário na íntegra pode ser visualizado no apêndice B.

⁴ <https://www.google.com/intl/pt-BR/forms/about/>

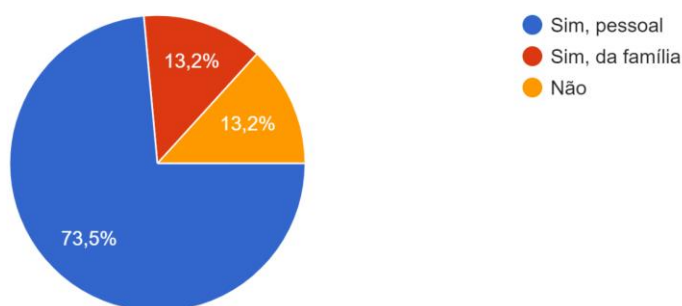
Facebook⁵ e Reddit⁶ e por grupos no Whatsapp⁷ com o objetivo de abranger diversos perfis.

O questionário foi respondido por 79 pessoas, com idades entre 15 a 45 anos. Sendo 79,8% com ensino superior incompleto, concluído ou com algum nível de pós-graduação.

Foi perguntado se os respondentes realizam algum tipo de atividade remunerada para obter as informações relacionadas ao planejamento financeiro. Dos 79 participantes, 68 realizam algum tipo de atividade remunerada. Dessas 68 pessoas, 59 pessoas realizam algum tipo de controle financeiro como mostra a Figura 1.

Figura 1: Realizam algum tipo de controle financeiro

Você realiza algum tipo de controle financeiro das suas finanças pessoais ou da sua família?
68 respostas



Fonte: Autoria própria.

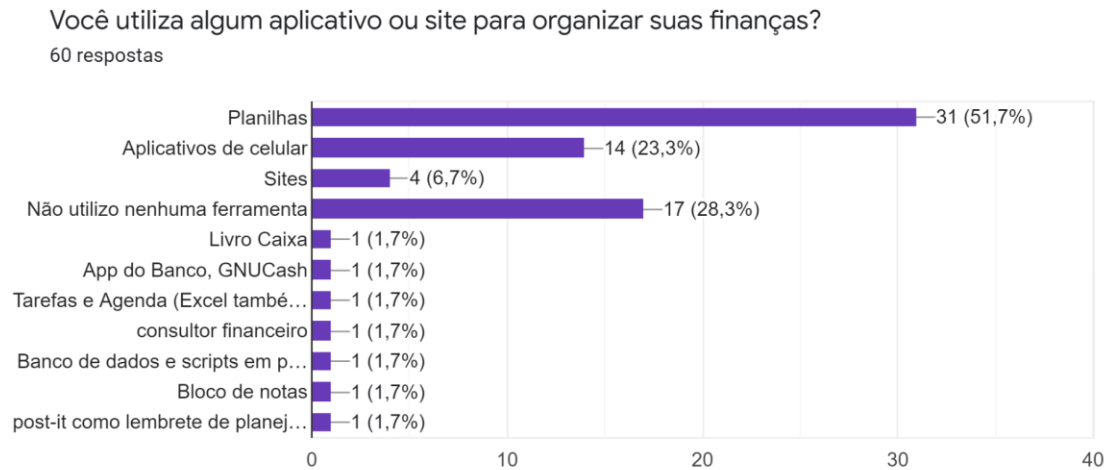
Sobre o tipo de controle financeiro, mais da metade das pessoas utilizam planilhas para fazer isso, como demonstra a Figura 2, mas quase um terço não utiliza nenhuma ferramenta para isso. Por fim, a Figura 3 mostra que mais de 68% dos respondentes tem interesse em um sistema web para auxiliar no planejamento financeiro.

⁵ <https://facebook.com/>

⁶ <https://www.reddit.com/>

⁷ <https://www.whatsapp.com/>

Figura 2: Formas de organizar as finanças

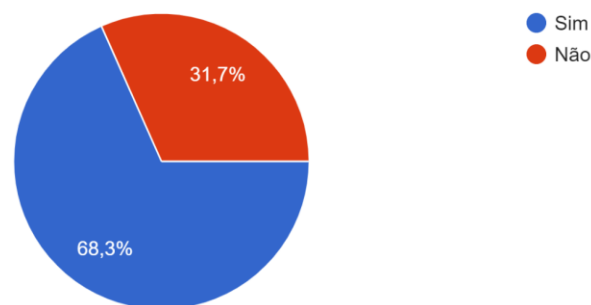


Fonte: A autoria própria.

Figura 3: Pessoas interessadas em uma aplicação web para planejamento financeiro

Você teria interesse em um site para realizar a gestão das suas finanças pessoais?

60 respostas



Fonte: A autoria própria

Considerando o exposto e a possibilidade de designar uma parte da responsabilidade do gerenciamento das finanças pessoais para um sistema informatizado, assim, reduzindo a probabilidade do erro humano, a proposta dessa pesquisa foi desenvolver uma aplicação para auxiliar no planejamento financeiro a partir do cadastro das contas a serem pagas, alerta dessas contas e o comparativo dos gastos com os valores recebidos pelo usuário. A aplicação se chama YAFMA

que é um acrônimo para a frase “Yet Another Financial Managment App”⁸. O nome foi inspirado pela linguagem de marcação YAML que significava, originalmente, “Yet Another Markup Language”⁹.

1.2 OBJETIVOS

No subcapítulo abaixo estão expostos os objetivos gerais e específicos deste trabalho.

1.2.1 Objetivo Geral

Desenvolver um sistema web para auxiliar o planejamento financeiro do usuário.

1.2.2 Objetivos Específicos

- Estudar sobre educação financeira e a sua importância;
- Realizar análise de requisitos e modelar o sistema;
- Escolher as tecnologias adequadas à implementação;
- Implementar um protótipo para validar a ideia;
- Liberar o protótipo para testes para o usuário final.

⁸ Hospedado temporariamente no endereço <https://yafma-app.herokuapp.com/>

⁹ Atualmente, o nome significa YAML Ain't Markup Language

2 ESTADO DA ARTE

Neste capítulo, estão relacionados os trabalhos acadêmicos e algumas aplicações comerciais já existentes cuja finalidade se assemelha à aplicação a ser desenvolvida.

2.1 Trabalhos acadêmicos

Em uma busca no Google Scholar com as palavras chaves “*financial literacy*”, “*financial planning*” e “benefícios planejamento financeiro” para achar artigos relacionados a planejamento financeiro foi encontrado o artigo “PLANEJAMENTO FINANCEIRO E SEUS BENEFÍCIOS” escrito por Vanderlinde e Godoy.

O objetivo do trabalho era mostrar a importância e os benefícios do planejamento e controle financeiro pessoal e empresarial. Porém, aqui será focado somente na parte do planejamento financeiro pessoal (VANDERLINDE, GODOY, 2014).

De acordo com os autores,

o planejamento financeiro familiar pode ajudar as famílias no papel de formadoras de indivíduos críticos e socialmente livres e independentes. Além disso, através de um planejamento é possível estabelecer metas de consumo realistas e planejar aquisições de médio e longo prazo, ou ainda, investir em algo de significativo retorno, como educação, moradia própria ou lazer.(VANDERLINDE, GODOY, 2014).

Logo, é necessário criar prioridades dos gastos, controlando o quanto se ganha e gasta no mês. Caso contrário, o consumo inconsequente pode ser muito prejudicial ao orçamento doméstico, pois as necessidades básicas ficam em segundo plano e se acumulam coisas supérfluas que não são realmente importantes para a família (VANDERLINDE, GODOY, 2014).

Os autores deixam claro a importância de se realizar orçamentos, registrar as receitas e despesas, para ter certeza de como e quando consumir. Assim é possível ajudar as pessoas a ficarem mais independentes, ensinando a criar metas de consumo mais realistas (VANDERLINDE, GODOY, 2014).

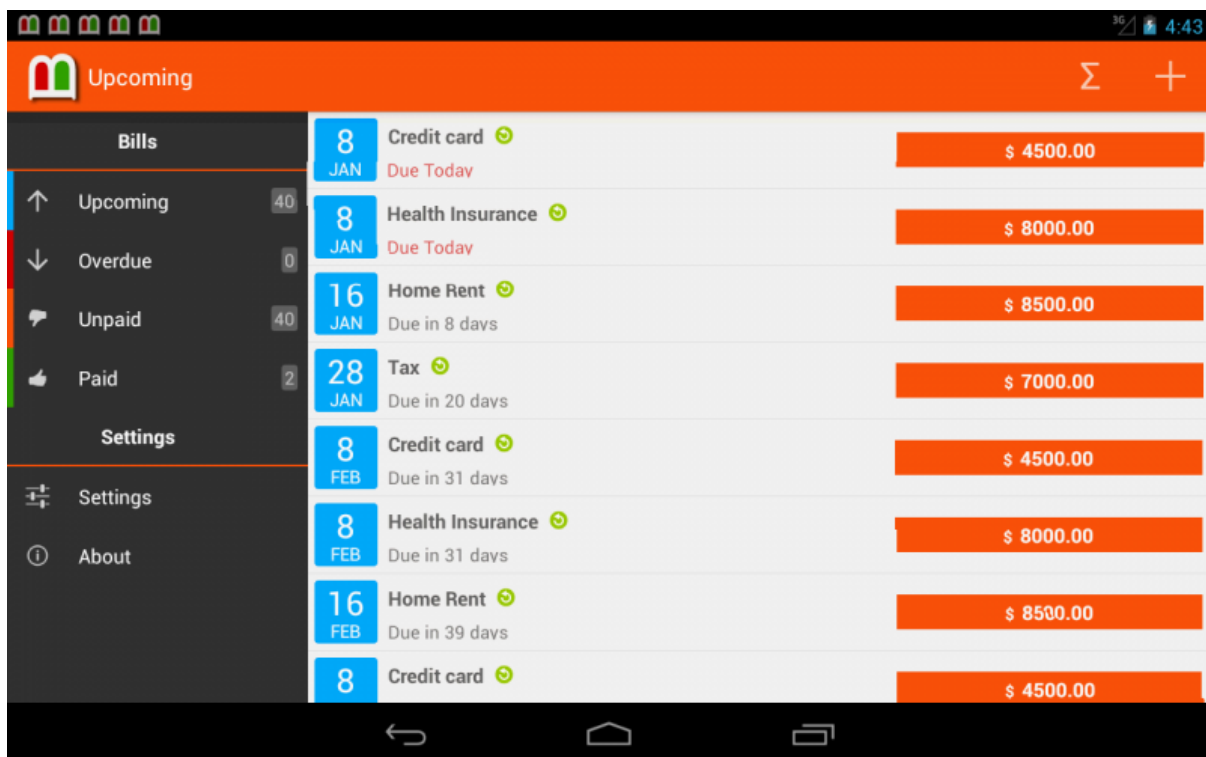
2.2 Aplicações comerciais

Neste subcapítulo foram analisadas algumas aplicações comerciais que lidam com a perspectiva de controle de gastos, orçamentos e planejamento financeiro.

2.2.1 Bills Reminder

Bills Reminder¹⁰ é um aplicativo mobile gratuito para organizar as suas contas e receber notificações para lembrar-se de pagá-las. Ele é focado na funcionalidade de lembrete de contas e não tem outras funcionalidades complementares. A Figura 4 mostra a tela principal do aplicativo.

Figura 4: Aplicativo Bills Reminder



fonte: <https://bills-reminder.en.aptoide.com/app>

¹⁰ Disponível em <https://play.google.com/store/apps/details?id=com.amazier.apps.billsreminder>

O funcionamento deste aplicativo é simples: o usuário preenche um formulário com os dados da conta a pagar e o aplicativo irá lembrá-lo quando a data de vencimento estiver próxima.

No cadastro de uma conta, o aplicativo é bem flexível. Você pode criar contas recorrentes, com datas de repetições configuráveis como, a cada 30 dias, a cada 6 meses ou ano, etc. O usuário também tem a opção de colocar observações, link para o pagamento da conta, categoria e alguma observação.

Além disso, é possível também configurar com quantos dias de antecedência da data de vencimento que o aplicativo irá enviar a notificação. E por fim, esse aplicativo tem relatórios de forma gráfica para apresentar informações relacionadas aos gastos para o usuário.

Contudo, existem algumas funcionalidades que não estão presentes neste aplicativo. O mais relevante seria o fato de não haver um aplicativo web e nenhuma integração com o Google Calendar. Um comparativo de todas as funcionalidades será demonstrado mais à frente.

2.2.2 TimelyBills

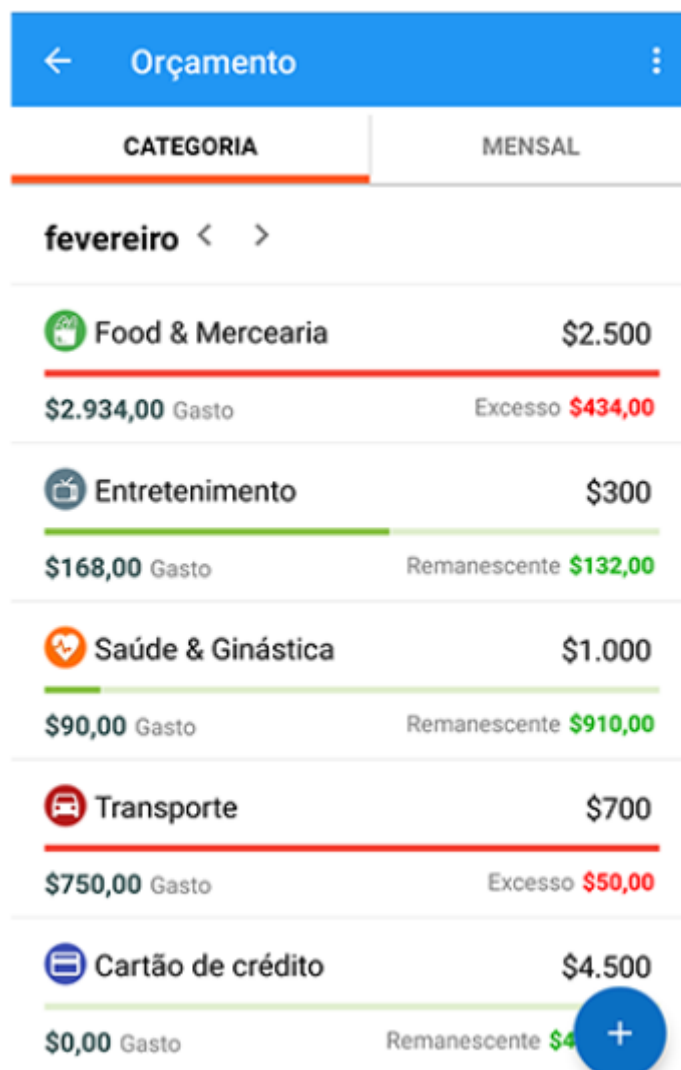
TimelyBills¹¹ é um aplicativo mobile com objetivo de ser um gerenciador das suas finanças pessoais. Ele possui algumas funcionalidades pagas, contudo, as funcionalidades principais relacionadas as gestões das finanças estão todas disponíveis gratuitamente.

Possui a funcionalidade de cadastrar as contas para a notificação do usuário, inclusive permitindo customização a partir de quantos dias antes do vencimento da conta que ele enviará a notificação.

Possui também outras funcionalidades como gerenciador de entrada e saída de dinheiro, planejamento de orçamento mensal para criar um limite de gastos para, por exemplo, aluguel, lazer, roupas, etc. E, assim como o Bills Reminder, também possui funcionalidades relacionadas a geração de relatórios dos gastos mensais podendo classificar por categoria. A Figura 5 apresenta uma aba do aplicativo.

¹¹ Disponível em <https://www.timelybills.app>

Figura 5: Aplicativo Timely Bills



fonte: <https://www.timelybills.app/>

Entretanto, existem propagandas no aplicativo e para removê-las é necessário pagar. É necessário pagar quarenta e nove reais e noventa e nove centavos para removê-las permanentemente ou assinar modo *premium* para desbloquear algumas funcionalidades como a integração com Google Calendar, download de relatórios para Microsoft Excel ou em arquivo formato PDF, acesso com biometria, etc. A assinatura custa quinze reais e noventa e nove centavos mensais, cinquenta reais e noventa e nove centavos por semestre ou setenta e sete reais e noventa e nove centavos por ano.

2.2.3 Olivia AI

Olivia AI¹² é um aplicativo gratuito de assistente financeiro que utiliza inteligência artificial para auxiliar o usuário. A Figura 6 mostra a tela principal do aplicativo.

Figura 6: Aplicativo Olivia AI



Fonte: <https://www.olivia.ai>

¹² Disponível em <https://www.olivia.ai>

As funcionalidades que mais diferenciam esse aplicativo são a integração com instituições financeiras, como bancos, vale alimentação, cartões de crédito. Conta com uma comunidade para a troca de informações entre os usuários. Além disso, dentro do aplicativo, é possível calcular se é recomendado realizar uma nova compra utilizando dados dos gastos mensais e a prevendo gastos futuros de acordo com o comportamento do usuário.

O aplicativo Olivia AI também possui a função de cadastrar lembretes de contas recorrentes, porém, é mais limitado comparado aos outros dois aplicativos. Além disso, não é possível criar categorias customizadas para os gastos ou customizar o período das notificações.

Porém, esse aplicativo também não possui uma versão web. E também, as funções relacionadas à classificação e organização de contas são bem limitadas. Como por exemplo, não existe uma forma de classificar uma conta como atrasada ou paga.

2.2.4 Mobills

Mobills¹³ também é outro aplicativo mobile, porém, diferente dos outros citados, tem uma versão web. O Mobills possui uma versão gratuita e outra paga para liberar mais funcionalidades. O aplicativo possui duas formas de assinatura, a mensal, custando dezoito reais e noventa centavos e a anual, custando noventa e nove reais e noventa centavos. A Figura 7 mostra uma das telas do aplicativo.

¹³ Disponível em <https://www.mobills.com.br>

Figura 7: Aplicativo Mobills



Fonte: <https://www.mobills.com.br>

Na versão gratuita, ele é limitado, as funcionalidades que ele disponibiliza são o cadastro de despesas e receitas e a sincronização com os dados bancários uma vez ao dia. Porém, na versão *premium*, ele é o aplicativo mais completo no quesito de funcionalidade para gestão das finanças pessoais de todos os aplicativos analisados.

Ele conta com integração a diversas instituições financeiras, lembrete de contas a pagar, aplicativo web, cadastro ilimitado de contas bancárias e cartões de crédito, relatórios financeiros, gerenciamento de contas com um nível de customização grande, criação de objetivos e metas financeiras além de algumas funcionalidades já citadas anteriormente que serão comparadas em seguida.

2.2.5 Quadro comparativo

Para comparar as principais funcionalidades disponíveis em cada aplicativo estudado, construiu-se o Quadro 1 exposto abaixo.

Quadro 1: Comparativo entre as funcionalidades dos aplicativos

Funcionalidades/ Aplicativo	Bills Reminder	TimelyBills	Olivia AI	Mobills	YAFMA
Lembrete de contas a pagar	Sim	Sim	Sim	Sim	Não
Customização nos períodos de notificações de contas	Sim	Sim	Não	Não	Não
Classificação de contas: atrasadas, pagas e próximas	Sim	Sim	Não	Sim, na versão paga	Sim
Criação de orçamento	Não	Sim	Não	Sim	Não
Cálculo de “Posso comprar algo?”	Não	Não	Sim	Não	Sim
Criação de transações manualmente	Sim	Sim	Sim	Limitado na versão gratuita	Não
Integração com instituições financeiras	Não	Não	Sim	Sim	Não
Classificação de gastos por categorias	Sim	Sim	Sim	Limitado na versão gratuita	Sim
Criação de categorias customizadas	Sim	Sim	Não	Sim, na versão paga	Sim
Previsão de gastos futuros	Sim	Não	Sim	Não	Sim
Relatório de gastos mensais	Sim	Sim	Sim	Sim	Sim
Criação de objetivos	Não	Não	Não	Sim	Não
Integração com o Google Calendar	Não	Sim, na versão paga	Não	Não	Sim
Comunidade no App	Não	Não	Sim	Não	Não
Plataforma web	Não	Não	Não	Sim, na versão paga	Sim

Fonte: Autoria Própria

Em uma visão geral das funcionalidades dos aplicativos disponíveis no mercado podemos observar que a grande maioria não possui versão web. O Mobills

é o único que possui, porém, somente na versão paga. Outra funcionalidade que está presente em somente um aplicativo é a integração com o Google Calendar. Outra funcionalidade interessante, que existe somente no Olivia AI, é o cálculo de “posso comprar algo?”. Essa funcionalidade está diretamente relacionada com educação financeira e ao objetivo desse trabalho. Portanto foi uma das funcionalidades que foi inserida, de forma um pouco mais simplificada, após essa revisão dos aplicativos existentes.

3 REVISÃO BIBLIOGRÁFICA

Neste capítulo são apresentados os conceitos técnicos que estruturam esse trabalho.

3.1 Educação Financeira

Nos últimos anos, a população, globalmente, começou a assumir uma maior responsabilidade em relação à saúde de suas finanças pessoais. Mudanças nos sistemas de aposentadoria, principalmente nos países desenvolvidos, combinado com a perspectiva de maiores alterações nas estruturas desses sistemas, foram um dos grandes catalisadores para isso. Por consequência, aumentando a responsabilidade individual sobre decisões de poupar e investir (STOLPER, WALTER, 2017).

Na literatura acadêmica, educação financeira é um termo usado para se referir ao conhecimento de produtos financeiros. Por exemplo, a definição do que é uma ação ou um título de tesouro, conhecimento de termos financeiros, como inflação, juros compostos, diversificação de risco, *score*, conhecimento matemático necessário para realizar uma decisão financeira e engajar em certas atividades como planejamento financeiro (HASTINGS, MADRIAN, SKIMMYHORN, 2013).

Porém, o que torna a educação financeira tão importante? Segundo Hastings, Madrian e Skimmyhorn (2013), “a falta de educação financeira é problemática caso ela atrapalhe o indivíduo em melhorar o próprio bem-estar” (HASTINGS, MADRIAN, SKIMMYHORN, 2013, p. 350, tradução nossa). Um estudo feito por Lusardi e Mitchell em 2011 concluiu que pessoas com pouca educação financeira são mais suscetíveis de entrar em dívidas com altos juros, pagar taxas mais caras em serviços financeiros e ter problemas com dívidas em geral (LUSARDI, MITCHELL, 2011). Isso tem consequências negativas tanto do ponto de vida individual quanto do bem-estar da sociedade. De acordo com Hastings, Madrian e Skimmyhorn (2013, p. 350, tradução nossa), “isso também torna modelos construídos para observar o comportamento do consumidor e modelar políticas econômicas menos úteis para esses casos” (HASTINGS, MADRIAN, SKIMMYHORN, 2013).

No contexto nacional, passamos nas últimas décadas por certa ascensão econômica da população. Segundo Ferreira (2017, p. 5),

cada ano fica mais fácil o acesso de praticamente todas as pessoas a diversos tipos de transações econômicas, cada vez mais os bancos abrem um leque maior de modalidades de créditos consequentemente cada vez mais difícil fica o entendimento de quais são as condições para essa facilidade toda (FERREIRA, 2017, p. 5).

Porém, apesar disso parecer positivo, as pessoas estão encontrando novas situações financeiras negativas devido a essa nova realidade. De acordo com Ferreira (2017) as consequências negativas da falta de conhecimento básico são: “[...] esse acesso pode se tornar desastroso para a vida do indivíduo e até mesmo para a saúde financeira de um país” (FERREIRA, 2017, p. 5).

E é possível analisar os reflexos negativos na sociedade brasileira atualmente. Segundo a reportagem da CNN, o endividamento das famílias alcançou a 77,7% em abril de 2022 e a inadimplência atingiu 28,6% no mesmo período aqui no Brasil. Por fim, segundo a mesma pesquisa, 10,9% das famílias não terão condições de pagar essas dívidas (PUENTE, JANONE, 2022).

Tratando-se do nível de educação financeira, pesquisas mostram que pessoas com um maior grau de educação financeira têm maior probabilidade de planejar a sua aposentadoria e, por consequência, acumular muito mais patrimônio (KLAPPER, LUSARDI, PANOS, 2013).

Além disso, Klapper, Lusardi e Panos (2013, p. 3905, tradução nossa) observaram que “esse grupo possui maior participação em veículos financeiros formais, como bolsa de valores, maiores níveis de poupança e uma correlação negativa com o uso de empréstimos informais”. E, por fim, um alto nível de educação financeira pode ajudar as pessoas a enfrentarem crises financeiras em caso de uma grande redução na fonte de renda (KLAPPER, LUSARDI, PANOS, 2013).

Entendendo a importância da educação financeira na vida das pessoas, é necessário explorar como introduzir esse assunto. Segundo Saito (2007 *apud* ALBERTO, VERNIZZI, CLEDERSON, 2020, p. 1), a melhor forma de abordar isso é no escolar e familiar. Assim, entendo que a educação financeira, não tem como objetivo o enriquecimento e sim, a conscientização. Isso iria fazer com que o jovem

desenvolvesse hábitos para conseguir lidar com o próprio dinheiro de forma responsável e benéfica.

3.2 Regra 50-30-20

Segundo Miranda (2019), a regra 50-30-20, criada pela senadora americana Elizabeth Warren, é uma técnica de criação de orçamento. Essa regra determina que as despesas com necessidades não devem passar de 50% da renda, 30% para gastos variáveis e 20% da renda deve ser comprometida para objetivos financeiros. Portanto, para aplicar essa regra, é necessário definir o que é cada tipo de gasto (MIRANDA, 2019).

Gastos necessários e essenciais são aquelas despesas que são obrigatórias para manter a rotina funcionando. Como por exemplo, despesas com moradia, conta de luz, alimentação, transporte e saúde. Claro que, o que é essencial irá variar de acordo com a realidade de cada pessoa ou família (INFOMONEY, 2019).

A segunda categoria seria para despesas pessoais, como viagens, lazer de modo geral, como cinema e restaurantes. Essa parcela é a mais pessoal, porém, fundamental para que as pessoas organizem o orçamento mantendo seus gostos (INFOMONEY, 2019).

Por fim, tem a terceira categoria que é a parcela destinada para projetos financeiros e sonhos. Investimentos para formar uma reserva de emergência, cursos, comprar um imóvel ou até mesmo abrir o negócio próprio. Porém, aqui entraria também o pagamento de dívidas caso necessário (INFOMONEY, 2019).

Portanto, essa regra busca tentar equilibrar o orçamento da pessoa para que todas as necessidades pessoais sejam atendidas. Desde as necessidades básicas até projetos financeiros ambiciosos, mas sem abrir mão do bem-estar pessoal (INFOMONEY, 2019).

Contudo, a aplicação dessa regra não é unanimidade. No contexto do Brasil, essa aplicar essa regra pode ser um grande desafio. Pois, dependendo da situação, somente o aluguel pode ultrapassar o valor dos 50% que seriam na teoria destinados a todos os gastos relacionados a necessidade. Porém, ela foi adotada nesse trabalho pelo motivo de tentar atender todas as necessidades de um

indivíduo. Essa regra serviu como base para a funcionalidade de “Posso comprar algo?” que será explicada mais adiante.

3.3 Engenharia de Software

Engenharia de software é, de acordo com Pressman (2011, p.39), “o estabelecimento e o emprego de sólidos princípios de engenharia de modo a obter software de maneira econômica, que seja confiável e funcione de forma eficiente em máquinas reais”. Para atingir esse objetivo, a engenharia de software possui quatro camadas: foco na qualidade, processos, métodos de gerenciamento e desenvolvimento de software e, por fim, ferramentas (PRESSMAN, 2011, p. 39).

Ainda de acordo com Pressman (2011), qualquer abordagem de engenharia, incluindo a de software, ter um comprometimento com a qualidade. Pois será isso que irá dar frutos para uma cultura de aperfeiçoamento contínuo de processos e, por consequência, levará ao desenvolvimento de metodologias cada vez mais efetivas. Pressman (2011, p. 39) define que: “a pedra fundamental que sustenta a engenharia de software é o foco na qualidade” (PRESSMAN, 2011, p. 39).

Agora, tratando-se da camada de processo, de acordo com Pressman (2011, p. 40) “[...] é a liga que mantém as camadas de tecnologia coesas e possibilita o desenvolvimento de software de forma racional e dentro do prazo”. Ela é a base para o gerenciamento dos projetos, estabelecendo prazos, garantindo que seja mantida e as mudanças são administradas da melhor forma possível (PRESSMAN, 2011, p.40).

Tratando-se sobre o processo, foi utilizado para realizar o desenvolvimento do trabalho o Scrum Solo. O Scrum, de acordo com Pagotto (2016), “é um *framework* ágil para gerenciamento de projetos que se destaca por sua abordagem enxuta de desenvolvimento”. Ele divide o projeto em vários ciclos curtos, geralmente com duração de duas semanas, onde ocorre o desenvolvimento de alguma parte do software que é denominado de *sprints*. O que será desenvolvido em cada *sprint* é definido de acordo com a prioridade do *Product Owner* (proprietário do produto). Durante esse tempo, o time de desenvolvimento se dedica para desenvolver e testar uma pequena parte das funcionalidades (PAGOTTO, 2016).

O Scrum Solo é uma customização do processo Scrum voltada para o desenvolvimento individual de software. Nele, é proposto que os sprints tenham durações de uma semana e não existam reuniões diárias. E, ao final de cada sprint, deve ser entregue um protótipo do software com novas funcionalidades e, quando necessário, reuniões de orientação entre o grupo de validação (clientes e usuários finais) e o desenvolvedor (PAGOTTO, 2016).

Agora, tratando-se da camada de métodos, ela tem o objetivo de informar os detalhes técnicos necessários para criar-se o software. De acordo com Pressman (2011, p.40), essa camada

[...] envolvem uma ampla gama de tarefas, que incluem: comunicação, análise de requisitos, modelagem de projeto, construção de programa, testes e suporte. Os métodos da engenharia de software baseiam-se em um conjunto de princípios básicos que governam cada área da tecnologia e inclui atividades de modelagem e outras técnicas descritivas (PRESSMAN, 2011, p. 40).

Por fim, temos a camada de ferramentas. Elas, de acordo com Pressman (2011, p.40),

[..] fornecem suporte [...] para o processo e para os métodos. Quando as ferramentas são integradas, de modo que as informações criadas por uma ferramenta possam ser usadas por outra, é estabelecido um sistema para o suporte ao desenvolvimento de software, denominado engenharia de software com o auxílio do computador (PRESSMAN, 2011, p. 40).

3.4 UML

Para Mota (200?, p. 1), “a linguagem de modelagem unificada (UML) é uma linguagem gráfica para a visualização, especificação, construção e documentação de um software”. Ela é utilizada para demonstrar as relações, funcionamento, estados, sequências de ações etc. de um software. Para realizar isso, a UML oferece diversos tipos de diagramas gráficos para representar uma parte de um software (MOTA, 200?).

Ainda segundo Mota (200?), o desenvolvimento de uma modelagem de um sistema de software para idealizá-lo antes da sua construção é essencial, tem um papel fundamental na hora de garantir a qualidade do sistema, proporcionando um papel importante na comunicação entre as equipes do projeto. Porém, existem vários outros fatores que contribuem para o êxito do projeto, porém, a modelagem

do sistema para exemplificar para diversas pessoas como o software é um dos fatores essenciais (MOTA, 200?).

Neste projeto, foi escolhido o diagrama de Casos de Uso e o de atividades. O objetivo do diagrama de caso de uso é representar as funcionalidades que o sistema irá ter. E o de atividade é representar o fluxo de uma ação que faz parte de uma funcionalidade do sistema (FOWLER, 2005, p. 104).

3.5 Arquitetura Cliente-Servidor

A arquitetura cliente-servidor tornou-se muito popular em diferentes tipos de aplicações nos últimos tempos à medida que a sociedade passou a ser extremamente conectada com a internet. A ponto de ela estar presente na grande maioria das aplicações modernas que utilizamos na nossa rotina. Essa arquitetura de software geralmente é composta por três partes: servidor da aplicação, servidor de banco de dados e um cliente (SULYMAN, 2014).

Um servidor refere-se a um programa em execução em um computador conectado a uma rede pública ou privada, que aceita requisições de outros computadores para executar algum serviço (COULOURIS, 2013). Como, por exemplo, verificando se o cliente está autorizado a acessar determinado recurso, se a requisição é válida, se o recurso existe e, por fim, retornando uma resposta para o cliente (TANENBAUM, 2011).

Além disso, pode ser necessário armazenar ou modificar alguma informação relacionada ao pedido do cliente para ser consumida em algum outro momento. Para isso, o servidor irá realizar a comunicação com um banco de dados que será responsável por guardar todas as informações da aplicação (TANENBAUM, 2011).

Ou seja, essa parte é responsável por todas as funcionalidades que o usuário não interage diretamente. Em uma aplicação web, é comum esse lado ser denominado de *back-end* (KENZIEACADEMY, 2020).

O cliente, também chamado de *front-end*, no contexto de uma aplicação web, é o dispositivo que o usuário utiliza para acessar uma página web na rede. Nele, o foco é na criação dos elementos visuais que serão visualizados diretamente pelo usuário (KENZIEACADEMY, 2020). Os usuários acessam a aplicação a partir de um navegador que é o responsável por exibir o conteúdo da página para o usuário. Mas

o processamento desse conteúdo pode ser feito tanto no computador do cliente, quanto no servidor (TANENBAUM, 2011).

Para realizar essa comunicação entre cliente e servidor, existem alguns protocolos padrões. Alguns exemplos deles são: *File Transfer Protocol* para transferência de arquivos, *Simple Mail Transfer Protocol* para o envio de emails e *Hypertext Transfer Protocol* para enviar de mensagens de texto (SULYMAN, 2014).

Existem três grandes vantagens para essa arquitetura que tornou ela dominante no desenvolvimento de sistemas. A primeira vantagem é que ela permite a divisão de processamento da aplicação em diversas máquinas. A segunda é permitir o compartilhamento de recursos entre cliente e o servidor de maneira mais fácil. Por fim, também reduz a replicação de dados armazenados, armazenando-os somente no servidor (SULYMAN, 2014).

Além disso, a arquitetura cliente servidor pode ser implementado de duas formas: duas camadas ou três camadas. A arquitetura com duas camadas é composta somente pelo cliente e um banco de dados. Os clientes irão rodar a aplicação localmente e conectar diretamente com o banco de dados sem nenhum intermediário (SULYMAN, 2014).

Em contrapartida, a arquitetura com três camadas o cliente se comunica diretamente com o servidor ao invés de acessar o banco de dados diretamente. Assim, o cliente fica responsável somente pela lógica de apresentação do conteúdo e o servidor irá atuar como um intermediário entre o cliente e o banco de dados e executando a lógica de negócio (SULYMAN, 2014).

A arquitetura cliente-servidor, portanto, pode ser definida como uma arquitetura de *software* composta por um cliente e um servidor. Onde o cliente é responsável por realizar os pedidos e o servidor responder a eles (SULYMAN, 2014).

4 METODOLOGIA

A pesquisa realizada foi de caráter qualitativo e aplicada. Segundo Silveira e Gerhardt (2009, p. 32), “as características da pesquisa qualitativa são: objetivação do fenômeno; hierarquização das ações de descrever, compreender, explicar, precisão das relações entre o global e o local em determinado fenômeno”. E em relação a uma pesquisa aplicada, ainda de acordo com Silveira e Gerhardt (2009), ela tem o objetivo de “gerar conhecimentos para aplicação prática, dirigidos à solução de problemas específicos” (GERHARDT, SILVEIRA. 2009).

A análise de requisitos foi feita com base no problema central: planejamento financeiro. Primeiramente, foram identificadas quais as funcionalidades chaves para um sistema que tem o objetivo de auxiliar o planejamento financeiro. Além disso, também foi feito o estado da arte realizada em cima das aplicações web e *mobile* já existentes no mercado. Isso gerou como resultado a adição de algumas funcionalidades importantes dentro do sistema, como, por exemplo, a integração com o Google Calendar que não estava previsto no planejamento inicial.

Após isso, iniciou-se o desenvolvimento do sistema. A metodologia de desenvolvimento foi realizada em um Scrum solo com algumas alterações. Devido ao fato de não haver um contato direto com um cliente durante o desenvolvimento, não era possível realizar uma validação da entrega com o usuário final. Outra modificação feita foi remover o tempo de duração fixo de cada *sprint*. Logo, cada sprint era composta por uma funcionalidade do sistema sem uma data final determinada.

Em paralelo ao desenvolvimento, ocorreu a realização dos testes de integração à medida que as funcionalidades do sistema atingiam um nível de maturidade e estabilidade aceitável.

E a partir dos requisitos, com o auxílio da UML, foram criados os diagramas de casos de uso para a visualização dos requisitos funcionais do sistema. E também o diagrama de atividades para demonstrar como o sistema deve se comportar na atividade representada.

O sistema foi desenvolvido na linguagem Javascript¹⁴ no *front-end* com React¹⁵. A estruturação básica dos elementos na página foi com HTML¹⁶ e a

¹⁴ <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript>

estilização foi feita com CSS¹⁷ em conjunto com Chakra UI¹⁸, uma biblioteca de componentes React para acelerar a criação da interface visual.

No *back-end*, foi utilizado o Hasura¹⁹. Ele é uma ferramenta capaz de criar APIs GraphQL²⁰ instantaneamente a partir de um banco de dados PostgreSQL²¹. Em conjunto com o servidor Hasura, foi criado um servidor Node.js²², que também utiliza Javascript, em conjunto com o *framework* Express.js²³. Ele servirá para a criação de *Web Hooks* que são consumidos pelo Hasura para realizar algumas regras de negócio que não são possíveis somente com o Hasura.

Em relação aos testes, foram realizados os seguintes: unitários e *end-to-end* e os testes de validação com o usuário final. Os testes unitários foram realizados tanto no *front-end* quanto no *back-end*, com a ferramenta Jest²⁴, para garantir a qualidade do código e facilitar futuras melhorias do sistema. Em relação aos testes *end-to-end*, que servirão para garantir que a integração do sistema está funcionando, para isso, foi utilizado o *framework open-source* Cypress²⁵.

Por fim, para realizar os testes com o usuário final, foi distribuída uma versão de testes para alguns usuários que se voluntariam para testar a aplicação. Em conjunto ao sistema, foi entregue um formulário para recolher feedbacks dos usuários para analisar se o objetivo do sistema foi atingido.

¹⁵ <https://pt-br.reactjs.org/>

¹⁶ <https://developer.mozilla.org/pt-BR/docs/Web/HTML>

¹⁷ <https://developer.mozilla.org/pt-BR/docs/Web/CSS>

¹⁸ <https://chakra-ui.com/>

¹⁹ <https://hasura.io/>

²⁰ <https://graphql.org/>

²¹ <https://www.postgresql.org/>

²² <https://nodejs.org/en/>

²³ <https://expressjs.com/pt-br/>

²⁴ <https://jestjs.io/pt-BR/>

²⁵ <https://www.cypress.io/>

5 DESENVOLVIMENTO

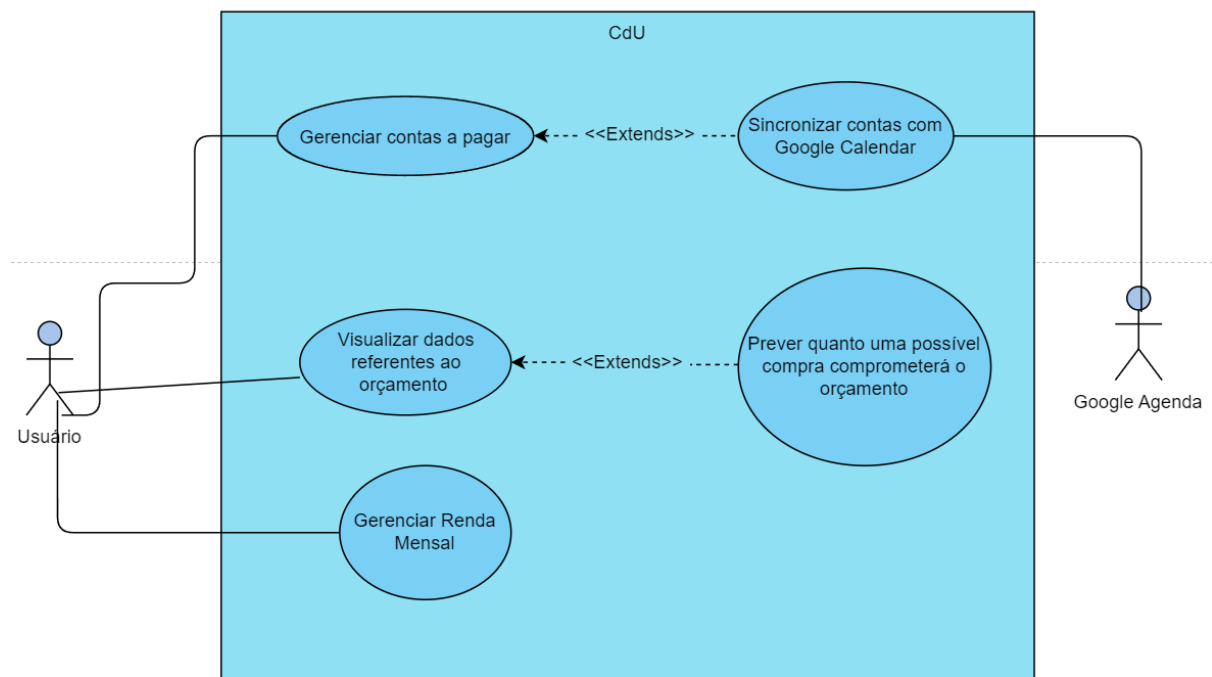
Este capítulo detalha o desenvolvimento da aplicação desde a sua modelagem até os testes.

5.1 Modelagem

A partir da análise de requisitos, foi desenvolvido o diagrama de caso de uso para descrever as funcionalidades existentes no sistema.

Como mostra a Figura 8, o sistema possui somente um ator que é o usuário final. Todas as funcionalidades do *software* serão voltadas para auxiliá-lo a planejar as suas finanças pessoais.

Figura 8: Diagrama de caso de uso



Fonte: Autoria Própria

O Caso de Uso (CdU) **gerenciar contas a pagar** engloba as funcionalidades referentes às contas a pagar do usuário, como por exemplo: criar, editar, excluir contas e visualizar informações sobre o seu vencimento, pagamento, etc. Esse CdU possui um fluxo adicional opcional que é **sincronizar contas com Google Calendar**. Essa extensão permite que o usuário vincule a data de vencimento das

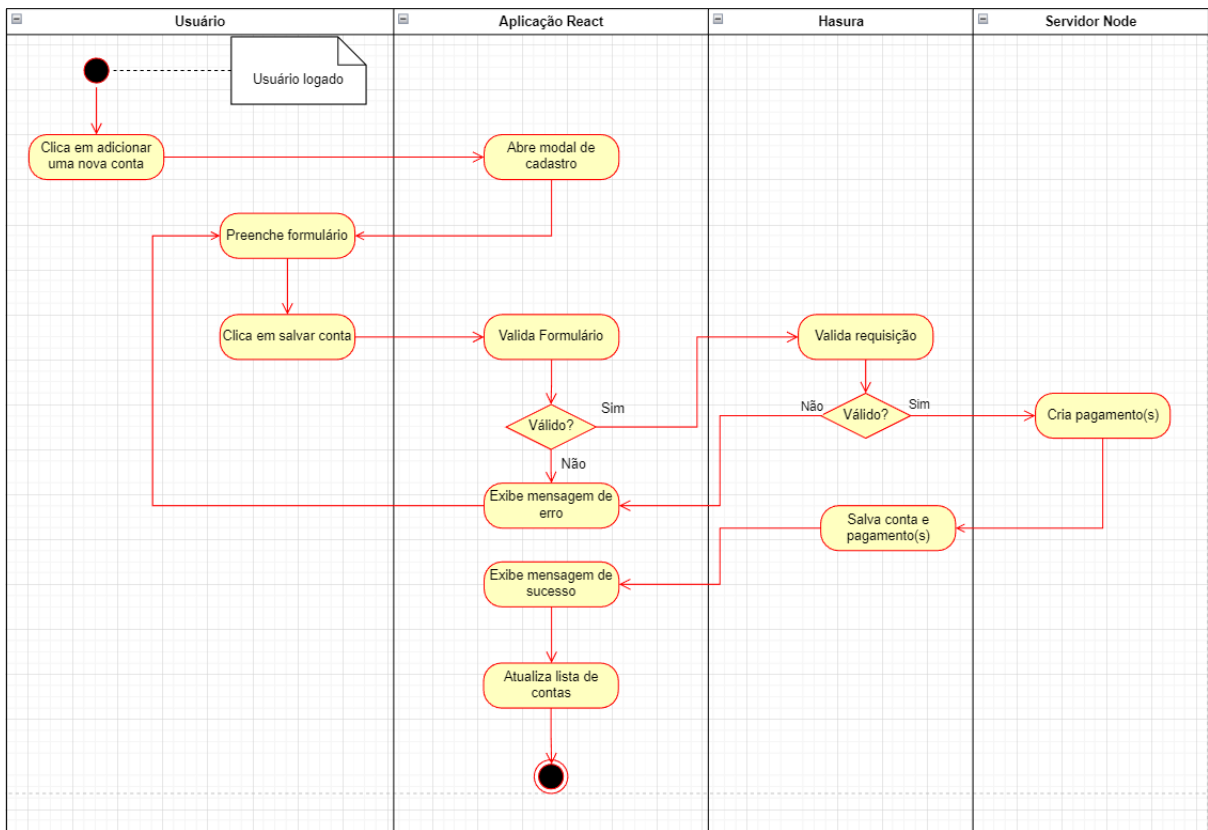
contas com o aplicativo Google Calendar possibilitando que receba lembretes em seu e-mail e celular.

O CdU **visualizar dados referentes ao orçamento** trata dos gráficos referentes aos gastos cadastrados na plataforma. O usuário terá a possibilidade de visualizar gastos passados e futuros, comparar renda e despesas e verificar a categoria dos seus gastos, o que o auxiliará a tomar decisões sobre sua saúde financeira. Esse CdU possui o fluxo adicional **prever orçamento comprometido** onde o usuário poderá verificar o quanto um possível gasto compromete o seu orçamento e ainda se informar se é recomendado ou não fazer essa compra de acordo com a regra 50-30-20 explicada anteriormente.

Por fim, o CdU **gerenciar renda mensal** reúne funções relacionadas à renda do usuário. Ele poderá cadastrar e editar o salário e adicionar eventuais bônus extras que receber no mês. A especificação de todos os casos de uso está no apêndice.

Além do CdU, foi criado o diagrama de atividade para demonstrar como é o fluxo para a criação de uma conta no sistema. Cada retângulo de bordas arredondadas representa uma ação realizada por um participante do sistema. A Figura 9 demonstra o diagrama de atividades referente à ação de criar conta no sistema.

Figura 9: Diagrama de Atividade da criação de conta

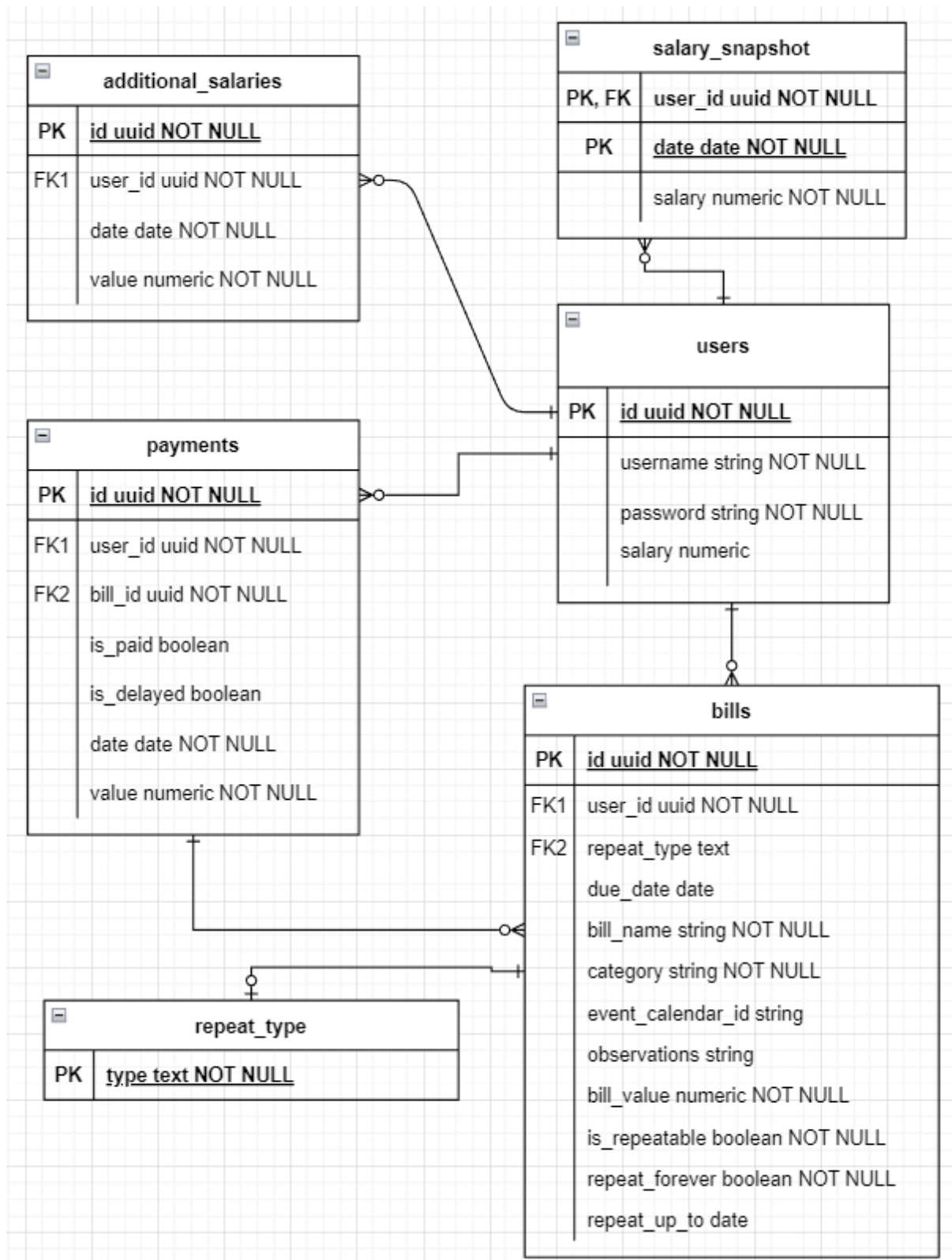


Fonte: Autoria própria

Para ser possível armazenar os dados do sistema de forma eficiente foi criado um diagrama para representar a estrutura do banco de dados. Ele é denominado de modelo Entidade-Relacionamento e tem como objetivo representar como os dados se relacionam dentro do sistema (DEVMEDIA, 2014). Para realizar a criação deste diagrama - Figura 10 - foi utilizado a ferramenta draw.io²⁶ com a notação "pé de galinha" (LUCIDCHART, 2018).

²⁶ <https://draw.io>

Figura 10: Diagrama Entidade-Relacionamento



Fonte: Autoria própria

O sistema possui seis tabelas, sendo a tabela *bills*, *payments* e *users* a base de todas as funcionalidades do sistema. A tabela *users* tem o objetivo de armazenar as informações de cadastro do usuário e o seu salário caso tenha sido cadastrado. Relacionadas a essa tabela, temos a tabela de *additional_salaries*, *salary_snapshot*, *payments* e *bills*.

As tabelas *additional_salaries* e *salary_snapshot* são relacionadas a funcionalidade de renda do usuário. Sendo, respectivamente, para armazenar o dinheiro extra que o usuário pode vir a receber e para persistir o histórico de salário dos usuários considerando a renda extra no cálculo.

A tabela *bills* tem o objetivo de guardar as informações de uma conta cadastrada pelo usuário. Aqui terá todas as informações para classificar essa conta, como data de vencimento, valor, categoria, nome, etc. Além disso, ela está relacionada com a tabela *repeat_type* caso ela seja uma conta que se repita. Essa tabela possui três valores somente: “WEEKLY” para as contas que se repetem semanalmente, “MONTHLY” para mensais e “YEARLY” para contas anuais.

Um detalhe importante que gerou a criação dessa tabela ao invés de deixar como um campo comum na tabela de *bills* foi relacionado a uma funcionalidade do Hasura. Ao criar uma tabela para representar valores *enum*, isso irá permitir que o ele tenha conhecimento dos únicos valores possíveis para aquele determinado campo na própria requisição GraphQL. Por consequência, isso gera uma camada adicional de segurança.

Por fim, temos a tabela *payments*. Cada registro desta tabela representa um pagamento que é fornecido somente pelo servidor da aplicação. Nela são armazenadas as informações relacionadas aos pagamentos de uma conta, como se a conta está paga ou atrasada. Além disso, temos uma coluna de identificação do usuário mesmo já tendo a mesma informação na entidade *bill*.

O motivo dessa escolha foi devido a uma particularidade do GraphQL relacionada à forma de como é feita a atualização e inserção de dados. Caso o banco estivesse completamente normalizado, isso iria aumentar um pouco a complexidade do código para realizar atualizações relacionadas a pagamentos. Portanto, nessa situação, era melhor sacrificar o armazenamento do banco de dados em troca de menor complexidade do código no *front-end* e *back-end*.

5.2 Tecnologia

Depois de modelado, a aplicação foi desenvolvida com as tecnologias e ferramentas descritas neste subcapítulo.

Tanto para o *front-end* quanto *back-end* foi utilizado o Javascript que é uma linguagem de programação dinâmica, leve e interpretada (MOZILLA, 2022). Criada em 1995 pela Netscape, ela originalmente foi construída para ser uma linguagem do lado do cliente e ser executada no browser do usuário. Porém, atualmente ela pode ser executada no servidor ou em qualquer dispositivo que tenha a *engine* do Javascript instalada (TUTORIALSPPOINT, 2016).

A sua aplicação mais comum é no browser, onde ele é utilizado para tornar as páginas web mais dinâmicas. Como ele tornou-se possível executar diversas ações, como mudar o conteúdo da página, realizar requisições HTTP, reagir a interações do usuário, como eventos de cliques e validar dados antes de enviá-los para um servidor (TUTORIALSPPOINT, 2016).

Um dos principais motivos para o Javascript ter sido a linguagem principal dessa aplicação foi o fato da possibilidade de utilizar a mesma linguagem tanto no lado do cliente quanto no lado do servidor. Para utilizar Javascript no *back-end* precisamos de um ambiente capaz de executar Javascript no lado do servidor, para isso, foi escolhido o Node.js. Enquanto no lado do cliente foi utilizado a biblioteca React.

Começando pelo servidor, Node.js é um ambiente de execução de Javascript baseado em eventos no lado do servidor. Diferente de outros ambientes, um processo Node não depende de *multithreading* para suportar execução paralela de lógica de negócio (TILKOV, VINOSKI, 2010). Por esse motivo, com Node.js não existe a necessidade de se preocupar com *deadlocks* em processos já que não existem *locks* (NODEJS, 2011).

Já no *front-end* foi utilizado React. De acordo com Gackenheimer (2015, p. 1) “Ela foi criada pelo time de engenheiros do Facebook para resolver problemas que envolviam criar interfaces de usuários complexas”. O objetivo principal do React é minimizar possíveis os erros que podem ocorrer no desenvolvimento (MOZILLA, 2021a).

Um dos motivos pela escolha do React como tecnologia de *front-end* é pela possibilidade de criar *single page applications* (SPA). Uma aplicação SPA é uma

aplicação web que tem a característica de carregar os dados uma única vez do servidor e o conteúdo dinamicamente através de APIs do Javascript (MOZILLA, 2021b). Isso proporciona uma melhor experiência já que a sua navegação não é interrompida em nenhum momento, fazendo com que o site se comporte de forma similar a uma aplicação desktop (DESHMUKH, MANE, RETAWADE, 2019).

Como no *front-end* basicamente a linguagem predominante no mercado é Javascript, não havia nenhum motivo para utilizar outra linguagem. E a escolha do React foi devido ser a ferramenta com maior adoção no mercado e ser apenas uma biblioteca, sendo extremamente leve e enxuto permitindo instalar somente pacotes externos realmente necessários para o projeto.

Porém, no *back-end* existem mais opções, mas devido ao fato do autor do trabalho ter mais familiaridade com Javascript, e Node.js ser o *framework* mais utilizado no universo Javascript, ele é a ferramenta que mais fazia sentido para esse projeto.

Contudo, para complementar o desenvolvimento com Javascript, também foi utilizado o Typescript. O Typescript é uma extensão do Javascript com o objetivo de prover uma maior facilidade no desenvolvimento de aplicações de Javascript de grande escala. Ele oferece um sistema de módulos, interfaces, classes e, principalmente, um sistema de tipagem gradual. A intenção é que o Typescript proporcione uma transição suave para os programadores de Javascript sem a necessidade de refatorações grandes (BIERMAN, ABADI, TORGERSEN, 2014).

A escolha de utilizar Typescript no projeto foi principalmente para adicionar o suporte a tipagem ao Javascript devido ao fato que Javascript não possui suporte a tipagem estática nativamente. Isso evita inúmeros *bugs* relacionados a tipo de variáveis ou retorno de uma função inesperada.

Seguindo com as ferramentas utilizadas no projeto, na parte de armazenamento de dados foi utilizado o banco de dados relacional PostgreSQL. O PostgreSQL é um SGBD relacional, multiplataforma e *open-source*. Ele é de tecnologia extremamente robusta e escalável, sua arquitetura proporciona alto desempenho mesmo com hardware limitado e grandes volumes de dados (POSTGRESQL, 2022). Um grande fator que pesou na hora da escolha do sistema de banco foi a ferramenta escolhida para APIs para o sistema, o Hasura. No momento atual, o único banco suportado pelo Hasura é o PostgreSQL.

No servidor do banco de dados, foram criados alguns *cronjobs*, que são gatilhos agendados para executar lógica de negócio em períodos específicos do tempo via chamadas HTTP ao servidor (HASURA, 2020a). Como por exemplo, ao final do mês, salvar o histórico do salário do usuário, verificar se há alguma conta atrasada, criar novos registros de futuros pagamentos caso seja um pagamento recorrente.

É importante ressaltar que os *cronjobs* não foram utilizados para implementar notificações de vencimentos de conta. Devido a complexidade de implementar e gerenciar notificações *push* em uma aplicação, o Google Calendar ficou responsável por gerenciar os lembretes de contas a pagar caso o usuário deseje vincular uma conta ao seu calendário.

O Hasura GraphQL engine é um projeto *open-source*, construído em Haskell, que fornece APIs GraphQL em uma base de dados PostgreSQL. Ele conta com diversas funcionalidades prontas, como *queries* com paginação, filtros, busca por padrões, *triggers* para a execução de *webhooks* ou funções *serverless* para acelerar a construção de uma aplicação (HASURA, 2020b).

Ele foi construído com o objetivo de executar em qualquer lugar a partir de uma máquina local ou um servidor com recursos bem limitados, a um servidor na nuvem como AWS (Amazon Web Services), Google ou Microsoft. Conseqüentemente, ele foi otimizado para consumir pouca memória, suportando até 1000 requisições por segundo consumindo aproximadamente 50 MB de memória RAM (HASURA, 2020b).

GraphQL é uma linguagem para realizar consultas em bancos de dados a partir do lado do cliente da aplicação. Criada pelo Facebook em 2012, ela foi desenvolvida com o objetivo de reduzir a quantidade de pacote de dados utilizados necessário para fazer consultas nas suas aplicações mobile (STEMMELER, 2021).

O GraphQL é uma alternativa à APIs *Representational State Transfer* (REST). E uma das características principais para isso é que ele possibilita ao cliente determinar quais os dados que ele irá consumir, reduzindo a quantidade de dados desnecessários consumida (STEMMELER, 2021).

A escolha do Hasura e GraphQL foi devido a sua característica de fornecer básicas APIs GraphQL de criação, atualização, leitura e deleção sem a necessidade de nenhum tipo de configuração. Isso, combinado com a flexibilidade do GraphQL,

possibilitou o desenvolvimento das diversas funcionalidades de forma extremamente ágil.

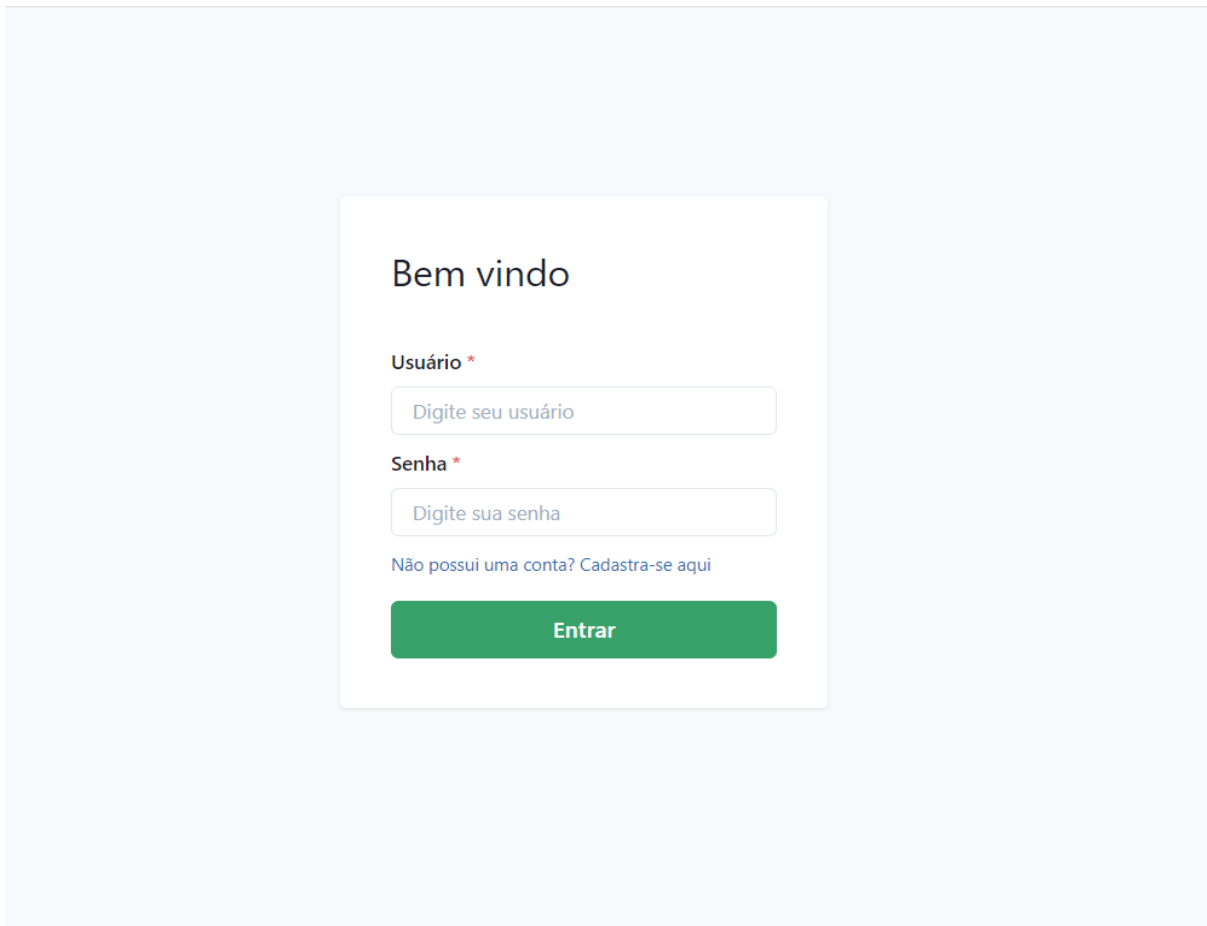
Em relação à hospedagem, foi utilizado o serviço gratuito do Heroku. O *front-end*, o servidor *back-end*, o banco de dados em conjunto com o servidor do Hasura estão todos nesse serviço. Um fator que pesou na escolha dessa ferramenta foi devido a integração com o Github para atualizar o ambiente com novas atualizações sempre que houver mudanças nos repositórios. Não havendo a necessidade de controlar atualizar manualmente os ambientes.

5.3 Aplicação

Este subcapítulo detalha a aplicação denominada YAFMA – aplicação web para auxiliar no planejamento financeiro pessoal.

A Figura 11 mostra a tela inicial do sistema, caso o usuário não esteja autenticado, é a tela de login. Nela o usuário poderá autenticar-se no sistema caso ele já possua uma conta. Além disso, também há um link para criar uma conta caso o usuário não tenha uma conta.

Figura 11: Tela inicial do sistema



Bem vindo

Usuário *

Senha *

Não possui uma conta? Cadastre-se aqui

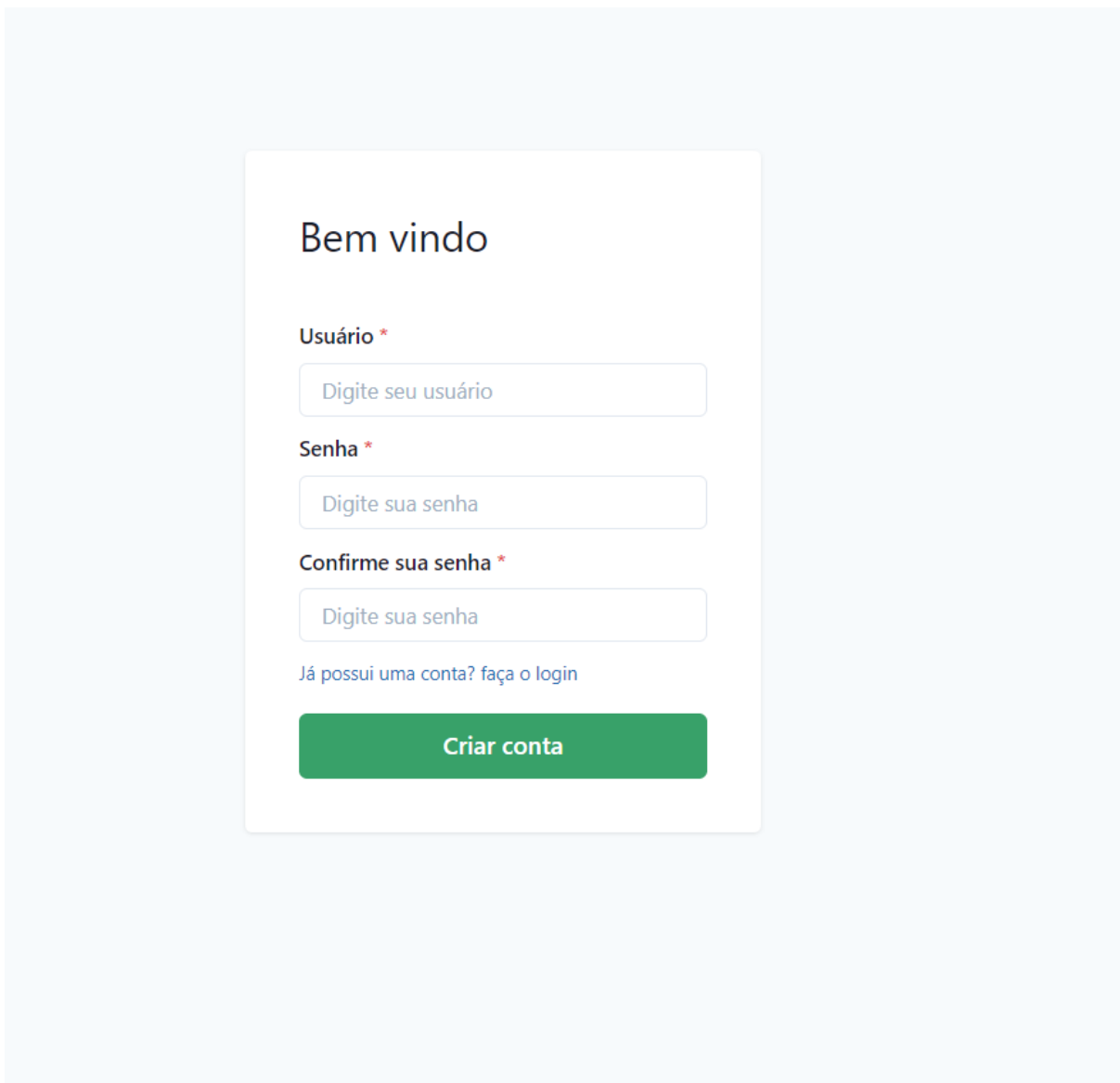
Entrar

Fonte: Autoria própria

Caso o usuário clique no link para se cadastrar, a aplicação irá para a página de criação de conta, como mostra a Figura 12. A criação de conta é realizada de forma básica, necessitando somente de um nome de usuário e senha. Após o usuário preencher os campos corretamente, o sistema irá validar se não existe um usuário com o mesmo nome e se as senhas são iguais, cadastrando-o no sistema. A senha do usuário é guardada no banco de dados de forma criptografada e o servidor irá retornar um *JSON Web Token (JWT)*²⁷ para o cliente que será utilizado para garantir que o usuário esteja autenticado em todas as requisições dentro do sistema.

²⁷ Um token JWT é uma técnica para garantir a transmissão de dados entre duas partes de forma segura (<https://jwt.io/introduction>).

Figura 12: Tela de cadastro do usuário



Bem vindo

Usuário *

Senha *

Confirme sua senha *

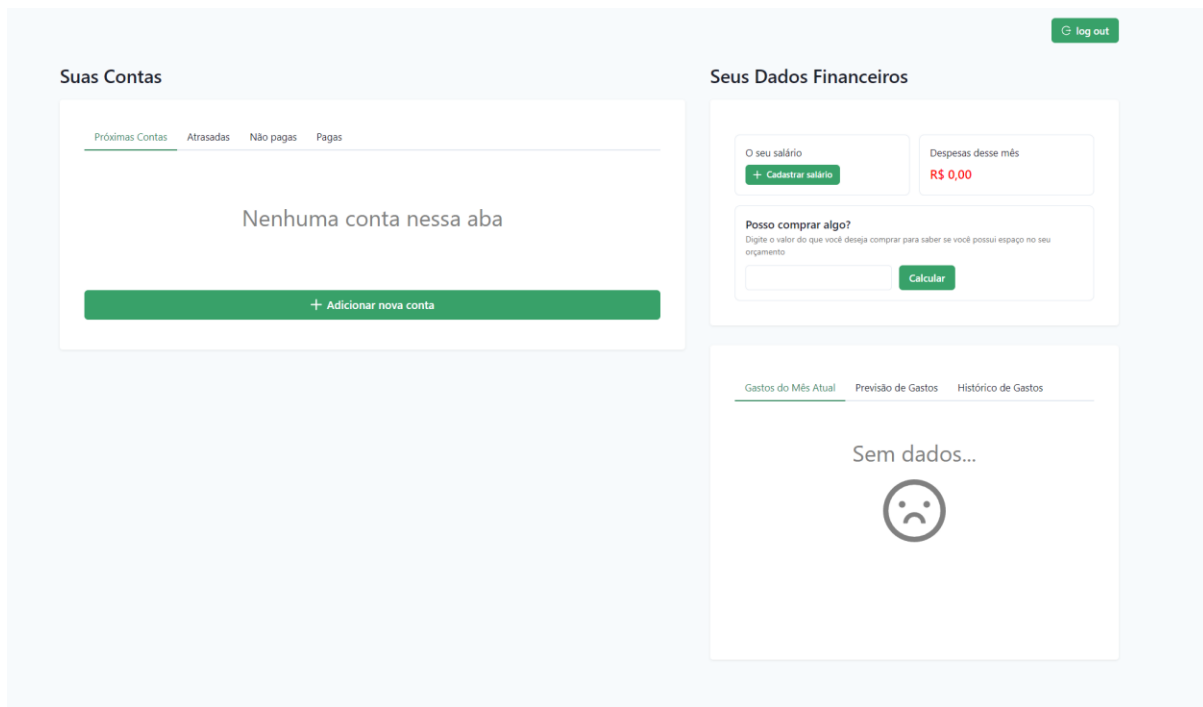
Já possui uma conta? faça o login

Criar conta

Fonte: Autoria própria

Após o usuário se autenticar, ele será redirecionado a tela inicial do sistema, como mostra a Figura 13. Esse painel inicial contém todas as funcionalidades do sistema. O sistema pode ser dividido em duas partes: contas e dados financeiros. Nas contas é centralizado todas as informações das contas do usuário. Nos dados financeiros são reunidas as informações referentes às finanças, como salário e despesas.

Figura 13: Tela principal do sistema



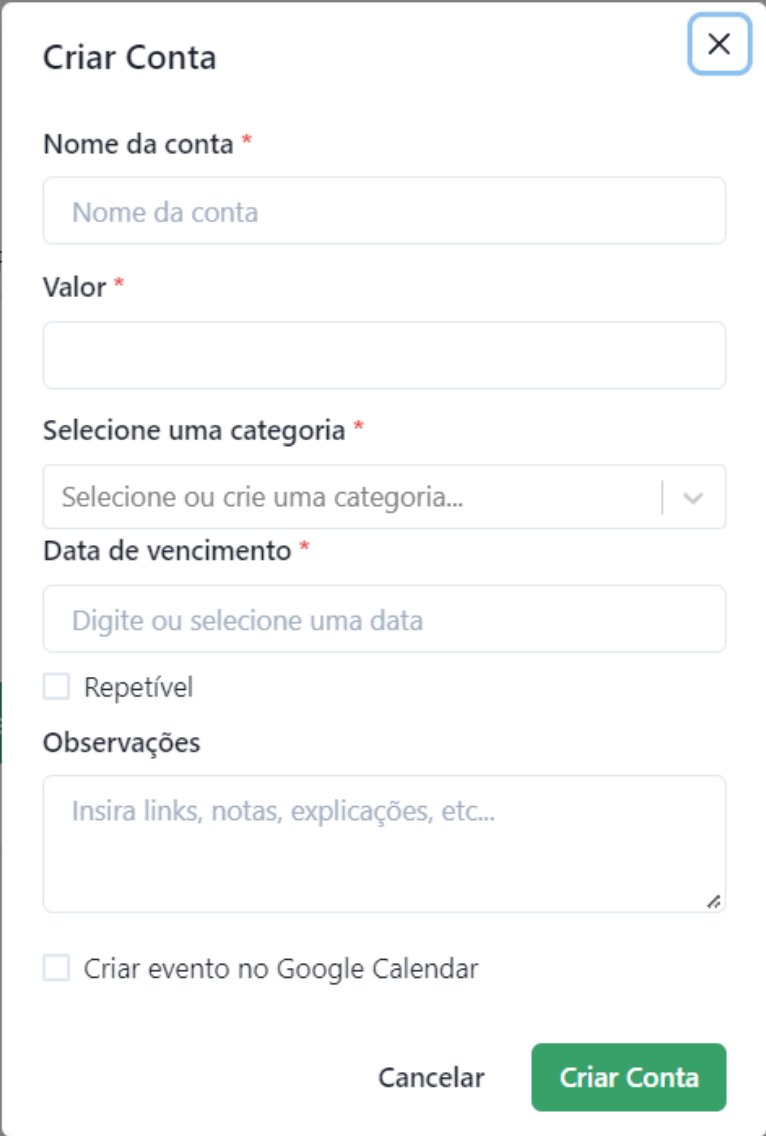
Fonte: Autoria própria

O núcleo do sistema gira em torno das contas. Então para isso é necessário criar uma conta. Para fazer isso, basta clicar no botão “Adicionar nova conta”.

Ao clicar nesse botão, a Figura 14 e 15 ilustra a modal²⁸ que irá abrir que contém um formulário para criar uma nova conta. Além de informações básicas como nome e valor, o sistema permite que o usuário selecione ou crie uma categoria para essa conta. Para isso, basta digitar o nome de uma categoria ao invés de selecionar uma pré-existente no campo de categoria. Além disso, em relação ao vencimento, existem dois tipos, as contas que se repetem em um determinado período e as de único pagamento.

²⁸ Na informática, modal é uma janela que abre sobre o conteúdo da página sem removê-lo. (<https://www.homehost.com.br/blog/tutoriais/modal-bootstrap/>)

Figura 14: Modal de criação de conta



Modal de criação de conta com os seguintes campos e opções:

- Criar Conta** (título do modal)
- Nome da conta *** (campo de texto)
- Valor *** (campo de texto)
- Selecione uma categoria *** (menu suspenso)
- Data de vencimento *** (campo de texto)
- Repetível
- Observações** (campo de texto)
- Criar evento no Google Calendar
- Botões: **Cancelar** e **Criar Conta**

Fonte: Autoria própria

Figura 15: Opção de repetições de conta

The image shows a 'Criar Conta' (Create Account) modal form. At the top left of the modal is the title 'Criar Conta' and a close button 'X'. The form contains several fields and options:

- Nome da conta ***: A text input field with the placeholder 'Nome da conta'.
- Valor ***: A text input field.
- Selecione uma categoria ***: A dropdown menu with the placeholder 'Selecione ou crie uma categoria...' and a downward arrow.
- Data de vencimento ***: A text input field with the placeholder 'Digite ou selecione uma data'.
- Repetível**: A checked checkbox.
- Qual a recorrência da conta? ***: Three radio button options: 'Semanal', 'Mensal', and 'Anual'.
- Repete até o dia ***: A text input field with the placeholder 'Digite ou selecione uma data'.
- Sem prazo final**: An unchecked checkbox.
- Observações**: A text area with the placeholder 'Insira links, notas, explicações, etc...'.
- Criar evento no Google Calendar**: An unchecked checkbox.

At the bottom of the modal, there are two buttons: 'Cancelar' and 'Criar Conta' (highlighted in green). In the background, a dashboard is visible with a 'nova conta' button, a balance of 'R\$ 1.000', and a chart showing expenses.

Fonte: Autoria própria

Tratando-se das contas que se repetem, existem basicamente tipos: semanal, mensal e anual. Além disso, é possível que tenha uma data final para o último pagamento ou ela se repita por um tempo indeterminado.

Por fim, uma das principais funcionalidades do sistema é a capacidade de vincular os vencimentos das contas pela ferramenta de calendário da Google, o Google Calendar. Para isso, é necessário ter uma conta da Google e se autenticar na janela que abre após clicar no botão que irá aparecer ao marcar a opção de “Criar evento no Google Calendar”, como mostra a Figura 16.

Figura 16: Opção de vincular conta com Google Calendar

Criar Conta ×

Nome da conta *

Nome da conta

Valor *

Selecione uma categoria *

Selecione ou crie uma categoria... | ▾

Data de vencimento *

Digite ou selecione uma data

Repetível

Observações

Insira links, notas, explicações, etc...

Criar evento no Google Calendar

É necessário logar no Google Calendar

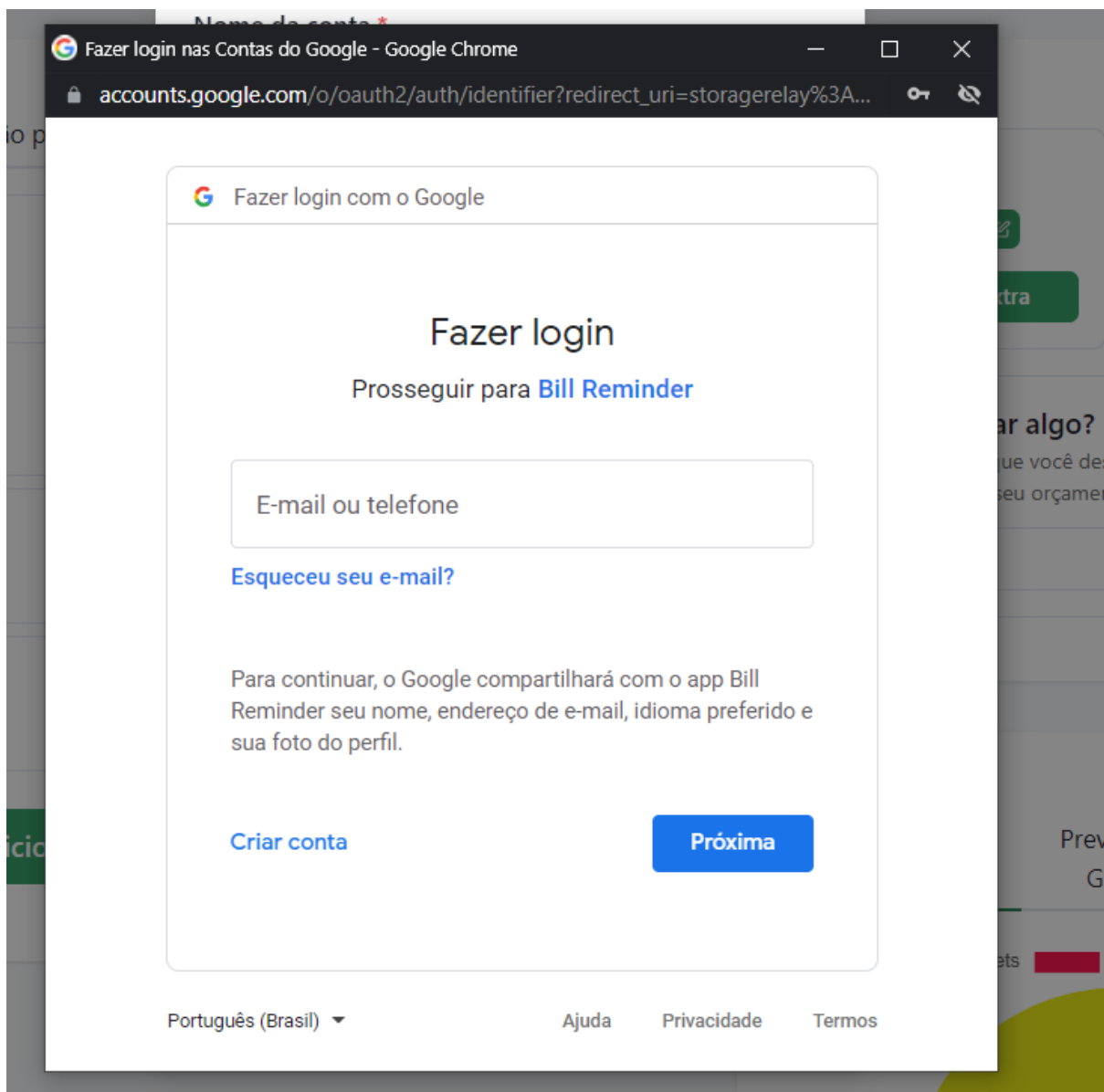
Sign-in Google Calendar

Cancelar **Criar Conta**

Fonte: Autoria própria

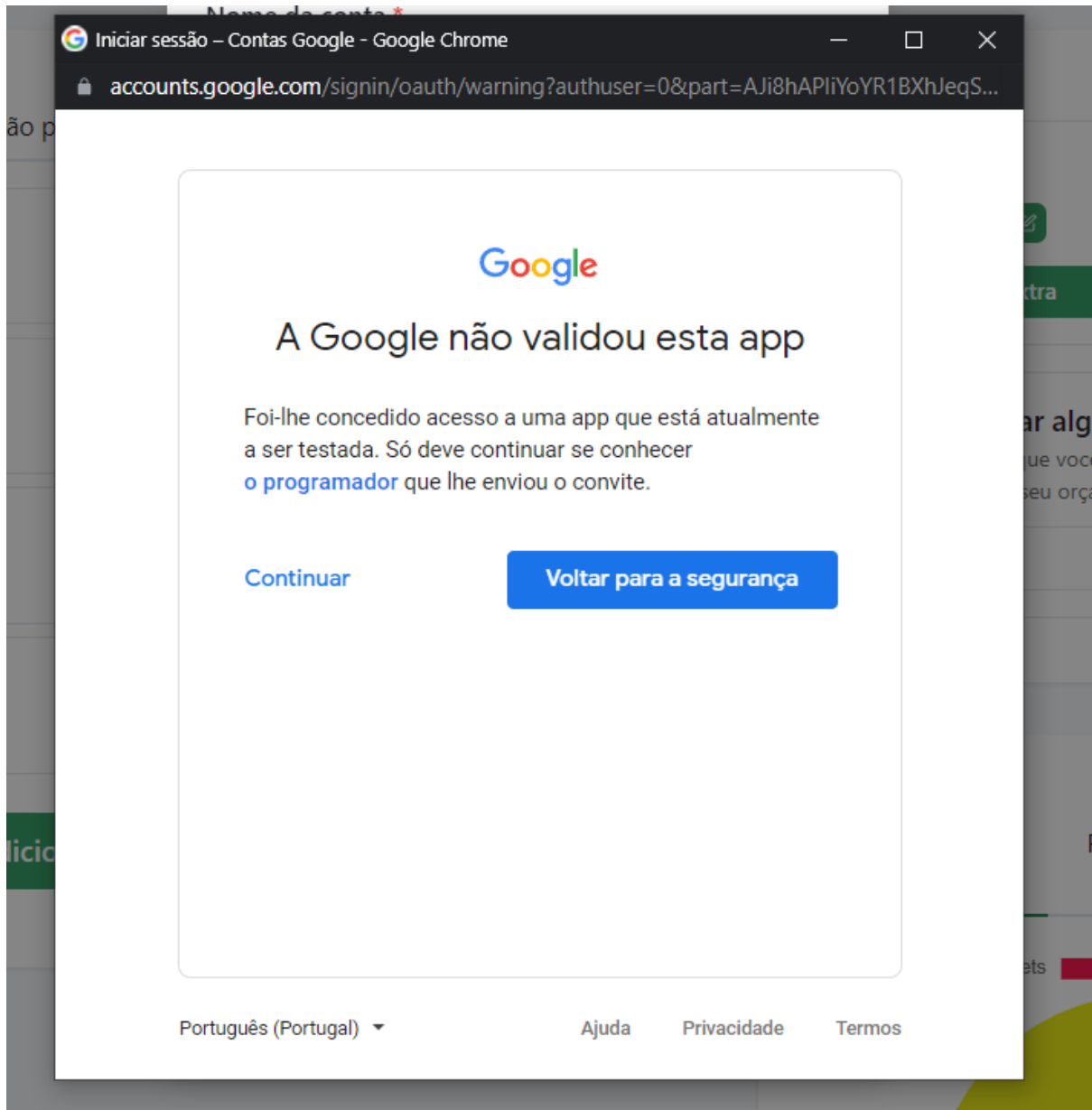
Após clicar nesse botão é aberto a janela para o usuário preencher com a suas credenciais da sua conta Google, como mostra a figura 17. Após o *login* com a sua conta da Google, irá aparecer um aviso enviado pela Google informando que o aplicativo não foi validado por ela, como mostra a Figura 18. Isso é uma medida de segurança para proteger os usuários de aplicativos maliciosos. Basta clicar em “continuar” para prosseguir com a vinculação da conta com o Google Calendar.

Figura 17: Janela de login Google



Fonte: Autoria própria

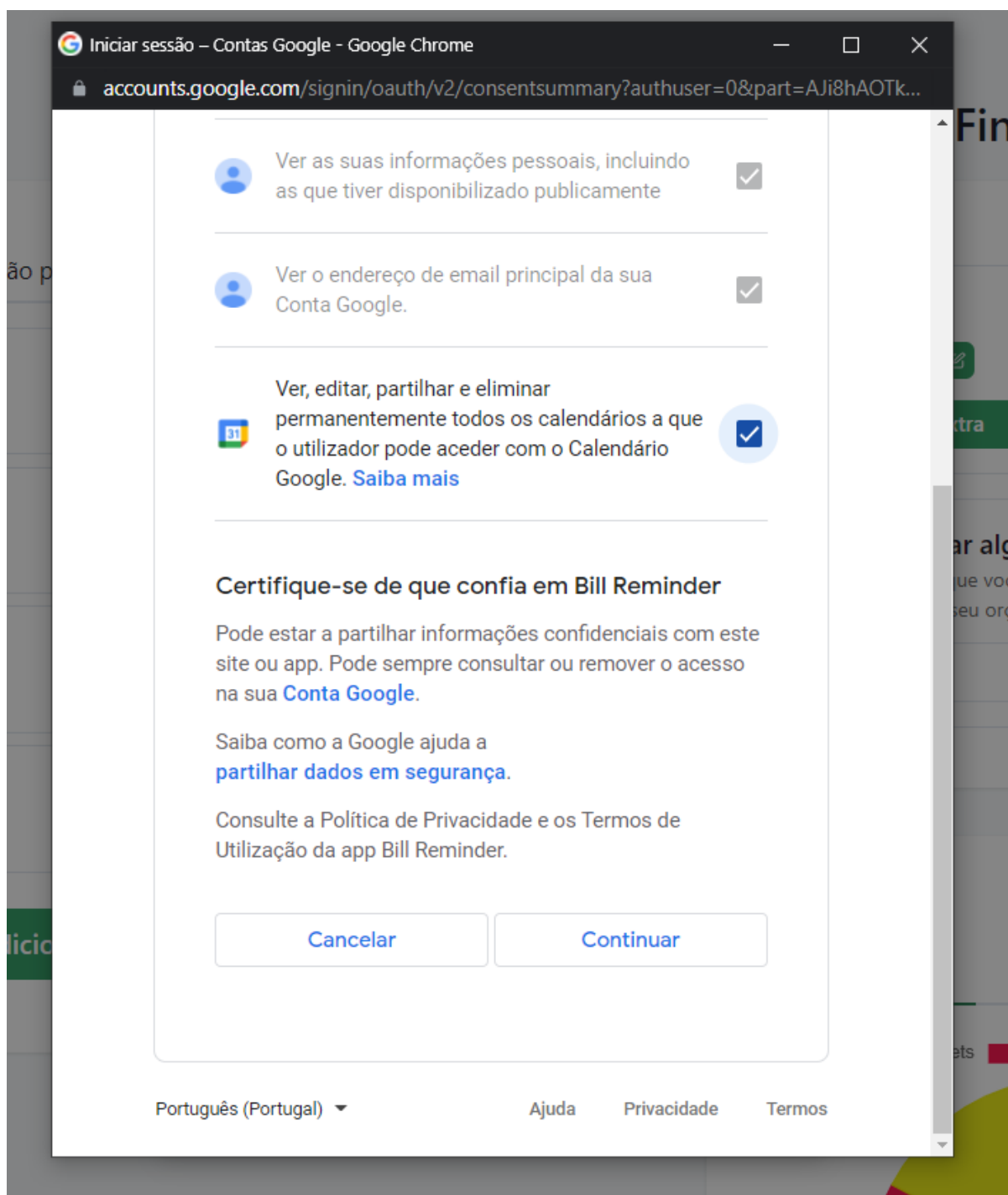
Figura 18: Tela de aviso de aplicativo não validado



Fonte: Autoria própria

Por fim, é necessário que o usuário dê permissão para o sistema editar o seu calendário e prosseguir com a criação da conta no sistema, como é ilustrado na figura 19.

Figura 19: Tela de fornecer permissão ao calendário

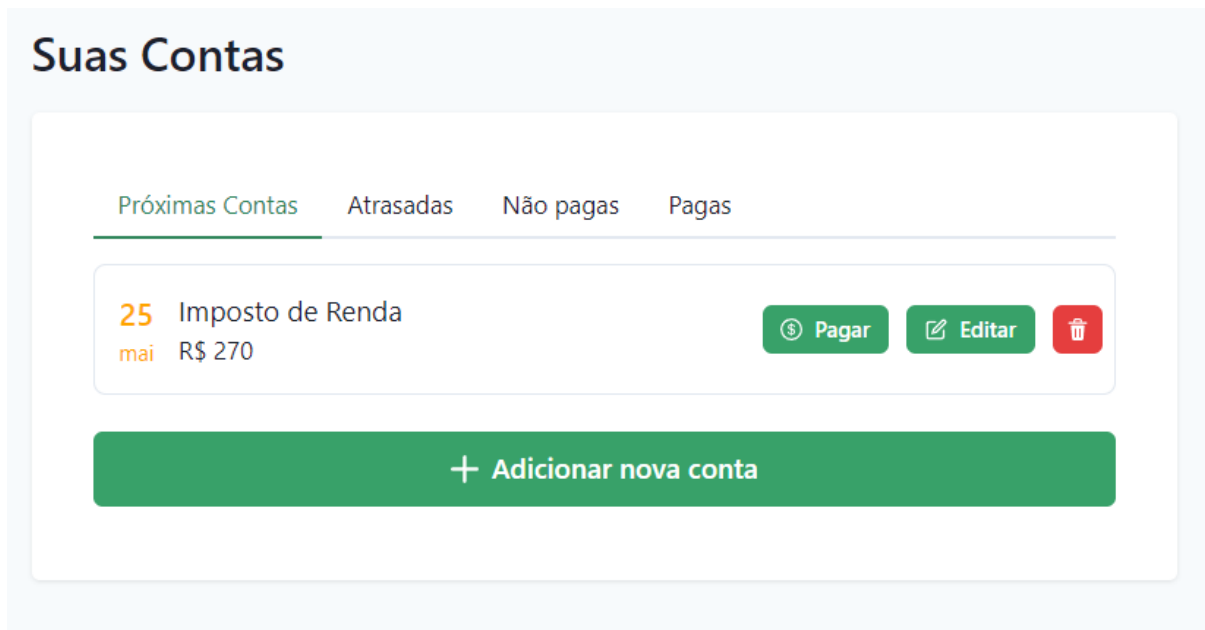


Fonte: Autoria própria

Após a criação da conta, o painel principal será atualizado de acordo com a data de vencimento dessa nova conta. Caso ela esteja entre os próximos 30 dias, ela irá aparecer na aba “Próximas Contas”, como mostra a Figura 20. Nela, aparecerão algumas informações relacionadas a conta, como nome, valor e data de vencimento. Além disso, a data de vencimento possui uma coloração diferente de acordo com o quão próximo ela está da sua data de vencimento. Caso falte 72

horas ou menos para o vencimento da conta, a cor ficará laranja. Caso falte 24 ou menos, a cor será vermelha.

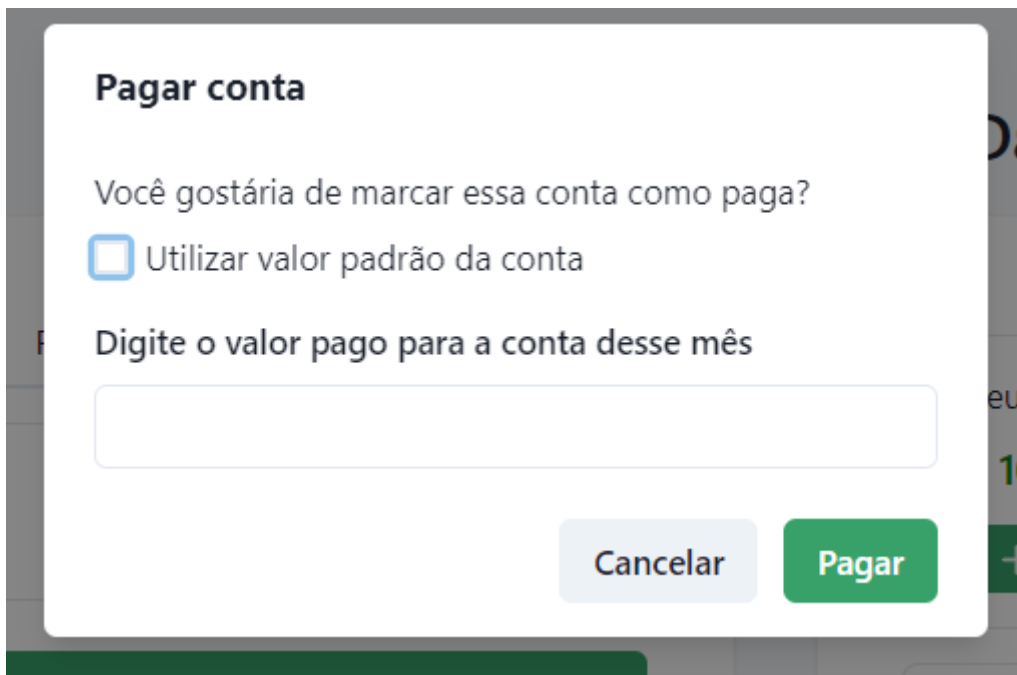
Figura 20: Aba próximas contas



Fonte: Autoria própria

A conta nessa aba também possui três botões de ação, "Pagar", "Editar" e "Excluir". Começando pela funcionalidade de marcar uma conta como paga, após o usuário clicar no botão "Pagar", irá abrir uma *modal*, como mostra a Figura 21. Nela, o usuário pode optar por utilizar o valor padrão da conta ou utilizar um valor customizado para o pagamento. Isso permite com que o usuário possa informar um valor maior por estar atrasada ou menor por pagamentos adiantados.

Figura 21: Modal de pagamento



Pagar conta

Você gostaria de marcar essa conta como paga?

Utilizar valor padrão da conta

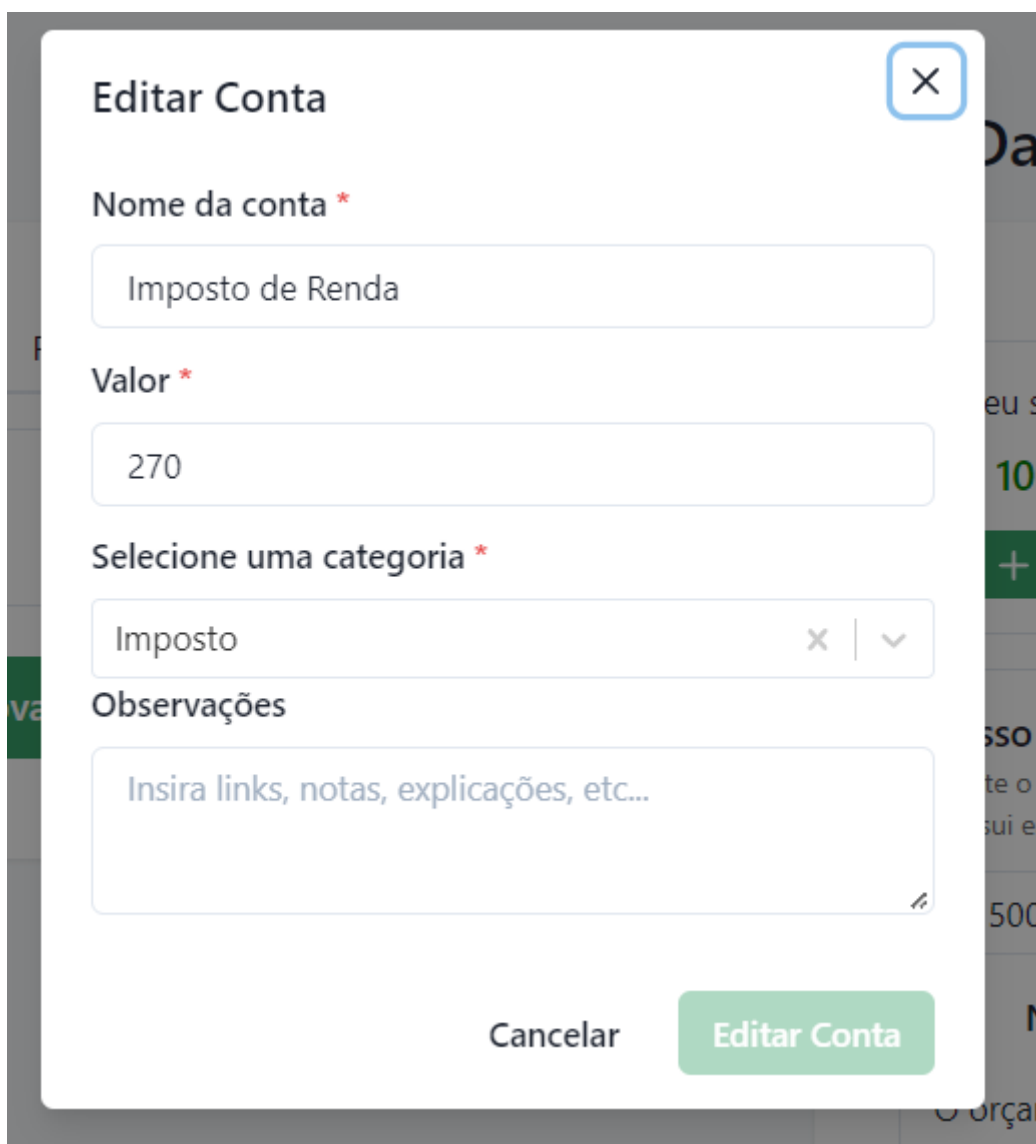
Digite o valor pago para a conta desse mês

Cancelar Pagar

Fonte: Autoria própria

A segunda ação que o usuário poderá realizar é editar algumas informações da conta. Após o usuário clicar em “Editar” irá abrir uma *modal* com um formulário preenchido com as informações atuais da conta, como mostra a Figura 22. Não é possível editar todas as informações como data de vencimento por limitações de como foi projetado o sistema. Portanto, caso o usuário deseje modificar a data de vencimento de uma conta será necessário excluir a conta original e recriá-la com a nova data de vencimento.

Figura 22: Modal edição de conta



Editar Conta

Nome da conta *

Valor *

Selecione uma categoria *

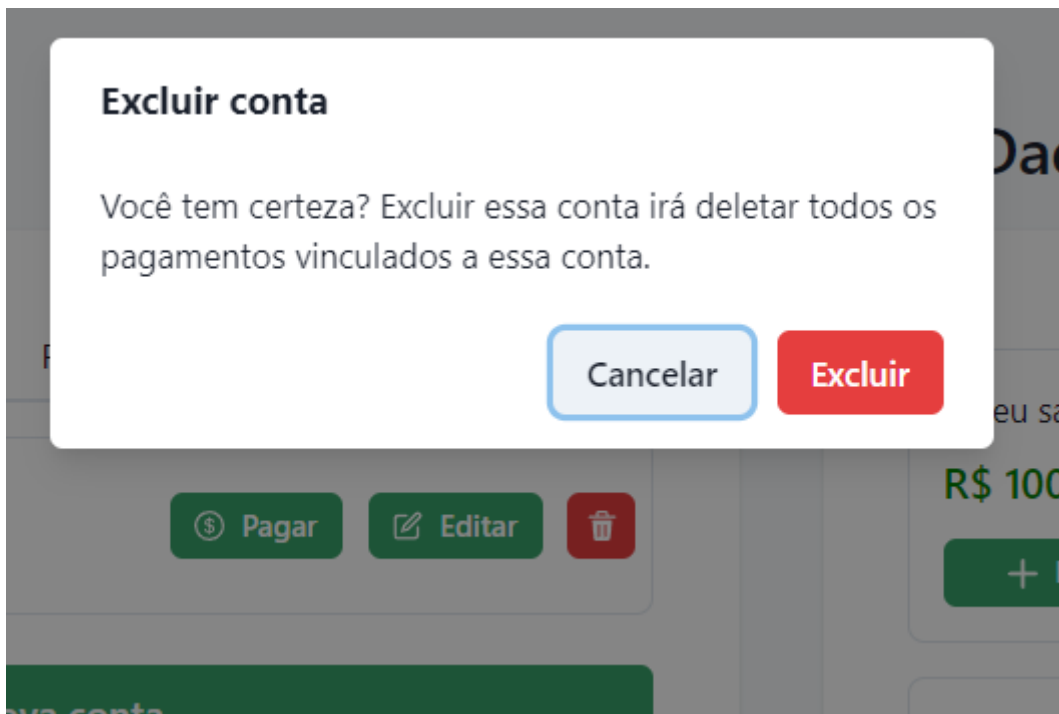
Observações

Cancelar Editar Conta

Fonte: Autoria própria

Por fim, a última ação disponível para o usuário nas contas dessa aba é a exclusão de uma conta. Ao clicar no ícone de lixeira, irá abrir a modal de exclusão confirmando se o usuário realmente deseja excluir essa conta, como exemplificado na Figura 23. Nela, também há um aviso informando que todos os pagamentos vinculados serão excluídos juntos e não somente a do mês atual.

Figura 23: Modal de confirmação de exclusão



Fonte: Autoria própria

Indo para a aba seguinte na seção das contas do usuário. Ao clicar na aba “Atrasadas”, o usuário irá visualizar quais pagamentos não foram marcados como pagos e estão atrasados, como mostra a Figura 24. As ações disponíveis nesta aba são as mesmas do que na aba anterior.

Para o sistema verificar se uma conta está atrasada, foi configurado no Hasura um *cron trigger*. Um *cron trigger* é para executar chamadas HTTP para um servidor para executar lógica de negócio baseada em uma rotina (HASURA, 2022). Essa *trigger* é executada no início de todos os dias, verificando quais pagamentos passaram da data de vencimento e marcando elas como atrasadas.

Figura 24: Aba de contas atrasadas



Fonte: Autoria própria

A Figura 25 mostra a aba “Não pagas”. O sistema irá exibir ao usuário todos os pagamentos futuros pendentes. O período de exibição das contas é entre os próximos seis meses. Para realizar a criação desses pagamentos futuros, semelhante as contas atrasadas, existe um *cron trigger* executar todos os dias para criar os pagamentos dos próximos seis meses de todas as contas cadastradas no sistema.

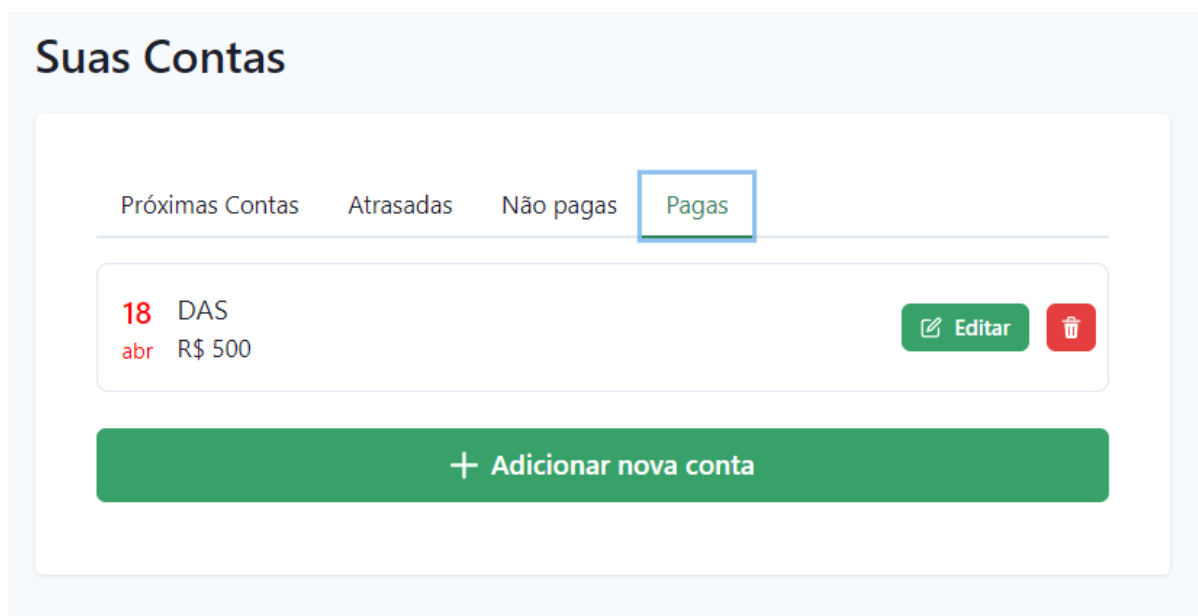
Figura 25: Aba de contas não pagas

Próximas Contas	Atrasadas	Não pagas	Pagas
		18 DAS jun R\$ 500	Pagar Editar
		25 Imposto de Renda jun R\$ 270	Pagar Editar
		18 DAS jul R\$ 500	Pagar Editar
		25 Imposto de Renda jul R\$ 270	Pagar Editar
		18 DAS ago R\$ 500	Pagar Editar
		25 Imposto de Renda ago R\$ 270	Pagar Editar
		18 DAS set R\$ 500	Pagar Editar
		25 Imposto de Renda set R\$ 270	Pagar Editar

Fonte: Autoria própria

A Figura 26 mostra a última aba relacionada às contas: “Pagas”. Nela, será exibida todas as contas marcadas como pagas dos últimos seis meses. Ao contrário das abas anteriores, a ação de pagar não está presente nesta aba.

Figura 26: Aba de contas pagas



Fonte: Autoria própria

Em relação à área designada para exibir os dados financeiros, temos inicialmente a parte em relação ao salário e despesas do mês. Na parte das despesas, o sistema irá calcular de acordo com os gastos previstos do mês atual.

Em relação ao salário, o usuário tem a opção de cadastrar um salário fixo mensal. Ao clicar no botão “Cadastrar salário”, o sistema irá exibir uma caixa contendo um espaço para o usuário informar o seu salário, como mostra a Figura 27.

Figura 27: Modal de cadastro de salário



Fonte: Autoria própria

Além disso, como a figura 28 mostra, o usuário também possui a possibilidade de registrar uma eventual renda extra que ele receber no mês. Ele também poderá editar o salário fixo mensal dele caso haja a necessidade.

Para registrar o histórico de renda mensal do usuário, foi criado um *cron trigger* no Hasura para executar no início de todos os meses para calcular o salário final do mês anterior e guardar o registro no banco de dados.

Figura 28: Modal para adicionar renda extra

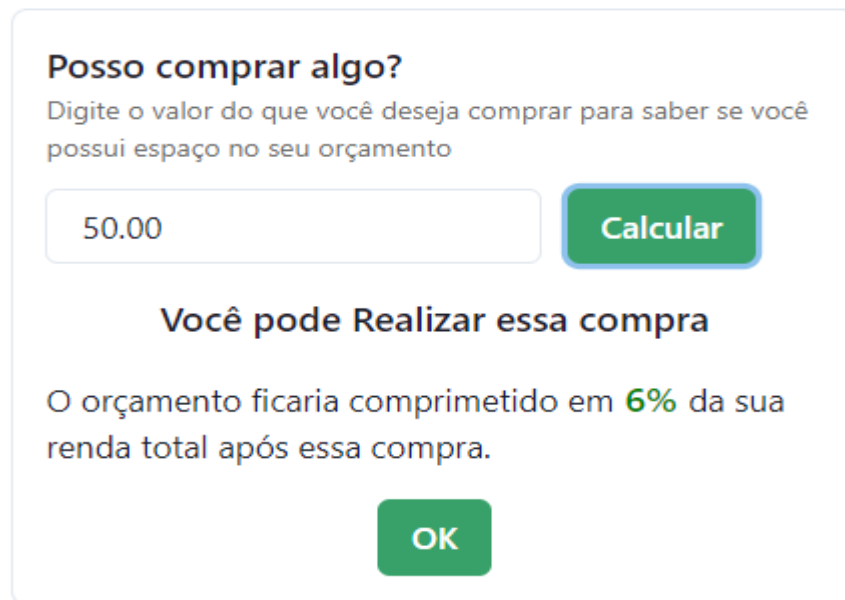


Fonte: Autoria própria

Abaixo do salário e despesas, a seção “Posso comprar algo?” servirá para o usuário simular eventuais compras. O objetivo dessa funcionalidade é informar a ele se ele tem espaço no orçamento para realizar determinada compra. Para isso, foi utilizado a regra do 50/30/20 como base do cálculo.

Foi determinado que caso 80% do orçamento do usuário esteja comprometido, é recomendado que não realizasse essa compra caso não seja uma emergência. As figuras 29 e 30 mostram as mensagens do sistema informando ao usuário se é recomendado fazer determinada compra.

Figura 29: Mensagem informando que o usuário pode realizar determinada compra



Posso comprar algo?
Digite o valor do que você deseja comprar para saber se você possui espaço no seu orçamento

50.00 **Calcular**

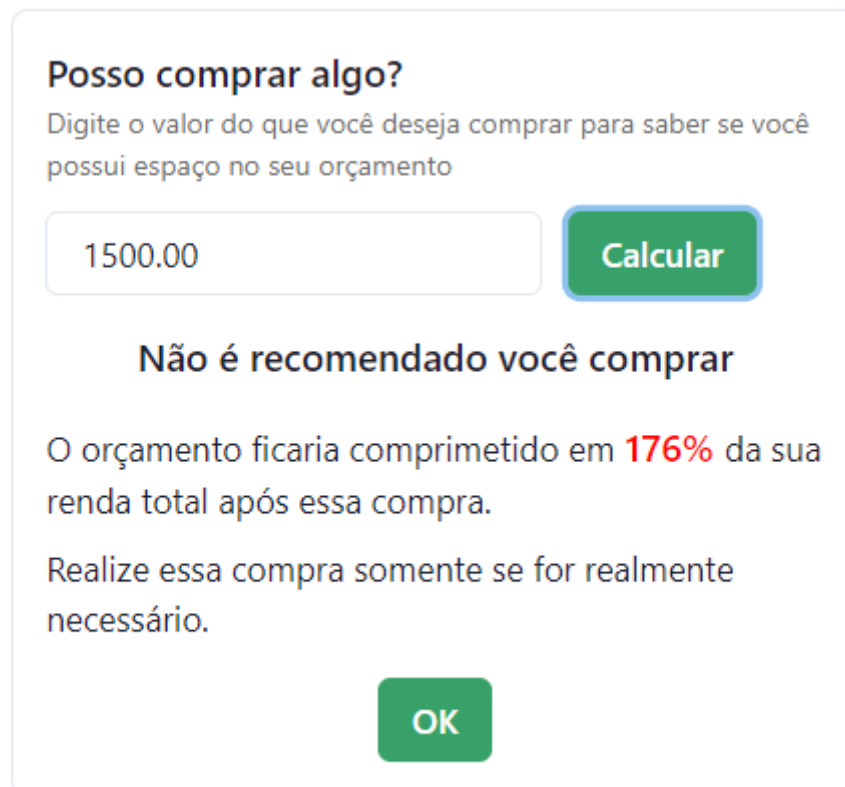
Você pode Realizar essa compra

O orçamento ficaria comprimetido em **6%** da sua renda total após essa compra.

OK

Fonte: Autoria própria

Figura 30: Mensagem informando que o usuário não pode realizar determinada compra



Posso comprar algo?
Digite o valor do que você deseja comprar para saber se você possui espaço no seu orçamento

1500.00 **Calcular**

Não é recomendado você comprar

O orçamento ficaria comprimetido em **176%** da sua renda total após essa compra.

Realize essa compra somente se for realmente necessário.

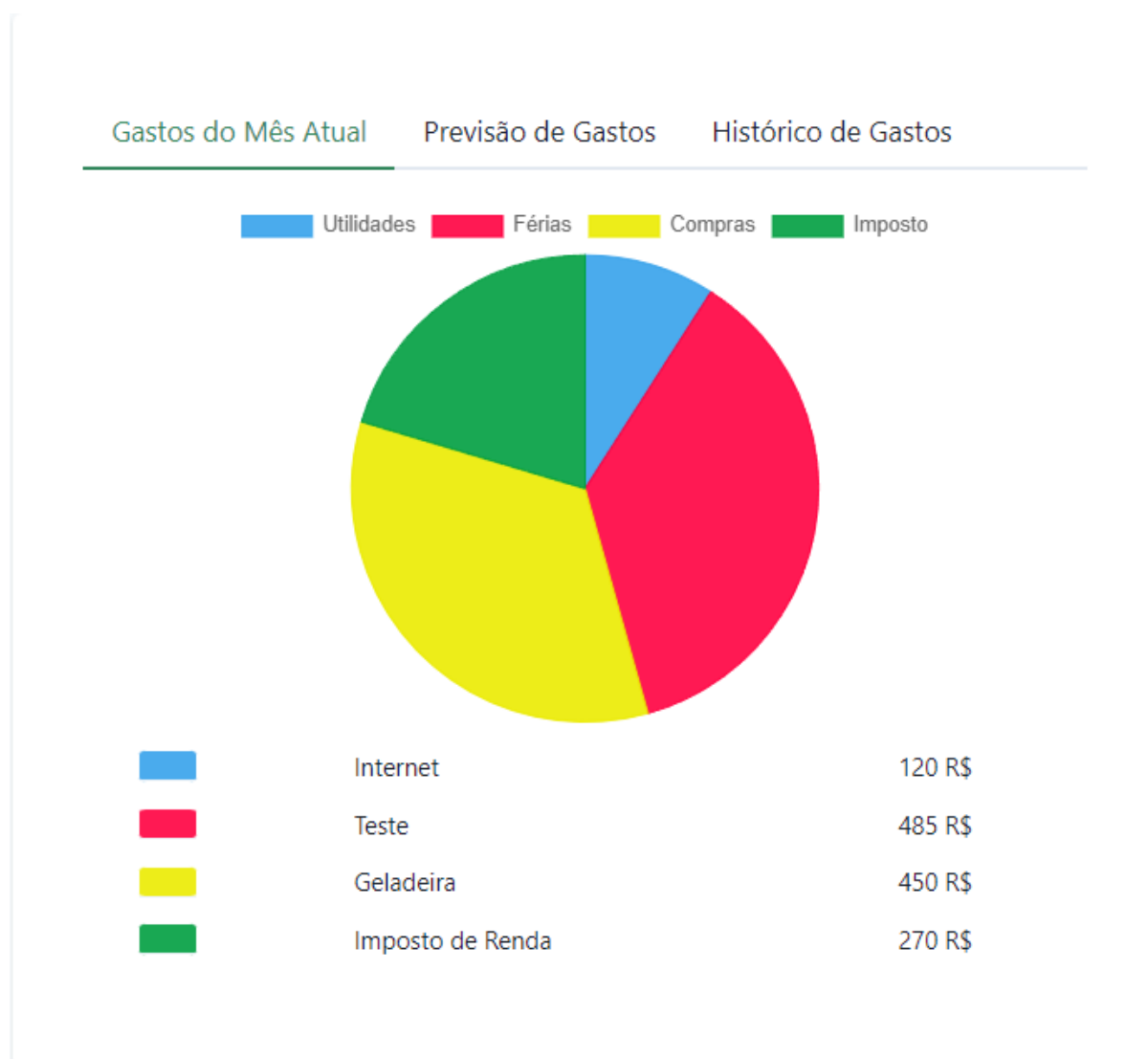
OK

Fonte: Autoria própria

Logo abaixo, está localizada a parte de relatórios gráficos do sistema. Ela possui três seções: “Gastos do Mês Atual”, “Previsão de Gastos” e “Histórico de Gastos”.

A figura 31 mostra a primeira aba, a “Gastos do Mês Atual” onde é exibido as despesas do mês atual por categoria de gasto. Abaixo do gráfico, é exibido cada conta que está sendo contabilizada no cálculo, uma legenda de cor e o valor individual dela.

Figura 31: Gráfico de Gastos do mês atual



Na aba “Previsão de Gastos” será informado os gastos futuros que já cadastrados no sistema. Nela, será mostrado em um período de seis meses para

frente. Além disso, existem dois tipos de visualização: “Todos os meses” para ter um comparativo de mês, como a figura 32 mostra, a mês e “Por categoria” para ter uma visão dos gastos futuros por categoria, ilustrado na figura 33.

Figura 32: Gráfico de previsão de gastos dos últimos seis meses



Fonte: Autoria própria

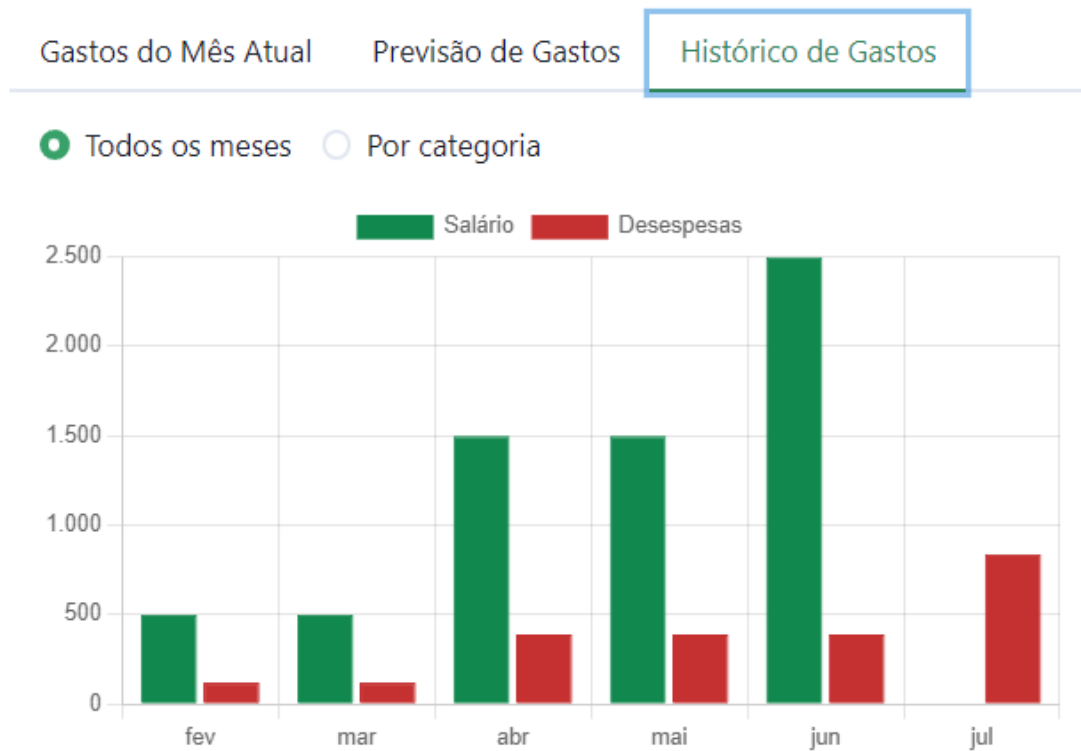
Figura 33: Gráfico de previsão de gastos por categoria de gastos



Fonte: Autoria própria

Também temos a aba “Histórico de gastos”. Ela é semelhante à seção anterior, porém os dados são relacionados aos seis meses anteriores. Nela, é possível você comparar o salário do mês em relação às despesas na visão “Todos os meses” e categoria na visão “Por categoria”, como mostrado na figura 34.

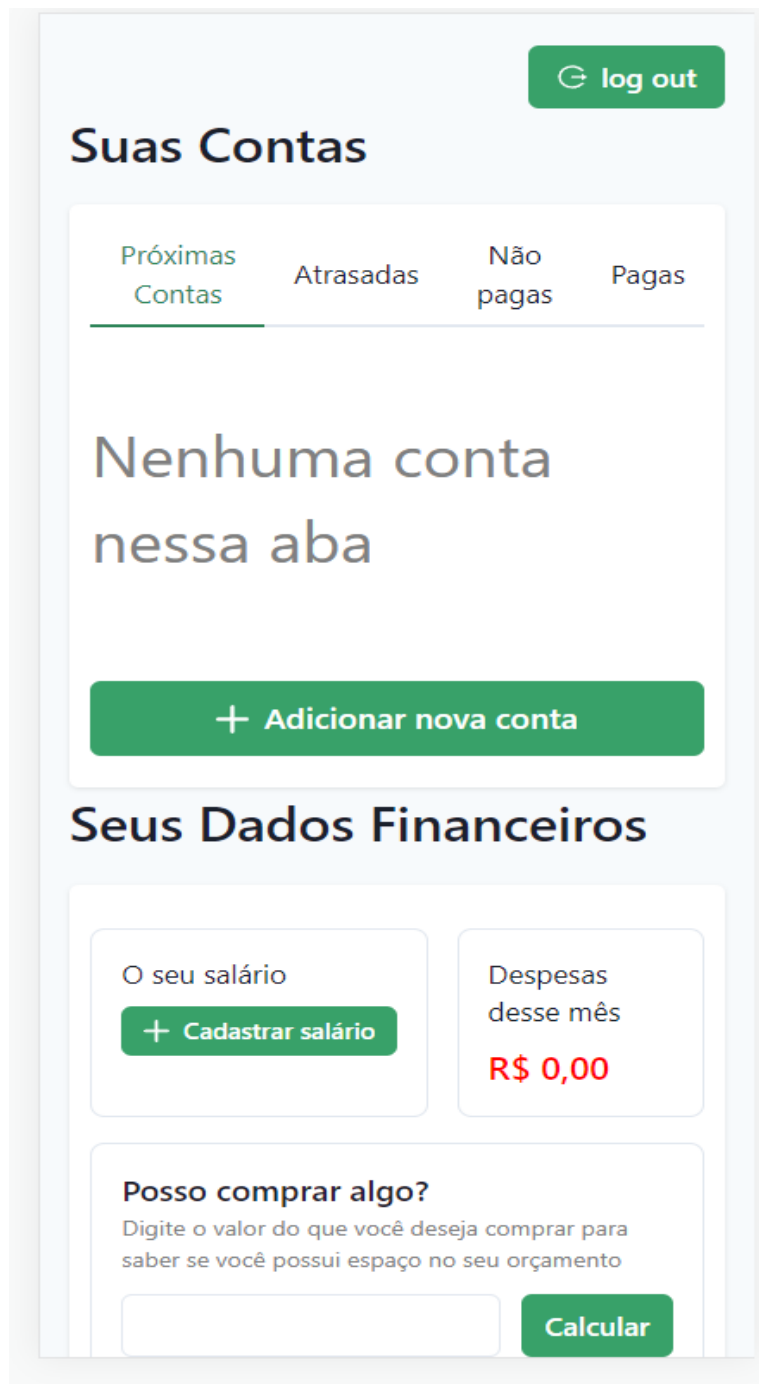
Figura 34: Gráfico de histórico de gastos



Fonte: Autoria própria

Por fim, outro aspecto que foi considerado desde o começo do desenvolvimento do sistema foi a acessibilidade em dispositivos móveis. A interface do sistema foi desenvolvida primeiramente para *smartphones*, com o objetivo de proporcionar uma experiência de usuário agradável mesmo em dispositivos *mobile*, como mostra a figura 35. O motivo para o desenvolvimento ter sido feito primeiramente para celulares foi devido ao fato que desenvolver interfaces para dispositivos menores logo no começo torna muito mais fácil a portabilidade para dispositivos maiores do que o contrário.

Figura 35: Interface do sistema em dispositivos mobile



Fonte: Autoria própria

5.4 Testes

Em relação aos testes realizados nesse trabalho, foram implementados três tipos de testes: testes unitários, testes *end-to-end* e testes de validação com o usuário.

De acordo com Martin Fowler (2018), os testes unitários formam a base de todos os testes de um software. Eles vão garantir que certas partes do código, geralmente funções, estão com o comportamento esperado (FOLWER, 2018). Os testes unitários não foram aplicados em todas as funções do sistema, porém, para garantir que as partes mais críticas do sistema estavam funcionando corretamente, eles foram implementados com a ferramenta Jest tanto no *front-end* quanto *back-end*. As figuras 36 e 37 mostram os testes unitários executados na aplicação, no *front-end* e *back-end* respectivamente.

Figura 36: Testes unitários no *front-end*

```
PASS src/hooks/useCalendar/index.test.ts
useCalendar
  getDayMonthYearString
    ✓ should return the correct object with day and month less than 10 (2 ms)
    ✓ should return the correct object with day and month equal to 9
    ✓ should return the correct object with day and month equal to 10
  buildRecurrenceObj
    ✓ should create monthly recurrence object without up to (2 ms)
    ✓ should create monthly recurrence object with up to
    ✓ should create weekly recurrence object without up to (1 ms)
    ✓ should create weekly recurrence object with up to
    ✓ should create yearly recurrence object without up to (1 ms)
    ✓ should create yearly recurrence object with up to
  createRecurrenceString
    ✓ should RRule for yearly repetition type without up to
    ✓ should RRule for yearly repetition type with up to (1 ms)
    ✓ should ignore repeatUpTo date if its repeat forever
    ✓ should RRule for monthly repetition type without up to
    ✓ should RRule for monthly repetition type with up to
    ✓ should RRule for weekly repetition type without up to
    ✓ should RRule for weekly repetition type with up to

Test Suites: 1 passed, 1 total
Tests:       16 passed, 16 total
Snapshots:   0 total
Time:        2.238 s
Ran all test suites.

Watch Usage: Press w to show more.
```

Fonte: autoria própria.

Figura 37: Testes unitários no *back-end*

```
PASS src/routes/hasura/actions/calculatePayments/index.test.ts
#createPaymentsList
Payment doesn't repeat
  ✓ should create payment already delayed (5 ms)
  ✓ should create payment with due date of current day (1 ms)
  ✓ should create payment with due date in the future (1 ms)
Repeat type once month
  ✓ should create payments already with due date in the past (1 ms)
  ✓ should create payments with due date of the current day (1 ms)
  ✓ should create payments with due date in the future (1 ms)
  ✓ should create payments with final date (4 ms)
Repeat type once week
  ✓ should create payments already with due date in the past (2 ms)
  ✓ should create payments with due date of the current day (1 ms)
  ✓ should create payments with due date in the future (1 ms)
Repeat type once year
  ✓ should create payments already with due date in the past (1 ms)
  ✓ should create payments with due date of the current day
  ✓ should create payments with due date in the future (1 ms)
  ✓ should create payments without final date (1 ms)

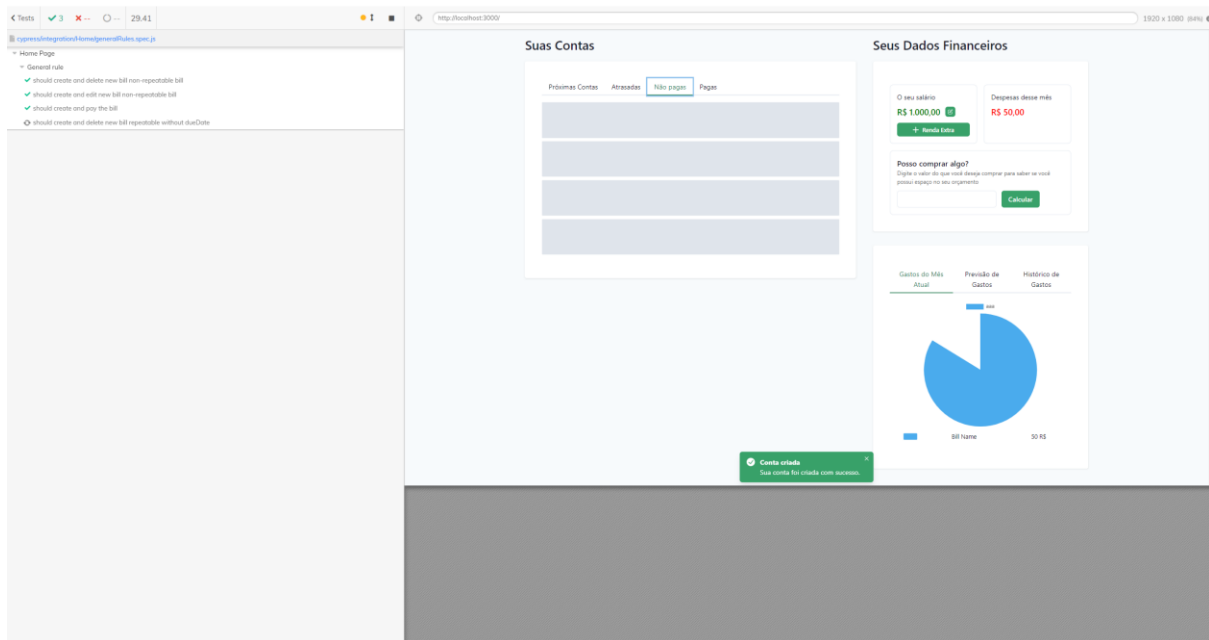
Test Suites: 1 passed, 1 total
Tests: 14 passed, 14 total
Snapshots: 0 total
Time: 4.119 s, estimated 6 s
Ran all test suites.
```

Fonte: Autoria própria

Agora, em se tratando dos testes *end-to-end*, para Martin Folwer (2018), é o tipo de teste que testa todas as funcionalidades do sistema de ponta a ponta, ou seja, do cliente até o servidor. FOLWER (2018) afirma que “testes *end-to-end* são os que dão a maior confiança para decidir se o seu software está funcionando ou não.” (FOLWER, 2018, tradução nossa).

Para realizar testes *end-to-end* foi utilizado a ferramenta Cypress. Com ele é possível escrever testes para simular um usuário interagindo com a aplicação. A implementação da automatização foi direcionada para as funcionalidades de login e algumas funcionalidades relacionadas a criação de uma conta. A figura 38 ilustra a execução dos testes com o Cypress, simulando um usuário real utilizando o sistema.

Figura 38: Testes *end-to-end* com Cypress



Fonte: Autoria própria

Por fim, os testes de validação com o usuário, têm por objetivo para garantir que o software esteja funcionando na perspectiva do usuário, ou seja, atendendo a suas necessidades (FOLWER, 2018).

Foi distribuído via grupos de WhatsApp o acesso a aplicação para os testes de validação serem realizados. Em conjunto, foi entregue um questionário²⁹ no Google Forms para coletar os resultados. As perguntas foram relacionadas ao grau de dificuldade de cada funcionalidade, se vincularam uma conta ao Google Calendar e se o sistema realmente cumpriu o objetivo de auxiliar no planejamento financeiro. Ao todo, foram 6 pessoas que participaram dos testes e responderam ao questionário.

Em relação as perguntas sobre o grau de facilidade, foi questionado sobre as funcionalidades de criação de conta, visualização dos dados financeiros e cadastro na plataforma. As respostas para elas foram em uma escala de um a cinco, onde um significa que foi muito difícil a utilização e cinco muito fácil.

Em relação a facilidade de criação de conta, cinco dos seis participantes classificaram como muito fácil. Todos os participantes classificaram a utilização da visualização dos dados financeiros como muito fácil. E, por fim, todos participantes acharam que o cadastro de conta muito fácil.

²⁹ O questionário na íntegra pode ser visualizado no apêndice C.

Além disso, quatro pessoas utilizaram a funcionalidade de vincular uma conta ao Google Calendar. E perguntados sobre o objetivo do trabalho que era auxiliar no planejamento financeiro, numa escala de um a cinco, onde um era não ajudou e cinco ajudou muito, quatro dos seis participantes responderam que a aplicação ajudou muito na organização do planejamento financeiro.

6 CONSIDERAÇÕES FINAIS

Esse trabalho explicou o desenvolvimento de uma aplicação web chamada YAFMA. A motivação principal para a criação desse sistema foi o baixo nível de educação na população brasileira. A metodologia desse trabalho foi de caráter qualitativo e aplicada com o objetivo de gerar uma solução prática para auxiliar o planejamento financeiro dos usuários.

Para fazer um levantamento dos requisitos para o desenvolvimento da aplicação, foram identificadas quais as funcionalidades ajudariam no planejamento financeiro. Além disso, também foi realizado uma pesquisa em aplicações web e *mobile* já existentes no mercado para analisar possíveis funcionalidades que poderiam ser adicionadas.

O desenvolvimento da aplicação, tanto no *front-end* quanto *back-end*, utilizou Javascript, React no lado do cliente e Node.js no lado do servidor em conjunto com o Hasura para a criação de APIs GraphQL. Após o desenvolvimento, a aplicação foi hospedada na nuvem na plataforma Heroku utilizando o plano gratuito.

Os testes realizados nessa aplicação foram os unitários, *end-to-end* e de validação com os usuários finais. Os testes foram implementados com o objetivo de aumentar a qualidade do software produzido e garantir que os requisitos foram atendidos.

Sabendo que o objetivo principal da aplicação era auxiliar as pessoas no planejamento financeiro. Para validar isso, foi entregue aos usuários o acesso a plataforma e entregue um formulário para coletar informações e saber, se na opinião deles, o sistema atingiu o objetivo. E como a maioria dos participantes responderam que a aplicação é útil para isso, podemos concluir que o objetivo foi alcançado.

6.1 Trabalhos Futuros

Existem diversas melhorias futuras que podem ser feitas para o sistema. Devido o sistema ser feito em React, existe a capacidade de criar um aplicativo *mobile* para os sistemas operacionais Android e iOS com React Native. Assim, o sistema poderia ser utilizado nos principais dispositivos do mercado de forma mais conveniente possível.

Outra melhoria poderia ser integrações com instituições financeiras para tornar o registro de despesas mais automatizado, sem a necessidade de inserção manual do usuário. Porém, isso era inviável para um sistema de porte pequeno e recursos limitados como esse trabalho.

E, por fim, para melhorar a experiência do usuário, realizar o processo de validação da aplicação na Google também seria uma evolução para remover o aviso de segurança na hora de vincular uma conta no Google Calendar.

7 REFERÊNCIAS

ABDALA, Vitor. **Número de endividados no país chega a maior patamar em 11 anos**. 29 jan. 2021. Agência Brasil. Disponível em: <https://agenciabrasil.ebc.com.br/economia/noticia/2021-01/numero-de-endividados-no-pais-chega-maior-patamar-em-11-anos>. Acesso em: 31 mar. 2021.

ALBERTO, Mario; VERNIZZI, Zambrana; CLEDERSON, Passos; ALVES; SANTANA, Rogério. I Encontro das Licenciaturas em Matemática do IFBA. **A IMPORTÂNCIA DA EDUCAÇÃO FINANCEIRA NA EDUCAÇÃO BÁSICA PARA UMA GESTÃO FINANCEIRA CONSCIENTE**, 15-17 out. 2020, Bahia. Disponível em: https://www.researchgate.net/publication/348050120_I_Encontro_das_Licenciaturas_em_Matematica_do_IFBA_A_IMPORTANCIA_DA_EDUCACAO_FINANCEIRA_NA_EDUCACAO_BASICA_PARA_UMA_GESTAO_FINANCEIRA_CONSCIENTE. Acesso em: 02 jun. 2022

BIERMAN, Gavin; ABADI, Martín; TORGERSEN, Mads. European Conference on Object-Oriented Programming, 14., 28 jun -1 ago. 2014, Uppsala. **Understanding TypeScript**. Disponível em: https://link.springer.com/chapter/10.1007/978-3-662-44202-9_11. Acesso em: 02 jun. 2022.

COULOURIS, George; DOLLIMORE, Jean; KINDERBERG, Tim; GORDON, Blair. **Sistemas Distribuídos: Conceitos e Projetos**. 5ª ed. Porto Alegre: Bookman, 2013.

DESHMUKH, Smita; MANE, Deepak; RETAWADE, Abhijeet. 2019 3rd International Conference on Computing Methodologies and Communication (ICCMC), 3., 27-29 mar. 2019, Erode. **Building a Single Page Application Web Front-end for E-Learning site**. Disponível em: <https://ieeexplore.ieee.org/document/8819703>. Acesso em: 17 jun. 2022.

DEVMEDIA. MER e DER: Modelagem de Bancos de Dados. **Devmedia**, 2014. Disponível em: <https://www.devmedia.com.br/mer-e-der-modelagem-de-bancos-de-dados/14332>. Acesso em: 29 maio 2022.

FERREIRA, Juliana Cezario. A importância da educação financeira pessoal para a qualidade de vida. **Caderno de Administração**, Bauru, v. 11, n. 10, dez. 2017. Disponível em: <https://revistas.pucsp.br/index.php/caadm/article/view/33268>. Acesso em: 29 jun. 2022.

FOWLER, Martin. **UML essencial: um breve guia para a linguagem-padrão de modelagem de objetos**. 3ª ed. Porto Alegre: Bookman, 2005.

GACKENHEIMER, Cory. **Introduction to React**. 1ª ed. New York: Apress, 2015.

GERHARDT, Tatiana; SILVEIRA, Denise. **Métodos de pesquisa**. 1ª ed. Porto Alegre: Ed. da UFRGS, 2009. Disponível em: <https://lume.ufrgs.br/handle/10183/52806>. acesso em: 17 jun. 2022.

HASTINGS, Justine; MADRIAN, Brigitte; SKIMMYHORN, William. Financial Literacy, Financial Education, and Economic Outcomes. **Annual Review of Economics**, California, v. 5, p. 347-373, aug. 2013. Disponível em: <https://www.annualreviews.org/doi/abs/10.1146/annurev-economics-082312-125807>. Acesso em: 02 jun. 2022.

HASURA, Introducing Scheduled Triggers: API Driven Cron Jobs and Scheduled Events. **Hasura**, 2020a. Disponível em: <https://hasura.io/blog/introducing-scheduled-triggers-api-driven-cron-jobs-scheduled-events>. Acesso em: 29 maio 2022.

HASURA. Creating a cron trigger. **Hasura**, 2022. Disponível em: <https://hasura.io/docs/latest/graphql/core/scheduled-triggers/create-cron-trigger/>. Acesso em: 27 maio 2022.

HASURA. What is Hasura?. **Hasura**, 2020b. Disponível em: <https://hasura.io/blog/what-is-hasura-ce3b5c6e80e8>. Acesso em: 27 maio 2022.

INFOMONEY. Regra 50-30-20: conheça um método para organizar suas finanças. **InfoMoney**, 2019. Disponível em: <https://www.infomoney.com.br/minhas-financas/regra-50-30-20-conheca-um-metodo-para-organizar-suas-financas>. Acesso em: 29 maio 2022.

KENZIEACADEMY. Front End vs Back End: What's the difference? **Kenzie Academy**, 2020. Disponível em: <https://kenzie.snhu.edu/blog/front-end-vs-back-end-whats-the-difference/>. Acesso em: 29 jun. 2022.

KLAPPER, Leora; LUSARDI, Annamaria; PANOS, Georgios. Financial Literacy and its Consequences: Evidence from Russia during the Financial Crisis. **Journal of Banking and Finance**, Amsterdam, v. 37, n. 10, p. 3904-3923, out. 2013. Disponível em: <https://www.sciencedirect.com/science/article/abs/pii/S0378426613002847>. Acesso em: 02 jun. 2022.

KLAPPER, Leora; LUSARDI, Annamaria; OUDHEUSDEN, Peter van. **Financial Literacy Around the World**. Washington, 2015. Disponível em: <https://www.bbvaedufin.com/en/publicacion/financial-literacy-around-the-world-insights-from-the-standard-poors-ratings-services-global-financial-literacy-survey/>. Acesso em: 02 jun. 2022.

LUCIDCHART. Símbolos e notação de diagramas entidade-relacionamento. **Lucid Chart**, 2018. Disponível em: <https://www.lucidchart.com/pages/pt/simbolos-de-diagramas-entidade-relacionamento>. Acesso em: 04 jun. 2022.

LUSARDI, Annamaria; MITCHELL, Olivia. Financial literacy around the world: an overview. **Journal of Pension Economics and Finance**, Cambridge, v. 10, n. 4, p.

497-508, out. 2011. Disponível em: <https://www.cambridge.org/core/journals/journal-of-pension-economics-and-finance/article/abs/financial-literacy-around-the-world-an-overview/0488F901318E0FBC4C92DC6E964AB89C>. Acesso em: 02 jun. 2022.

MIRANDA, Parr. **Budgeting for the Cost of Living**. Oklahoma, 2019. Disponível em: <https://shareok.org/handle/11244/329376>. Acesso em: 29 maio 2022.

MOTA, Antônio. *et al.* **UML 2.0 – Unified Modeling Language 2.0**. Porto, 200?. Disponível em: <https://paginas.fe.up.pt/~ei02084/artigo.pdf>. Acesso em: 05 jul. 2002.

MOZILLA. Começando com React. **MOZILLA**, 2021a. Disponível em: https://developer.mozilla.org/pt-BR/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/React_getting_started. Acesso em: 27 maio 2022.

MOZILLA. Começando com React. **MOZILLA**, 2021b. Disponível em: <https://developer.mozilla.org/en-US/docs/Glossary/SPA>. Acesso em: 27 maio 2022.

MOZILLA. O que é JavaScript?. **MOZILLA**, 2022. Disponível em: https://developer.mozilla.org/pt-BR/docs/Learn/JavaScript/First_steps/What_is_JavaScript. Acesso em: 27 maio 2022.

NODEJS. About Node.js. **NODEJS**, 2011. Disponível em: <https://nodejs.org/en/about>. Acesso em: 27 maio 2022.

NUBANK. Como criar um planejamento financeiro pessoal eficiente. **Nubank**, 2020. Disponível em: <https://blog.nubank.com.br/planejamento-financeiro-pessoal>. Acesso em: 31 mar. 2021.

PAGOTTO, Tiago; FABRI, José Augusto; LERARIO, Alexandre; GONÇALVES, José Antonio. Iberian Conference on Information Systems and Technologies (CISTI), 16., 15-18 jun. 2016, Gran Canaria. **Scrum solo: Software process for individual development**. Disponível em: <https://ieeexplore.ieee.org/document/7521555>. Acesso em: 02 jun. 2022.

POSTGRESQL. About. **POSTGRESQL**, 2022. Disponível em: <https://www.postgresql.org/about/>. Acesso: 17 jun. 2022.

PRESSMAN, Roger S. **Engenharia de software: uma abordagem profissional**. 7ª ed. Porto Alegre: AMGH, 2011.

PUENTE, Beatriz; JANONE, Lucas. Endividamento atinge 77,7% das famílias brasileiras, maior patamar desde 2010, diz confederação. **CNN Brasil**, 2022. Disponível em: <https://www.cnnbrasil.com.br/business/endividamento-atinge-777-das-familias-brasileiras-maior-patamar-desde-2010-diz-confederacao/>. Acesso em: 26 mai. 2022.

STEMMLER, Khalil. What is GraphQL? GraphQL introduction. **Apollo GraphQL**, 2021. Disponível em: <https://www.apollographql.com/blog/graphql/basics/what-is-graphql-introduction/>. Acesso em: 30 jun. 2022.

STOLPER, Oscar; WALTER, Andreas. Financial literacy, financial advice, and financial behavior. **Journal of Business Economics**, Berlin, n. 87, p. 581–643, mar. 2017. Disponível em: <https://link.springer.com/article/10.1007/s11573-017-0853-9>. Acesso em: 02 jun. 2022

SULYMAN, Shakirat. Client-Server Model. **IOSR Journal of Computer Engineering**, Oxford, v. 16, n. 1, p. 67-71, jan. 2014. Disponível em: https://www.researchgate.net/publication/271295146_Client-Server_Model. Acesso Em: 02 jun. 2022.

TANENBAUM, Andrew S. **Redes de computadores**. 5ª ed. São Paulo: Pearson, 2011.

TILKOV, Stefan; VINOSKI, Steve. Node.js: Using JavaScript to Build High-Performance Network Programs. **IEEE Internet Computing**, Estados Unidos, v. 15, n. 6, p. 80-83, nov. 2010 Disponível em: <https://ieeexplore.ieee.org/document/5617064>. Acesso em: 02 jun. 2022

TUTORIALSPPOINT. Como criar um planejamento financeiro pessoal eficiente. **Tutorialspoint**, 2016. Disponível em: https://www.tutorialspoint.com/javascript/javascript_overview.htm. Acesso em: 05 jul. 2022.

VANDERLINDE, Anair; GODOY, Nádia nara de. PLANEJAMENTO FINANCEIRO E SEUS BENEFÍCIOS. **Maiêutica**, Indaial, v. 1, n. 1, maio 2014. Disponível em: http://publicacao.uniasselvi.com.br/index.php/CTB_EaD/article/view/1230/389. Acesso em: 20 jun. 2022.

APÊNDICE A – Especificação dos Casos de Usos

Quadro 2: Especificação dos casos de usos

Código e nome do caso de uso	CDU001 Gerenciar contas a pagar
Ator Primário	Usuário
Fluxo Principal	<p>P1. O sistema exibe a tela principal. P2. O usuário clica no botão de adicionar nova conta. P3. O sistema apresenta a modal com o formulário de criação de conta. P4. O usuário preenche o formulário. P5. O sistema valida o formulário. P6. O usuário solicita a criação. P7. O sistema exibe a mensagem de sucesso. P8. Fim do Caso de Uso</p>
Fluxos Alternativos	<p>A1 Vincular conta no Google Calendar A1.1. Em P4, o usuário seleciona a opção de vincular conta com Google Calendar A1.2. O sistema apresenta o botão de autenticar-se com o Google. A1.3. O usuário clica no botão para se autenticar com o Google. A1.4. O sistema abre uma nova janela pedindo os dados da conta Google. A1.5. O usuário preenche o email e senha e os envia. A1.6. Sistema autentica o usuário, fechando a janela e retorna para P4 A1.7. retorna para P4</p> <p>A2 Editar Conta existente A2.1. Em P2, o usuário clica no botão de editar conta A2.2. O sistema abre a modal de edição A2.3. O usuário alterar algum valor existente e solicita a edição A2.4. O sistema altera os dados da conta e exibe a mensagem de sucesso A2.5. retorna para P1</p> <p>Excluir conta existente A3.1. Em P2, o usuário clica no botão de excluir conta A3.2. O sistema abre a modal de confirmação A2.3. O usuário confirma a exclusão A2.4. O sistema exclui a conta e exibe a</p>

	mensagem de sucesso A2.5. retorna para P1
Fluxos de exceção	E1 Formulário inválido E1.1. Em P4, o usuário preenche um campo incorretamente E1.2. O sistema destaca o campo inválido E1.3. Retorna para P4

Código e nome do caso de uso	CDU002 Visualizar dados referentes ao orçamento
Ator Primário	Usuário
Fluxo Principal	P1. O sistema exibe a tela principal. P2. O usuário clica em Gastos do mês atual. P3. O sistema apresenta os dados referente a aba. P4. Fim do caso de uso
Fluxos Alternativos	<p>A1 Visualizar previsão de gastos A1.1. Em P2, o usuário clica em previsão de gastos. A1.2. O sistema exibe os dados desta visão. A1.3. O usuário seleciona a forma de apresentar os dados. A1.4. Retorna para o P3</p> <p>A2 Visualizar histórico de gastos A2.1. Em P2, o usuário clica em histórico de gastos. A2.2. O sistema exibe os dados desta visão. A2.3. O usuário seleciona a forma de apresentar os dados. A2.4. Retorna para o P3</p> <p>A3 Verificar se é possível fazer compra A3.1. Em P2, o usuário fornece um valor ao formulário Posso comprar algo? e clica em calcular A3.2. O sistema realiza o calculo de acordo com a RNG14, e exibe o resultado. A3.3. O usuário clica em ok. A3.4. O sistema remove o resultado. A3.4. Fim do caso de uso</p>
Fluxos de exceção	E1 O usuário não possui dados cadastrados E1.1. Em P1, caso o usuário não possua

	dados, o sistema irá informá-lo E1.2. Fim do caso de uso
--	---

Código e nome do caso de uso	CDU003 Gerenciar renda mensal
Ator Primário	Usuário
Fluxo Principal	P1. O sistema exibe a tela principal. P2. O usuário clica em cadastrar salário. P3. O sistema exibe modal de inserção do salário P4. O usuário insere o salário e salva-o. P5. O sistema valida e exibe a mensagem de sucesso. P6. Fim do caso de uso.
Fluxos Alternativos	A1 Edição de salário A1.1. Em P2, o usuário clica em editar salário. A1.2. O sistema exibe modal de edição do salário atual A1.3. O usuário fornece o novo salário e salva-o. A1.4. O sistema valida, exibe a mensagem de sucesso e atualiza o salário. A1.5. Retorna para P1. A2 Adição de renda extra no mês A1.1. Em P2, o usuário clica em adicionar renda extra. A1.2. O sistema exibe modal de adição de salário. A1.3. O usuário fornece um valor e salva-o. A1.4. O sistema valida e exibe a mensagem de sucesso e adiciona ao salário mensal. P5. Retorna para P1.
Fluxos de exceção	E1 Salário inválido E1.1. Em P4, o usuário preenche o campo incorretamente E1.2. O sistema destaca o campo inválido E1.3. Retorna para P4

Regras de negócio

RNG01: A aba de próximas contas deve exibir as contas dos próximos 30 dias não atrasadas.

RNG02: A aba de atrasados deve exibir contas com o pagamento atrasado.

RNG03: A aba de não pagas deve exibir contas futuras que ainda não foram pagas.

RNG04: A aba de pagas deve exibir contas que o usuário já marcou como pagas.

RNG05: Ao excluir uma conta, todos os pagamentos vinculados a ela devem ser removidos do sistema.

RNG06: A data de vencimento da conta deve ficar com a cor amarela caso falte menos de 72 horas para o vencimento. E deve ficar com a cor vermelha caso falta menos de 24 horas para o vencimento.

RNG07: Para cadastrar uma conta é necessário fornecer data de vencimento, nome, valor e categoria.

RNG08: Uma conta pode ter três tipos de repetição: semanal, mensal, anual.

RNG09: Caso a conta seja repetível, é necessário fornecer uma data final do último pagamento ou informar que não possui uma data final determinada.

RNG10: Para vincular uma conta com o Google Calendar, é preciso realizar o login com uma conta Google na aplicação.

RNG11: Somente as seguintes informações podem ser editadas em uma conta: Nome, Valor, Categoria e observações.

RNG12: No momento que o usuário pagar a conta, ele pode escolher alterar o valor do pagamento.

RNG13: Deve ser possível marcar contas como pagas em qualquer momento.

RNG14: Caso a suposta compra faça com que 80% ou mais do orçamento fique comprometido, deve-se exibir um aviso para o usuário não recomendado a compra.

RNG15: Na aba de Gasto do mês atual, o gráfico deve exibir qual o gasto previsto desse mês. Separando por categoria e disponibilizando um detalhamento na parte inferior.

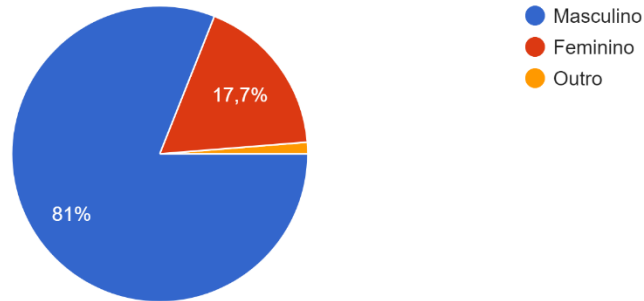
RNG16: Na aba de previsão de gastos, o sistema deve exibir a previsão dos gastos dos próximos seis meses. Podendo visualizar todos os meses simultaneamente ou cada mês individualmente por categoria de gastos.

RNG17: Na aba de histórico de gastos, o sistema deve exibir o histórico dos gastos dos seis meses anteriores em comparação com a renda do mês.

APÊNDICE B – Questionário sobre educação na íntegra

Figura 39: Gênero dos participantes

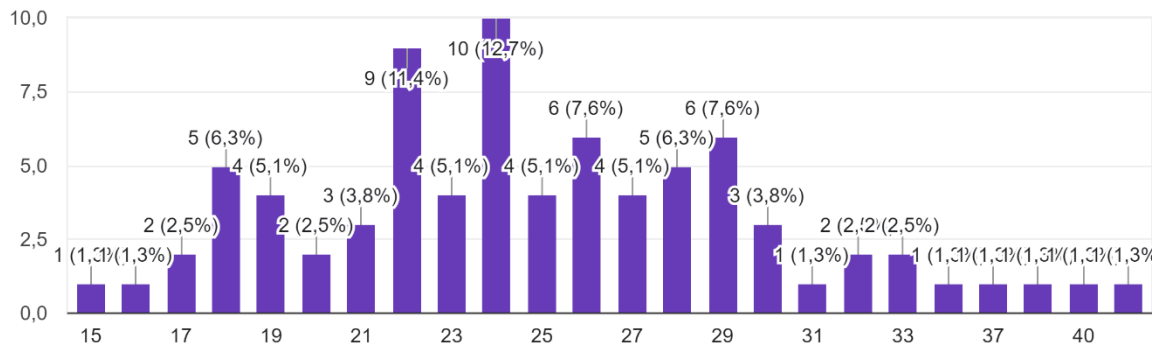
Gênero
79 respostas



Fonte: Autoria Própria

Figura 40: Idade dos participantes

Idade
79 respostas

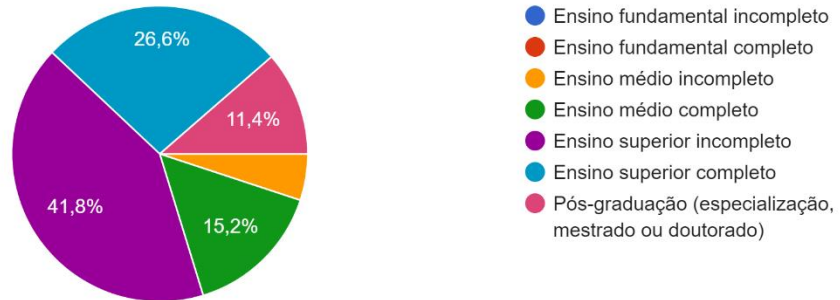


Fonte: Autoria Própria

Figura 41: Nível de escolaridade

Escolaridade

79 respostas

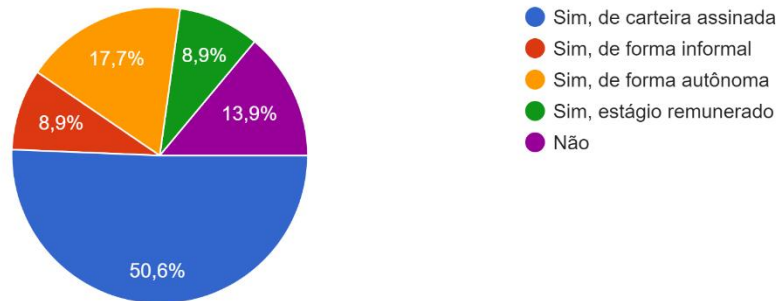


Fonte: Autoria Própria

Figura 42: Realizam alguma atividade remunerada

Você realiza alguma atividade remunerada?

79 respostas

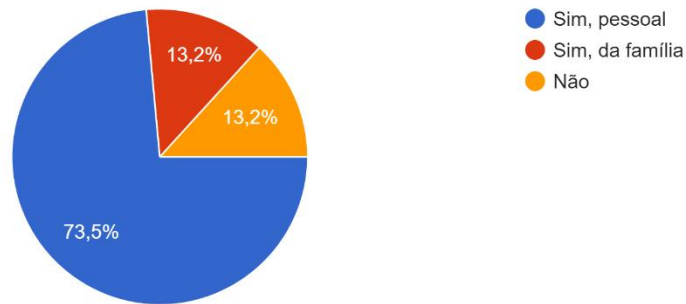


Fonte: Autoria Própria

Figura 43: Realizam controle financeiro

Você realiza algum tipo de controle financeiro das suas finanças pessoais ou da sua família?

68 respostas

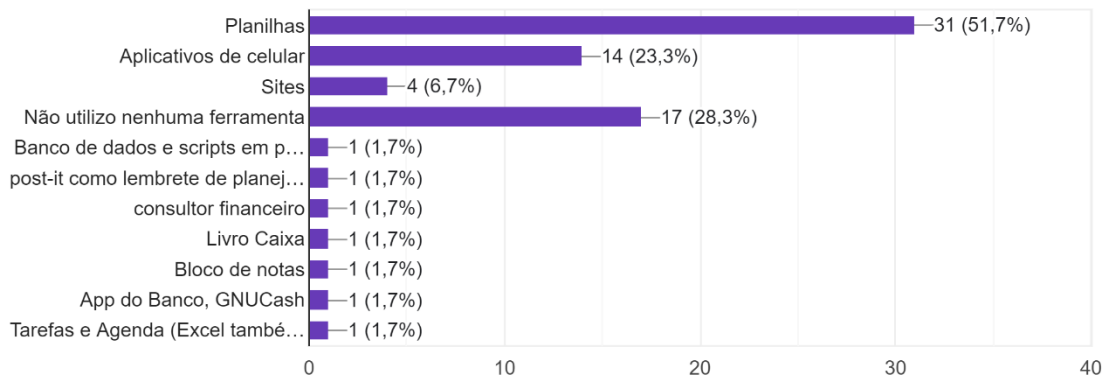


Fonte: Autoria Própria

Figura 44: Utilizam algum aplicativo ou site para organização

Você utiliza algum aplicativo ou site para organizar suas finanças?

60 respostas

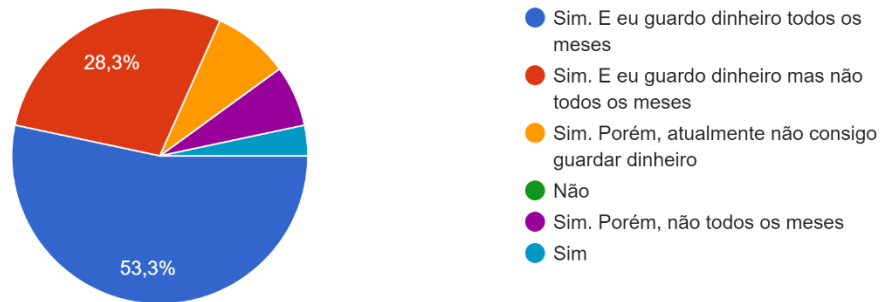


Fonte: Autoria Própria

Figura 45: Costumam planejar em poupar dinheiro

Você costuma se planejar em poupar o seu dinheiro?

60 respostas

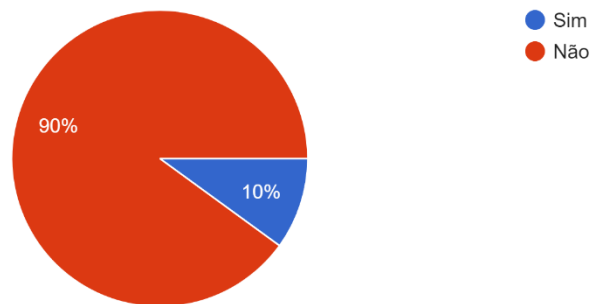


Fonte: Autoria Própria

Figura 46: Participantes endividados

Você está endividado atualmente?

60 respostas

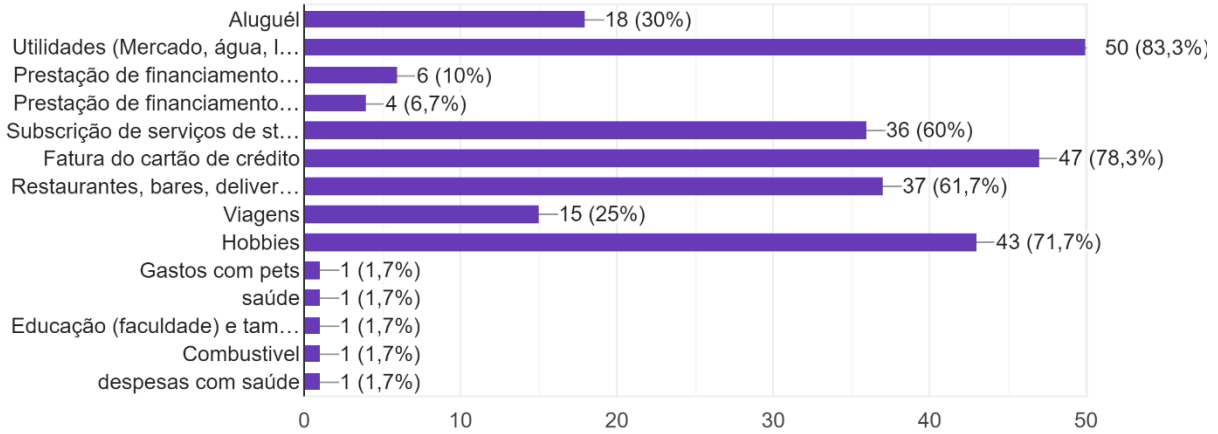


Fonte: Autoria Própria

Figura 47: Tipos de despesas

Quais os tipos de despesas que você possui?

60 respostas

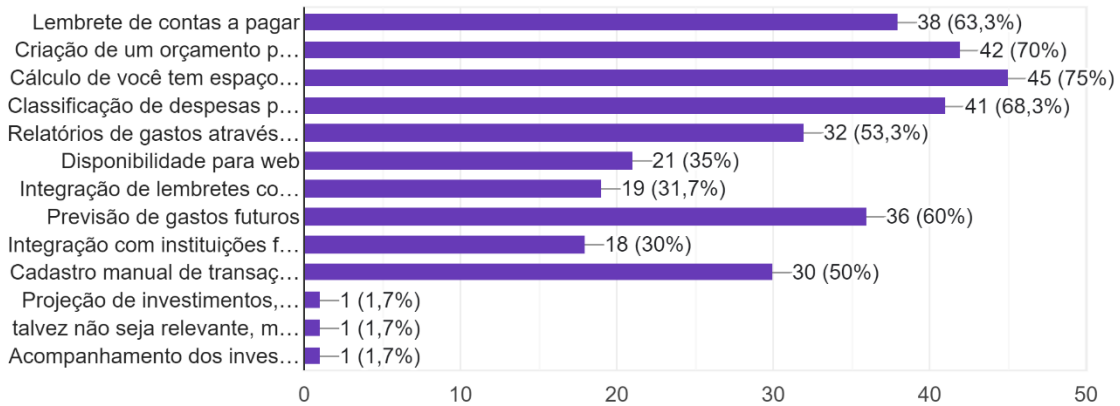


Fonte: Autoria Própria

Figura 48: Funcionalidades essenciais para um sistema de finanças pessoais

Quais funcionalidades você acha essencial para um sistema de controle de finanças pessoais

60 respostas

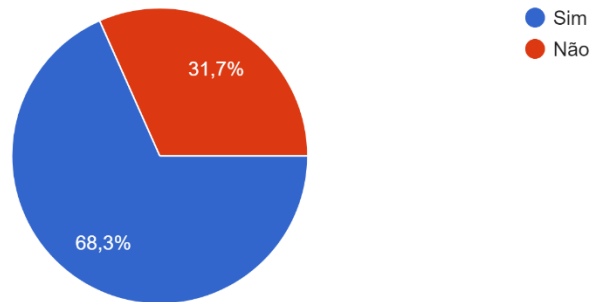


Fonte: Autoria Própria

Figura 49: Participantes interessados em um site de planejamento financeiro

Você teria interesse em um site para realizar a gestão das suas finanças pessoais?

60 respostas



Fonte: Autoria Própria

Figura 50: Considerações finais

Se você tiver alguma consideração final ou dica, utilize esse espaço para isso

8 respostas

Uma boa proposta seria a criação de metas individualizadas para fins de investimento e planejamento.

Um site muito brabo que unisse tudo seria muito brabo

Não

Site não, prefiro app

Uma dica de serviço oferecido seria simulação de gastos, empréstimos baseado na renda e na taxa de juros base, ou como eu deveria poupar para realizar um sonho de comprar um carro, baseado na renda, por exemplo.

site nao seria interessante mas app sim

Tenho mais interesse em aplicativos mobile do que em sites web.

Nunca conte com o dinheiro que você vá receber, guarde o que você já tem (sempre vai haver um imprevisto), prioridades em primeiro lugar.

Fonte: Autoria Própria

APENDICÊ C – Questionário para a avaliação da aplicação na integra

Figura 51: Gênero dos participantes

Gênero
6 respostas



Fonte: Autoria Própria

Figura 52: Idade dos participantes

Idade
6 respostas

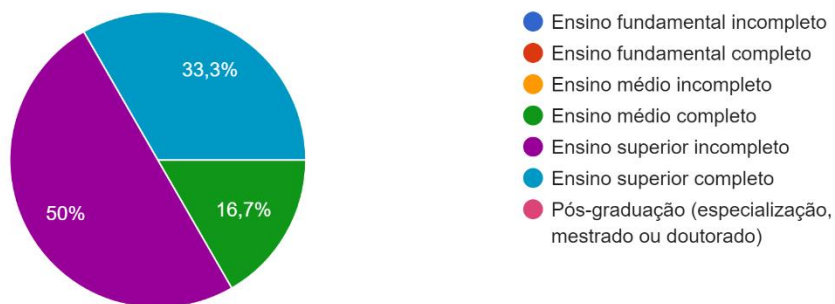


Fonte: Autoria Própria

Figura 53: Escolaridade dos participantes

Escolaridade

6 respostas

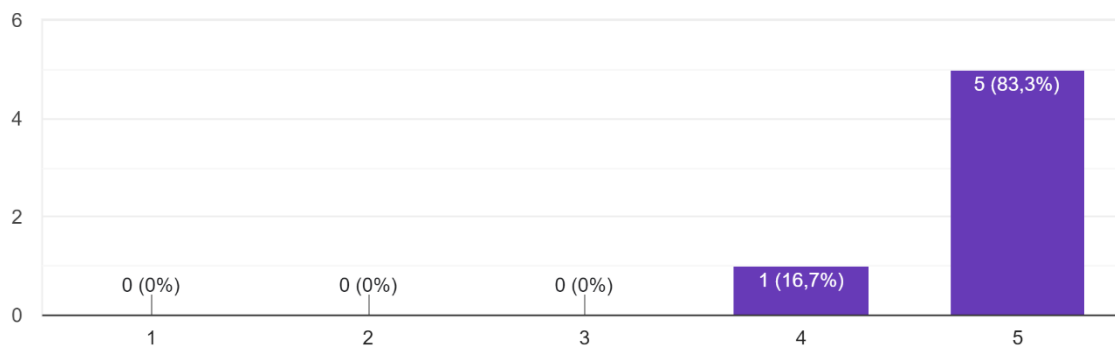


Fonte: Autoria Própria

Figura 54: Grau de facilidade para se cadastrar

Em uma escala de um a cinco, sendo um muito difícil e cinco muito fácil: Qual foi o grau de facilidade para criar um cadastro no sistema na sua opinião?

6 respostas

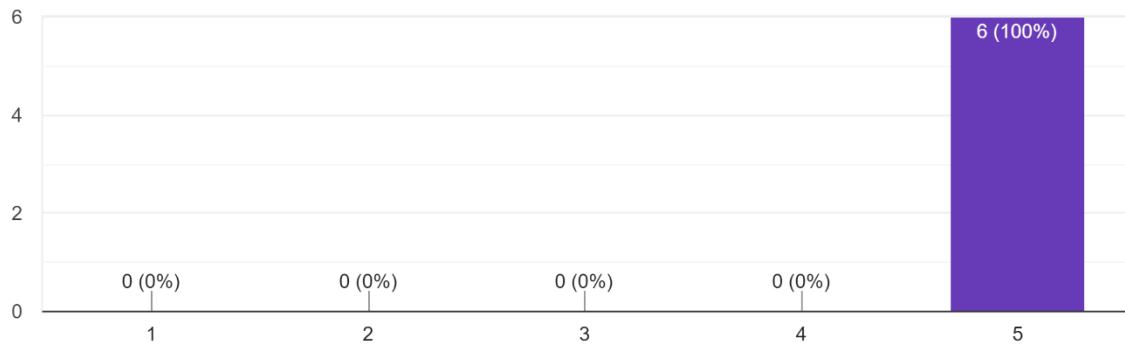


Fonte: Autoria Própria

Figura 55: Grau de facilidade para criar uma conta

Em uma escala de um a cinco, sendo um muito difícil e cinco muito fácil: Qual foi o grau de facilidade para criar uma conta na sua opinião?

6 respostas

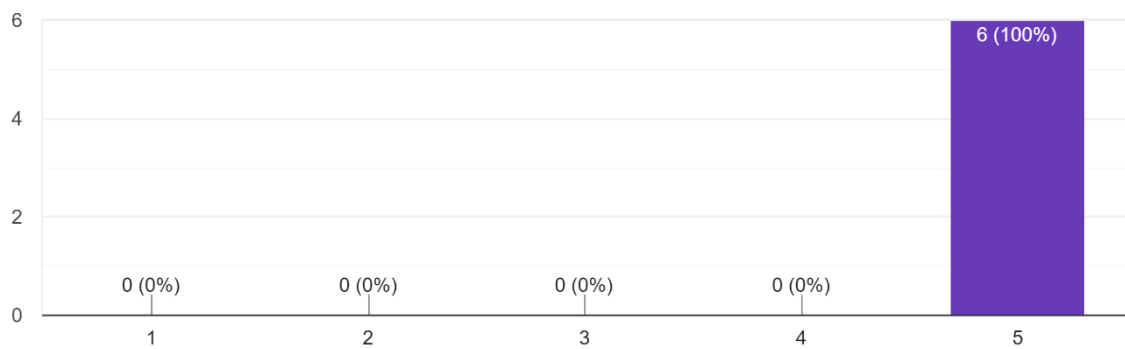


Fonte: Autoria Própria

Figura 56: Grau de facilidade para visualizar dados financeiros

Em uma escala de um a cinco, sendo um muito difícil e cinco muito fácil: Qual foi o grau de facilidade para visualizar os seus dados financeiros na sua opinião?

6 respostas

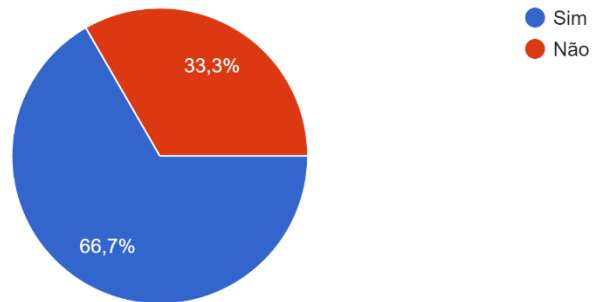


Fonte: Autoria Própria

Figura 57: Utilização do Google Calendar

Você utilizou a funcionalidade de vincular uma conta no seu Google Calendar?

6 respostas

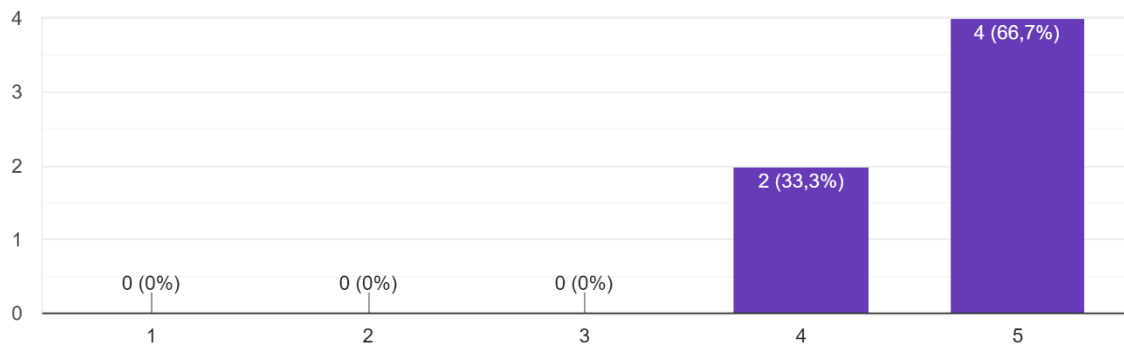


Fonte: Autoria Própria

Figura 58: A aplicação ajudou no planejamento financeiro?

O objetivo do trabalho de conclusão era auxiliar no planejamento financeiro. Em uma escala de um a cinco, sendo um não ajudou e cinco ajudou muit...ajuda na organização do planejamento financeiro?

6 respostas



Fonte: Autoria Própria

Figura 59: Observações finais

Se você desejar, deixe uma sugestão ou alguma observação que possa contribuir com o trabalho.

2 respostas

No histórico de gastos, gostaria de saber qual ano de qual mês estou visualizando.
Colocar um hover em alguns botões para demonstrar ser clicavel.
De resto, está bem intuitivo e com uma ótima responsividade.

excelente trabalho

Fonte: Autoria Própria