RECA
REVISTA ELETRÔNICA DE COMPUTAÇÃO APLICADA
VOL. 1 NRO. 2 2020

DESENVOLVIMENTO DE SISTEMA PARA AUXILIAR NA ORGANIZAÇÃO E PLANEJAMENTO DE CERIMÔNIAS MATRIMONIAIS

Diego Henrique Silva de Jesus
Graduando em Engenharia de Software – Uni-FACEF
diegohenriqsilva@hotmail.com

Henrique de Paula Abrahão
Graduando em Engenharia de Software – Uni-FACEF
henriquedepaulasp@gmail.com

Ely Fernando do Prado
Mestre em Ciência da Computação – Uni-FACEF
elyfprado@facef.br

Resumo

Organizar um casamento pode ser mais complexo do que parece, por isso ter a ajuda de um cerimonialista, profissional em organização de casamento, é indispensável, já que envolve muitos fatores e pessoas, e exige estar em constante comunicação com os noivos. Muitas vezes um cerimonialista precisa de ferramentas para auxiliar o seu trabalho, a partir dessa problemática, foi pensado na solução através de um aplicativo móvel que auxilie a comunicação e organização do casamento com os noivos e o cerimonialista. O software foi desenvolvido baseado em conversa com profissionais de organização de casamento, em que se percebeu a necessidade de uma ferramenta que contribuísse para a organização deste evento. Um site desenvolvido em React para o gerenciamento da cerimonialista e um aplicativo para os noivos, criado com kit de desenvolvimento de interface de usuário Flutter. Ao decorrer do desenvolvimento foi observado e concluído que o trabalho pode ser expandido devido o evento de uma cerimônia de casamento ser uma área bem extensa e precisa de diversas funcionalidades a serem criadas para facilitar a execução de grandes tarefas, tanto para noivos quanto para cerimonialista.

Palavras-chave: cerimonialista. Casamento. aplicativo móvel. React. Flutter.

Abstract

Organising a wedding can be more complex than it seems, that's why having help of a wedding planner, professional of wedding organisation is essential, since then it involves many factors and people, and requires having constant communication with the bride and the groom. Several times, the wedding planner needs tools to help with his work, from this problem, the solution has been thought through a mobile phone application that helps the communication and organisation of the wedding between the bride, the groom and the wedding planner. The software has been developed based on a conversation with wedding planners, which the need of a tool that contributes with the organisation of this event has been realised. A web site developed in React for the wedding planner's management and a mobile phone application for the bride and the groom, created with the user's interface development kit Flutter.

During the progress it had been observed and concluded that the work can be expanded, due to wedding ceremony's events, that is a very vast area and needs many functionalities to be created to help with the execution of big tasks, for the bride, the groom and the wedding planner.

Keywords: *wedding planner. Wedding. mobile phone application. React. Flutter.*

1 Introdução

A organização e o planejamento de um casamento podem ser muito complexos por envolver vários fatores e que, se não seguir um cronograma, várias coisas podem dar errado. Desse modo, pensando em soluções para este problema, foi elaborado um aplicativo móvel inovador que organiza, faz planejamentos e possui área de comunicação, para auxiliar o cerimonialista e os noivos na organização da cerimônia do casamento, agilizando o processo e tornando a experiência mais agradável, simples e eficiente.

O casamento é a união entre duas pessoas, tornando-as assim uma só família, principal eixo da sociedade, afinal, é o que rege a base do sistema social, cultural e moral do país.

Segundo Campos (2018) apud Diniz (2001, p. 33): "O casamento é um vínculo jurídico entre o homem e a mulher que visa o auxílio mútuo material e espiritual, de modo que haja uma integração fisiopsíquica e a constituição de uma família legítima*".*

Por se tratar de algo tão importante na nossa sociedade, o casamento é oficializado através de uma cerimônia, que pode ser religiosa ou não, e que pode acompanhar um grande evento comemorativo, o qual demanda uma grande quantidade de cuidados para que este momento único na vida do casal aconteça da melhor forma possível.

Para planejar uma celebração de casamento é preciso tempo e dedicação, por isso, pensando em facilitar a organização, muitos casais preferem contratar de terceiros para realização da cerimônia de casamento. Diante disso, um dos serviços essenciais para o grande dia é ter uma equipe competente de cerimonialistas.

O cerimonialista de casamento tem um papel essencial na formação e realização do casamento. Na etapa que antecede a cerimônia, ele cuidará dos orçamentos com fornecedores, ajudará a definir o cardápio do buffet, decoração do local, escolha da recepção, contratação do fotógrafo, iluminação, DJ, banda e toda a parte administrativa também ficará a seu encargo, contratos com fornecedores, escolha dos funcionários que trabalharão no dia do casamento, etc. No dia do casamento, ele deverá conferir se todos estão cumprindo suas funções, se estão dentro do tempo e se tudo está como foi planejado.

É responsabilidade do cerimonialista cuidar do casamento e dos noivos, desde quando foi contratado até o final da festa, para que saia tudo como planejado.

De acordo com o artigo publicado pelo site *Yes I Do* (2016)

> O atendimento pode ser desde uma Assessoria Completa, na qual o cerimonialista apresenta de três a quatro opções de fornecedores, agenda as visitas e acompanha os noivos, ou Parcial com reuniões periódicas, orientações e indicações, mas quem fica à frente das buscas e tomadas de decisões são os noivos, até a mais conhecida, que é o Cerimonial do Dia.

Guia de casamento (2001), descreve que a responsabilidade do cerimonialista para que a o planejamento e a organização da cerimônia de casamento seja um sucesso, evitando que ocorra problemas devido falhas de comunicação entre as partes, e com isso acarretando um atraso de fechamento de negócios.

O objetivo deste trabalho é, portanto, apresentar desenvolvimento de um aplicativo capaz de auxiliar e agilizar a organização de um casamento para noivos e cerimonialistas.

Foram usadas várias ferramentas para gerar a documentação do projeto: BPMN (*Business Process Model Notation*), Diagramas UML (*Unified Modeling Language*). Já para o desenvolvimento de interface do usuário foram utilizados a biblioteca do React Native, para desenvolvimento da interface do sistema Node Js, e para Sistema Gerenciador de Banco de Dados (SGBD) utilizado (SQlite).

O segundo capítulo apresenta os conceitos teóricos acerca do trabalho, o terceiro apresenta um tema específico da engenharia de software sobre testes aplicados no projeto, o quarto trata das questões de organização de uma cerimônia de casamento, o quinto capítulo fala sobre análise de requisitos, o sexto mostra como foi o desenvolvimento do projeto, o sétimo mostra os resultados da implementação de todo sistema e o oitavo é finalizado com a conclusão do projeto.

2 Referencial teórico

Neste capítulo serão abordados temas relacionados a engenharia de software empregues no desenvolvimento de aplicativos para dispositivos móveis, a importância da organização e planejamento de um casamento e os erros cometidos ao se planejar uma cerimônia de casamento, testes e validação de software, análise de requisitos de software, como JavaScript pode ser responsável tanto pela criação do *back-end*, com Node JS, e como também pode criar um sistema web, utilizando a biblioteca React, para desenvolvimento *mobile* o Dart utilizando a biblioteca Flutter, e o gerenciador de banco de dados SQLite.

2.1 Aplicativos em dispositivos móveis

Os dispositivos móveis como os tablets, smartphones, estão se tornando ferramentas essenciais no cotidiano das pessoas, muitas já não conseguem sair de casa sem seu aparelho, pois através dele é possível usar aplicativos para diversos fins, podendo fazer uma transação de banco ou até mesmo traduzir uma placa com uma língua estrangeira. Essa evolução causa um grande impacto que as empresas devem se adaptar. Hoje, elas precisam se reinventar e inovar todos os dias para atrair seus novos consumidores e manter os seus clientes.

De acordo com Rodriguez (2019), no ano de 1946, surgiu o *Electronic Numerical Integrator and Computer* (ENIAC), o primeiro computador do mundo. Foi criado por cientistas americanos para ser usado na 2º guerra mundial pelo exército. Porém, era um aparelho enorme e pesava em torno de 30 toneladas e tinha em média 180m². Este computador funcionou por dez anos, seu processamento era feito por meio de cartões perfurados, foi utilizado para resolver grandes cálculos complexos, e na época ele resolveu mais cálculos que qualquer outra pessoa teria resolvido. Desde então a tecnologia veio evoluindo, aumentando o poder de processamento dos computadores e reduzindo o seu tamanho até chegarmos nos smartphones que utilizamos hoje.

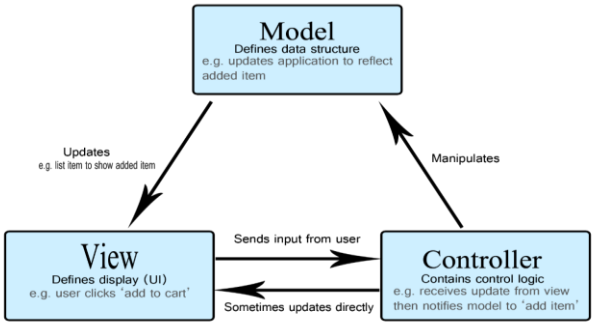
Definição.net (2018), software é um conjunto de instruções de componentes lógicos que controlam o funcionamento de um computador ou dispositivo móvel. É um termo para descrever aplicativos, programas de computadores, sistemas embarcados a fim de dizer o que uma máquina tem que fazer. Resumindo, todo programa no computador, televisão, smartphone, vídeo game é um software.

Rodriguez (2019), descreve que em 1994, foi combinada a telefonia celular com a computação, neste momento surgiu o IBM Simon, criado pela empresa americana IBM. Mas o grande diferencial foi a criação das lojas online de aplicativos, as mais utilizadas são a Google Play Store e a App Store, que vai depender do sistema operacional de cada dispositivo para serem utilizados, sendo a Google Play Store para dispositivos que possuem sistema Android e App Store para sistemas que possuem IOS.

2.2 React

De acordo com Pereira e Petrucelli apud Kostrzewa e Pandit (2018), o React foi criado em 2011 pela equipe de desenvolvedores do Facebook, a fim de oferecer uma solução devido a um constante problema devido a um componente inserido na interface de usuário. Devido a este problema, desenvolveram uma biblioteca em que o objetivo inicial era para reparar as mudanças visuais na interface de usuário, tornando-os mais rápidos e simples. Conforme mostra a camada visual da arquitetura Modelo-Visão-Controlador (MVC), ilustrada na figura 1:

Figura 1: MVC



Fonte: MDN web docs, 2005.

Segundo MDN web Doc (2005), MVC é um padrão de estrutura de software utilizado para desenvolver interface de usuários, que na maioria das vezes será utilizado para um projeto que possua uma grande quantidade de páginas. O MVC pode ser dividido em três partes e descritas da seguinte forma:

Modelo: que gerencia os dados coletados e a lógica de programação.

Visão: que lida com o layout e exibição.

Controlador: direciona os comandos para o modelo e exibe as visões correspondentes.

Existem muitas características do React, que foram utilizadas em vários projetos, como por exemplo, projetos derivados de dispositivo móveis o React Native. Porém, uma das características mais marcantes do React é o JSX, já que no desenvolvimento do React não é utilizado diretamente o JavaScript para estruturar a interface de usuário.

JSX é uma linguagem baseadas em TAGS, porém não utiliza a mesma semântica do HTML. É uma ampliação de sintaxe do JavaScript, que pode ser lembrado com uma linguagem de *template*, mas que usa todo potencial do JavaScript, produzindo “fundamentos” do React.

2.3 Node Js

Conforme descrito no site da Node, o sistema Node JS foi criado no ano de 2009, e foi desenvolvido pela Netscape, é um ambiente de tempo de execução JavaScript, uma ferramenta que pode ser usada para quase qualquer tipo de programa de código aberto e multiplataforma. Ele realiza o mecanismo JavaScript V8, que é a essência do Google Chrome, fazendo que o Node JS seja capaz de executar tarefas do lado do servidor.

> A principal característica que diferencia o Node.JS de outras tecnologias, como PHP, Java, C#, é o fato de sua execução ser *singlethread*. Ou seja, apenas um thread é responsável por executar o código Javascript da aplicação, enquanto que nas outras linguagens a execução é *multi-thread*. (LENON, 2018)

Um aplicativo desenvolvido em Node JS é aplicado em um único método sem criar uma conexão para cada requisição. Ainda oferece um kit de fonte assíncronas em suas bibliotecas padrão que impossibilita o bloqueio do código JavaScript.

Segundo Lenon (2018), uma da maior vantagem de usar o Node JS, é por conta do seu gerenciador de pacotes, o NPM (*Node Package Manager*), sendo o maior repositório do mundo. Sendo um do seu maior pacote reconhecido o Express.js, um *framework* voltado para criação de aplicações web.

Outra de suas vantagens também é a leveza, pois desenvolver uma aplicação que não tenha muitos recursos computacionais, utilizando uma ferramenta específica, o ganho de velocidade no *deploy*, pode ser muito eficaz e mais ágil.

2.4 Flutter

O Flutter é um *framework*[1] de múltiplas plataformas para codificar aplicativos no Android, IOS e em Desktop com uma performance nativa. Sua linguagem padrão é o Dart, que foi criada pelo Google no ano de 2011 que tinha objetivo inicial concorrer com o TypeScript.

De acordo com Abranches (2018), Dart é uma linguagem sucinta e orientada a objetos, sendo muito semelhante a linguagens com Swift, C#, Java e JS. O Dart é adequado para desenvolvimento *mobile* por ter um ótimo desempenho seja no desenvolvimento quanto em produção, sustentando a compilação JIT(*Just in Time*[2]), que permite que o Flutter recompile o código enquanto o aplicativo está sendo utilizado, tornado um ciclo de desenvolvimento muito rápido, e a compilação AOT (*Ahead of time*[3]), as bibliotecas e funções usadas pelo código do aplicativo são compilados no código nativo de cada plataforma, deixando perfeito para projetar compilações, iniciando o código nativo rapidamente.

Segundo Magalhãs (2019), o Flutter é divido em 3 camadas, a primeira é sua *engine*[4], é a responsável por todo o *framework* Flutter escrito em Dart. A segunda parte se trata do core do Flutter, que cuida de sistemas específicos e a terceira parte é o seu grande diferencial, sua *engine* gráfica (OpenGL ES).

2.5 SQLite

O projeto do SQLite foi criado em maio de 2000, desenvolvido por D. Richard Hipp, é uma biblioteca em linguagem C que implementa um banco de dados SQL embutido, rápido, independente, de alta confiabilidade e recursos completos. Programas que usam a biblioteca SQLite podem ter acesso a banco de dados SQL sem executar um processo SGBD separado.

O site oficial do SQLite (2020), apresenta suas vantagens, por ser um banco de dados embutido, de fácil manuseio e fácil implementação. É um sistema indicado para sistemas básicos onde não ocorrem muitas requisições, ou seja, um sistema manuseado por poucas pessoas.

Hoje, o SQLite está implementado em todos os celulares e em alguns computadores além de aplicativos que pessoas utilizam diariamente. O Código-fonte do SQLite é de domínio público e pode ser usado gratuitamente por todos para qualquer proposito.

De forma geral, diferente de sistemas cliente e servidor como MySQL ou Oracle, o SQLite lê e grava diretamente em arquivos de discos comuns, como se fosse um hd.

---

[1] É um conjunto de bibliotecas ou componentes utilizados para desenvolver uma base, onde será construída uma aplicação.

[2] Na hora certa.

[3] Antes do tempo.

[4] É uma ferramenta que une todos os arquivos e bibliotecas que o sistema vai precisar.

3. Testes e validação de Software

O processo para um desenvolvimento de software abrange inúmeras atividades, apesar de mesmo aplicando vários métodos e técnicas erros no produto final podem ocorrer. Atividades de garantia de qualidade de software empregadas no processo de desenvolvimento, com objetivo de diminuir a quantidade de erros e riscos agregados.

Segundo Barbosa et al. (2000), o teste de produto de software contém quatro etapas que é o planejamento de testes, projeto de caso de testes, execução e validação dos resultados de teste, sendo aplicadas em todo o processo de desenvolvimento do software, consistindo em três fases de teste: de unidade, integração e sistema.

Existem muitas formas de se testar um software, porém com as técnicas apropriadas, pois elas são classificadas de acordo o início das informações aplicadas para estabelecer os requisitos de software.

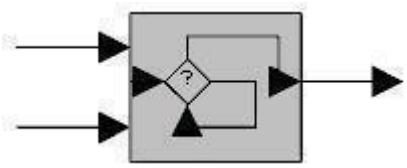
3.1 Teste Estrutural (ou teste de caixa branca)

A técnica que analisa o procedimento interno do componente de software (figura 2).

"Essa técnica trabalha diretamente sobre o código fonte do componente de software para avaliar aspectos tais como: teste de condição, teste de fluxo de dados, teste de ciclos e teste de caminhos lógicos (PRESSMAN, 2005)."

De acordo com a publicação do site ResearchGate (2008), essa técnica é recomendada para níveis de teste e teste de integração cujo critério é ficar por conta dos desenvolvedores de software, que conhecem o código fonte e podem planejar melhor os casos de testes.

Figura 2: Teste caixa branca



Fonte: Ladeira, 2014.

3.2 Teste Funcional (ou teste de caixa preta)

A técnica de teste funcional é tratada como se fosse uma caixa preta, não discorre do comportamento interno (Figura 3).

Coloca a entrada dos dados fornecidos, realiza o teste e obtém-se o resultado comparando com um resultado esperado, havendo sucesso se o resultado for idêntico.

"O componente de software a ser testado pode ser um método, uma função interna, um programa, um componente, um conjunto de programas e/ou

componentes ou mesmo uma funcionalidade. A técnica de teste funcional é aplicável a todos os níveis de teste (PRESSMAN, 2005).”

Figura 3: Teste caixa preta

Fonte: Ladeira, 2014.

4. A importância da organização e planejamento de casamento

De acordo com o artigo publicado por Arthur Rosa (2020) em seu próprio site, casamento envolve muitos fatores, várias etapas, desde o pedido de casamento tudo precisa ser bem pensado para que tudo ocorra da melhor forma, pois casamento exige tempo e um certa quantia em dinheiro, exigindo que envolva serviços de terceiros, como trajes, alianças, buffet, fotógrafos, viagens, e até mesmo a assessoria de casamento, que cuida e auxilia a o planejamento. Porém, este serviço depende muito da organização e comprometimento dos noivos para fechamento de contratos para dar continuidade nos procedimentos

Alden Souza (2019), diz que um dos erros mais comuns em casamentos é não iniciar o planejamento pela lista de convidados, pois através dela você busca todos os seus fornecedores, afinal, imagine contratar um salão de festa para 200 pessoas, e eventualmente seu casamento possui 350 convidados. Outro erro que o autor cita é não contratar uma assessoria de casamento e tentar fazer tudo por conta, já que a assessoria lida com eventos diariamente e sempre estão atentas aos detalhes que possam passar despercebidos.

É importante ter foco no fechamento de contrato com fornecedores, na listagem da ordem de importância de escolha do fornecedor, ou seja, deverá ser fechado primeiro, caso contrário o salão de festas não adiantará outros serviços necessários para este espaço.

5. Análise de requisitos

Desenvolver um software exige várias etapas, e é importante que esteja bem organizado e planejado para que não ocorra erros futuros, e a primeira fase para o desenvolvimento do software é a análise de requisitos que é feita com os analistas junto com seus clientes e *stakeholders*[5], ou seja, as partes interessadas do projeto

Como diz no artigo de Chyaromont (2016), requisitos de software são tudo aquilo que devemos pensar no que deve conter no software, todas suas funcionalidades, o que o sistema deve ou não deve ter, o que o sistema deve ou não

[5] Qualquer indivíduo ou organização que, de alguma forma, é impactado pelas ações de uma determinada empresa.

permitir. A verificação do requisito de software é de grande importância para assinar um documento com todos os requisitos para defesa jurídica de quem está contratando quanto para a empresa que está desenvolvendo o software.

Para aprofundar os conhecimentos acerca das necessidades e recursos necessários para a organização de uma cerimônia de casamento e por fim compreender os requisitos do aplicativo a ser desenvolvido neste projeto, foram realizadas entrevistas com uma empresa que trabalha há 7 anos como cerimonialista.

Segundo Sommerville (2011), uma das principais técnicas para realizar o levantamento de requisitos são as entrevistas com os interessados no software. Desta maneira, foram realizadas entrevistas abertas de forma que os *stakeholders* puderam expor seus principais problemas e dificuldades para o desenvolvimento de um aplicativo que pudesse suprir suas necessidades.

6. Desenvolvimento

Neste capítulo, mostraremos como o sistema de forma geral foi desenvolvido, como foi implementado o *back-end,* o *front-end* e o aplicativo, explicando as principais bibliotecas utilizadas em cada etapa, mostrando seu funcionamento, apresentando as linhas de códigos e em seguida mostrando os resultados. Também, será comentado sobre serviços externos como AWS (Amazon) e GitHub e como influenciaram no projeto.

6.1 Back-end

Para a construção do *back-end*, como demonstrado anteriormente, foi utilizado JavaScript, mais precisamente Node JS. Como apresentado na Figura 4, para a configuração do servidor, foi utilizado o *cors*, para dar acesso a qualquer dispositivo, o *express*, que fornece recursos para o desenvolvimento do *back-end*, e o *router*, que como mostrado na Figura 5, será responsável pelas rotas e criação de funções do *back-end*.

Figura 4: Configurações iniciais do back-end

```
const express = require('express');
const app = express();

const routes = require('./routes')
const cors = require('cors');


app.use(cors())
app.use(express.json());
app.use(routes);
app.listen(3333);
```

Fonte: Autores, 2020.

Figura 5: Arquivo de rotas back-end

```
const express = require('express');
const multer = require('multer');

const multerConfig = require('./config/multer')
const routes = express.Router();
const knex = require('../knexfile')

routes.get('/admin', async (req, res) => {
  const admin = await knex('cerimonialista').select('id').where('email', req.body.email);

  return res.json(admin)
});

routes.get('/client', async (req, res) => {
  const client = await knex('cliente').select('*');

  return res.json(client);
});
```

Fonte: Autores, 2020.

Duas bibliotecas essenciais para o projeto que também aparecem na Figura 5 são: Knex e Multer. O Knex é responsável pela criação e manipulação do banco de dados. Com ela, é possível escrever os comandos para utilização do banco de dados em formato JavaScript. A Figura 6 mostra como o Knex foi configurado, a Figura 7 mostra como a tabela do banco de dados foi criada e a Figura 8 apresenta como foi inserido dados no banco.

Figura 6: Configuração do arquivo knexfile.js

```
const path = require('path');


var knex = require('knex')(
  {
  client: 'sqlite3',
  connection: {
    filename: path.resolve(__dirname, 'src', 'database', 'database.sqlite')
  },
  migrations: {
    directory: path.resolve(__dirname, 'src', 'database', 'migrations')
  },
  seeds: {
    directory: path.resolve(__dirname, 'src', 'database', 'seeds')
  },
  useNullAsDefault: true
})


module.exports = knex;
```

Fonte: Autores, 2020.

Figura 7: Arquivo para criação da tabela “Cerimonialista”

```
const knex = require('knex');

exports.up = function(knex) {
  return knex.schema.createTable("cerimonialista", table => {
    table.increments("id").primary(),
    table.string('name').notNullable(),
    table.string('email').notNullable(),
    table.string('password').notNullable(),
    table.string('telefone').notNullable()
  })
}

exports.down = function(kenx) {
  return knex.schema.dropTable('cerimonialista');
}
```

Fonte: Autores, 2020.

Figura 8: Função para inserir dados na tabela “Cliente”

```
routes.post('/client', async (req, res) => {
  try {
    await knex('cliente').insert([{
      nome_noiva: req.body.nome_noiva,
      email_noiva: req.body.email_noiva,
      cpf_noiva: req.body.cpf_noiva,
      telefone_noiva: req.body.telefone_noiva,
      nome_noivo: req.body.nome_noivo,
      email_noivo: req.body.email_noivo,
      cpf_noivo: req.body.cpf_noivo,
      password: req.body.password,
      data_casamento: req.body.data_casamento,
      cerimonialista_id: req.body.cerimonialista_id
    }]);
    return res.json({ 'message': 'Cadastro concluido' });
  } catch (e) {
    return res.status(400).json({ 'message': 'Dados Inválidos' });
  }
},
);
```

Fonte: Autores, 2020.

O Multer foi essencial para o projeto, já que a partir dele, sistema consegue armazenar imagens e, simultaneamente armazenando na nuvem com os serviços da AWS, além disso, gera um link para acessá-las. Conforme a Figura 9, o Multer foi configurado para enviar arquivos para o servidor da AWS e na Figura 10, como foi utilizado para o realizar a tarefa e quais dados necessários para salvar os arquivos.

Figura 9: Configuração do Multer no projeto

```
const storageTypes = {
  local: multer.diskStorage({
    destination: (req, file, cb) => {
      cb(null, path.resolve(__dirname, "..", "..", "tmp", "uploads"));
    },
    filename: (req, file, cb) => {
      crypto.randomBytes(16, (err, hash) => {
        if (err) cb(err);

        file.key = `${hash.toString("hex")}-${file.originalname}`;

        cb(null, file.key);
      });
    }
  }),
  s3: multerS3({
    s3: new aws.S3(),
    bucket: 'uploadtcchenrique',
    contentType: multerS3.AUTO_CONTENT_TYPE,
    acl: "public-read",
    key: (req, file, cb) => {
      crypto.randomBytes(16, (err, hash) => {
        if (err) cb(err);

        const fileName = `${hash.toString("hex")}-${file.originalname}`;

        cb(null, fileName);
      });
    }
  })
};
```

Fonte: Autores, 2020.

Figura 10: Função para inserir dados na tabela “Images”

```
routes.post("/image", multer(multerConfig).single('file'), async (req, res) => {
  const { originalname: name_image, size, key, location: url = '' } = req.file;
  console.log(req.file)
  console.log(req.body.idPhoto)

  const image = await knex('images').insert([{
    name_image,
    size,
    key,
    url,
    photo_id: req.body.idPhoto,
  }
  ])

  return res.json(image)
})
```

Fonte: Autores, 2020.

6.2 Front-End

Ainda mantendo JavaScript para a criação do sistema, foi utilizado também para o desenvolvimento web do projeto. E para isso, a biblioteca optada para o desenvolvimento, foi o React Js. Com ela, é possível criar elementos HTML com os conceitos de JavaScript.

Por Exemplo, como é possível observar na Figura 11, responsável pela

criação do menu de navegação, podemos ver a utilização de *tags* como em HTML porém chamada em uma função como Js, aonde AsideMenu é um componente estilizado criado em cima de um aside e o Menu Link, criado em cima de um nav.

Figura 11: Criação do menu de navegação

```
import { AsideMenu, MenuLink, LinkCustom } from './styles'
import React from 'react'

export default props =>
  <AsideMenu className="menu-area">
    <MenuLink className="menu">
      <LinkCustom to="/index">Início</LinkCustom>
    </MenuLink>
    <MenuLink className="menu">
      <LinkCustom to="/createuser">Criar Usuário</LinkCustom>
    </MenuLink>
    <MenuLink className="menu">
      <LinkCustom to="/createplace">Criar Local</LinkCustom>
    </MenuLink>
    <MenuLink className="menu">
      <LinkCustom to="/local">Locais</LinkCustom>
    </MenuLink>
  </AsideMenu>
```

Fonte: Os autores, 2020.

Para a estilização do sistema web, foi utilizado *Styled Component*, com ele é possível estilizar componente escrevendo igual CSS, como demonstrado na Figura 12.

Figura 12: Estilização do *header*

```
import styled from 'styled-components';

export const HeaderCustom = styled.header`
  background-color: #FFF;
  padding: 0px 15px;
  overflow: hidden;
  white-space: nowrap;
  box-shadow: var(--shadow);
`;

export const H1 = styled.h1`
  font - size: 1.8em;
`
```

Fonte: Os autores (2020).

Também foi utilizado o React-bootstrap, uma biblioteca que fornece componentes já estilizáveis, responsivo e de fácil implementação. Na Figura 13, é possível identificar como utilizar os componentes para mostrar uma tabela.

Figura 13: Tela Criação de tabela utilizando React-Bootstrap

```
render() {
  return (
    <Table striped bordered hover >
      <thead>
        <tr>
          <th>Id</th>
          <th>Noiva</th>
          <th>Noivo</th>
          <th>Data do casório</th>
          <th>Numero de Convidados</th>
          <th>Ações</th>
        </tr>
      </thead>
      <tbody>
        {this.props.client.map(item => this.renderColumn(item))}
      </tbody>
    </Table>
  );
  }
}
```

Fonte: Autores, 2020.

Para consumo de API, foi utilizado o Axios, uma biblioteca que permite fazer requisições HTTP. Utilizado em vários sistemas pela facilidade de uso. Nas Figuras 14 e 15, é mostrado como fazer uma requisição utilizando GET e POST respectivamente

Figura 14: Utilização do Axios para consumir a dados da tabela

```
componentDidMount = async () => {
  const clients = await axios.get('http://localhost:3333/client')
  return this.setState({ client: clients.data });
}
```

Fonte: Autores, 2020.

Figura 15: Utilização do Axios para enviar dados para o *back-end*

```
handleClick = async () => {

  try {
    const data = await Axios({
      method: 'post',
      url: 'http://localhost:3333/photo',
      data: {
        name_companha: this.state.nome,
        desc: this.state.desc,
        telefone: this.state.fone,
      }
    })
    alert(`Companhia Criada com sucesso`);
    this.processUpload(data.data[0]);
    return this.props.history.push('/')
  }
  catch (e) {
    alert('Dados Inválidos')
  }
}
```

Fonte: Autores, 2020.

6.3 - Mobile

Diferente do *back-end* e do *front-end*, para o desenvolvimento *mobile*, foi escolhido a linguagem Dart, mais precisamente a biblioteca Flutter. Desenvolvimento pratico, rápido e por ser uma biblioteca de desenvolvimento hibrida. Como mostrado na figura 16, onde foi desenvolvido a tela de *login*, percebe-se como é de fácil desenvolvimento.

Para consumir Api, foi utilizado o HTTP, uma biblioteca para fazer requisições HTTP de maneira fácil e que também pode ser utilizado em sistemas web e desktop, além de *mobile*. Na figura 17, mostrado como é feito uma requisição POST no sistema, para verificar se o usuário é valido.

Figura 16: Código para criação da tela de login do aplicativo

```
return Scaffold(
  body: Container(
    margin: EdgeInsets.symmetric(horizontal: 20),
    child: ListView(
      children: <Widget>[
        Container(
            alignment: Alignment.center,
            child: Image.asset(
              'assets/images/logo_black.png',
              width: 240,
            )),
        Input(
          placeHolder: 'E-mail',
          marginBottom: 20,
          controller: _controllerEmail,
        ),
        Input(
          placeHolder: 'Senha',
          pass: true,
          marginBottom: 30,
          controller: _controllerPass,
        ),
```

Fonte: Autores, 2020

Figura 17:Utilização do Http para enviar dados para o back-end

```
Future<User> login({String email, String pass}) async {

final http.Response response = await http.post('http://192.168.15.200:3333/login',
    headers: <String, String>{
  'Content-Type': 'application/json; charset=UTF-8',
},
    body: jsonEncode(<String, String>{'email': email, 'pass': pass}));

if (response.statusCode == 200) {

  return User.fromJson(json.decode(response.body));
} else {
  throw Exception('Usuario Invalido');
}
```

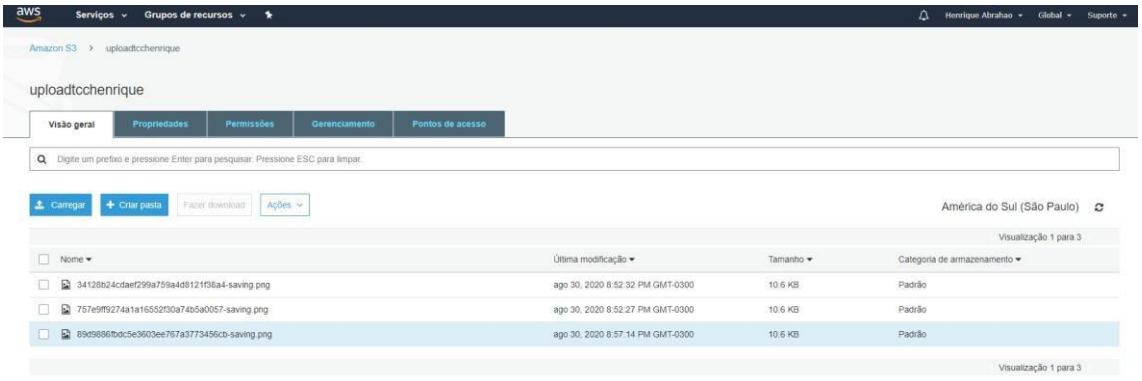
Fonte: Autores, 2020.

6.4 Serviços Externos

Foram utilizados 2 serviços externos para auxílio do desenvolvimento. São eles: GitHub, o qual é responsável por armazenar os códigos da aplicação na nuvem e pode ser utilizado para controlar as versões do sistema conforme o desenvolvimento, já que, como o projeto possui muitas funcionalidades, é essencial dispor de um versionamento com o objetivo de evitar problemas futuros causando atraso por falta de organização ao desenvolver o sistema, por exemplo. E AWS-S3, um serviço da AMAZON, que auxilia no armazenamento de objetos na nuvem, e como o sistema possui a funcionalidade de *uploads* de imagens, o AWS se tornou importante para o sistema em geral.

Na figura 18, mostrada a interface do serviço S3 e como ficam as imagens após serem enviadas.

Figura 18: Tela do AWS para visualizar dados salvos



Fonte: Autores, 2020.

6.5 Teste de caixa preta

Como citado no capítulo 3, executar testes é essencial para obter um produto final com boa qualidade, pois, mesmo executando as tarefas, ainda pode acontecer alguns erros, então para amenizar os erros foram executados os testes funcionais, conhecidos também como “teste de caixa preta”.

Como apresentado na figura 19, foi realizado o teste para verificar a entrada de cadastro de usuário ou fornecedor, onde é colocado caractere de tipo numérico é gerado uma mensagem de erro ou quando deixa o campo vazio e tenta criar, é também gerado a mensagem de erro.

Figura 19: Teste – Cadastrar

Artefato a ser avaliado		
Cadastrar usuário ou fornecedor	Entrada	Saída
	1	Campo inválido
	nulo	Para cadastrar usuário preencha o campo com *
	usuario@email.com	Senha temporária gerada com sucesso!

Fonte: Autores, 2020.

A figura 20, apresenta o teste efetuado na tela de *login* de usuário, caso seja seu primeiro acesso, como ele recebe uma senha temporária, o usuário deve inserir uma senha de no mínimo 6 dígitos, e caso contenha caracteres inválidos ou tenha campo nulo, o sistema emite uma mensagem de erro, não permitindo a criação da nova senha.

Figura 20: Teste – Primeiro login

Artefato a ser avaliado			
Efetuar primeiro login	Entrada A	Entrada B	Saída
	usuario@email.com	1	Usuário ou senha incorreto.
	usuarioemail.com	##1548884	Usuário ou senha incorreto, verifique e tente novamente.
	nulo	1548884	Para efetuar seu login, preencha todos campos.
	usuario@email.com	1548884	Primeiro login efetuado com sucesso!

Fonte: Autores, 2020.

Um outro exemplo de teste executado, é compartilhado na figura 21, sendo um pouco diferente na questão de entrada, pois este o usuário tem que selecionar um tipo de imagem para alterar seu perfil, e caso a imagem que ele selecione não seja no formato PNG (*Portable Network Graphics*[6]) ou JPEG (*Joint Photographic Experts Group*)[7] ou a entrada de nome seja nulo, o sistema emite uma mensagem de alerta.

Figura 21: Teste – Editar perfil

Artefato a ser avaliado			
Editar perfil	Entrada A	Entrada B	Saída
	Clarice	Carregar foto em formato Gif	Erro! Escolha um arquivo de formato adequado.
	nulo	Carregar foto PNG	Erro! Campo vazio.
	Marcelo	Carregar foto JPG	Perfil alterado com sucesso
	Joyce	Carregar foto PNG	Perfil alterado com sucesso

[6] Gráficos portáteis de rede

[7] Nome do grupo responsável por criar este método (método de compreensão de imagens)

Fonte: Autores, 2020.

7. Resultados

Neste capítulo, após todo o desenvolvimento todo sistema, serão apresentado os resultados conforme as funcionalidades necessárias e *layout* padronizado a fim de facilitar navegabilidade e utilização do sistema.

7.1 Web

Na figura 22, Tela de *login é* responsável pelo cerimonialista ter acesso a camada de administração do sistema.

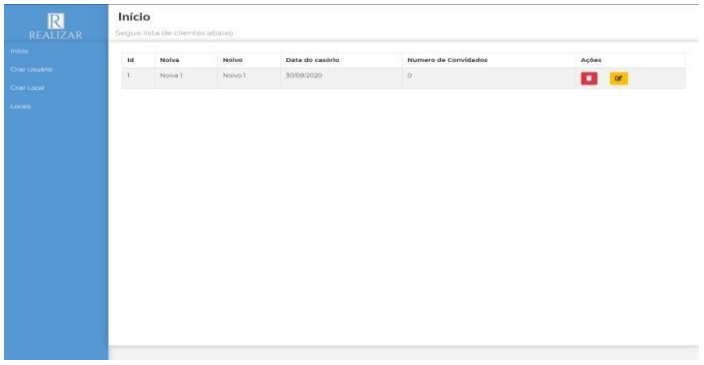
Tela *home* do sistema, figura 23, inicial do site e nela pode ver alguns dados essenciais dos noivos cadastrados no sistema, podendo alterar e excluir se necessário.

Na Tela para criação de usuário, figura 24, a partir dela é possível criar usuários para os noivos terem acesso ao aplicativo.

Figura 22: Tela Login

Fonte: Autores, 2020.

Figura 23: Tela Home

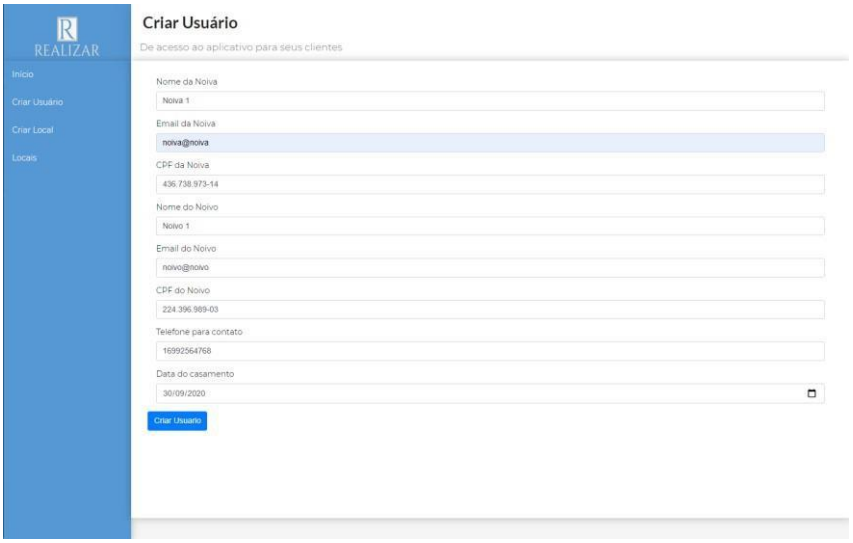


Fonte: Autores, 2020.

A figura 25, apresenta a Tela de criação de serviço com dados alternativos dependendo do serviço, tem também a finalidade de inserir locais/empresas, a quais os noivos poderão consultá-las a fim de agendar datas para o casamento.

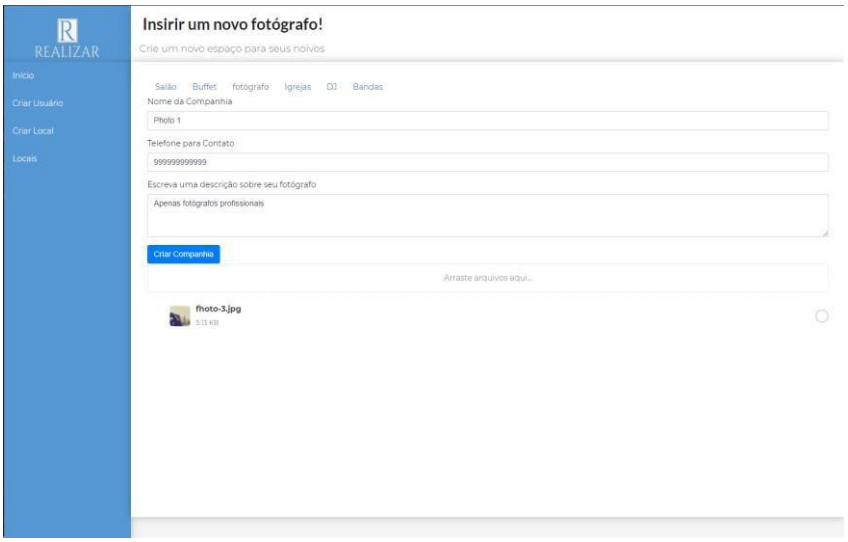
Listar serviços, figura 26, tela responsável por consultar serviços cadastrados podendo editá-los ou deletá-los se necessários.

Figura 24: Tela para criação de usuário



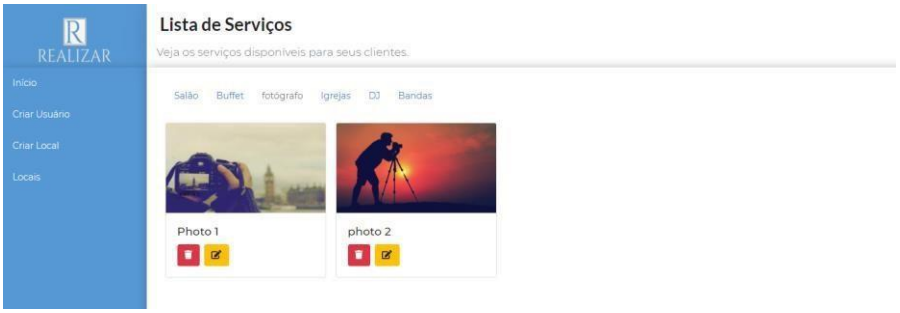
Fonte: Autores, 2020.

Figura 25: Tela de Criação de Serviço



Fonte: Autores, 2020.

Figura 26: Listar serviços



Fonte: Autores, 2020.

7.2 Mobile

Seguindo padronização do sistema web, o sistema *mobile*, também possui uma tela de *login* simples, conforme a figura 27.

Figura 27: Tela Login Mobile

Fonte: Autores, 2020.

Tela de introdução, conforme as figuras 28, 29 e 30, texto referente a boas-vindas para os noivos que estão utilizando o aplicativo pela primeira vez.

A seguinte é tela home do aplicativo, a partir dela, é possível ter acesso a todas as telas do aplicativo, nela é possível verificar o tempo para a data escolhida para a cerimônia no momento do cadastro, é possível visualizar nas figuras 31 e 32.

Figura 28: Tela Introdução 1

Figura 29: Tela introdução 2

Fonte: Autores, 2020.

Figura 30: Tela Introdução 3

Fonte: Autores, 2020.

Figura 31: Tela Home Mobile

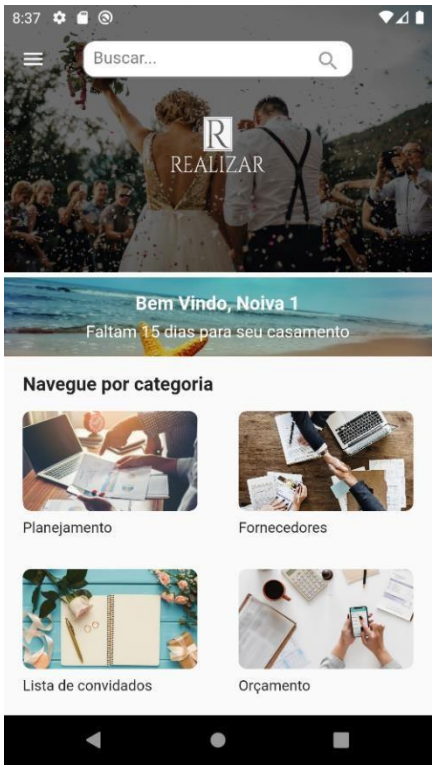
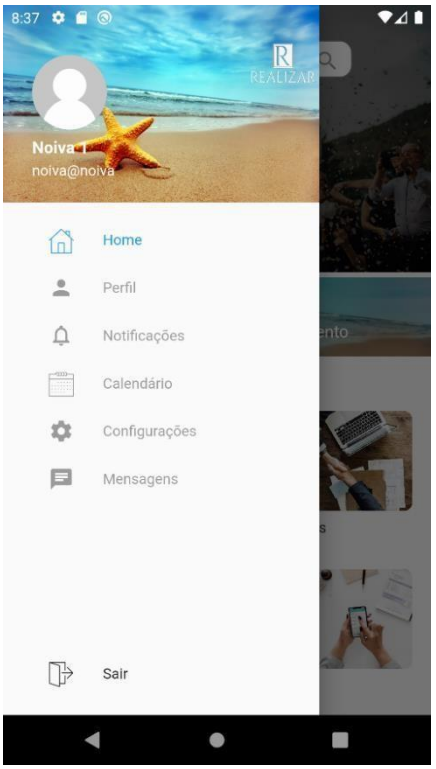


Figura 32: Drawer navigation



Fonte: Autores, 2020.

Na tela de orçamento, em que os noivos conseguem visualizar os valores estimados para cada setor do casamento, mostrando porcentagem necessária para cada um deles, onde, de acordo com a figura 33, onde os noivos colocariam os valores a quais eles desejam gastar na cerimônia, e nas figuras 34 e 35 mostrariam os valores para cada setor

A próxima é a tela de planejamento, de acordo com as figuras 36 e 37. Nessa tela, os noivos poderão planejar e ver a lista de tarefas ainda necessárias para ser feita até a data da celebração

Conforme a figura 38, tela para gerar lista de convidados, é possível colocar quem são os convidados dos noivos, podendo selecionar padrinhos e madrinhas.

Figura 33: Tela de orçamento

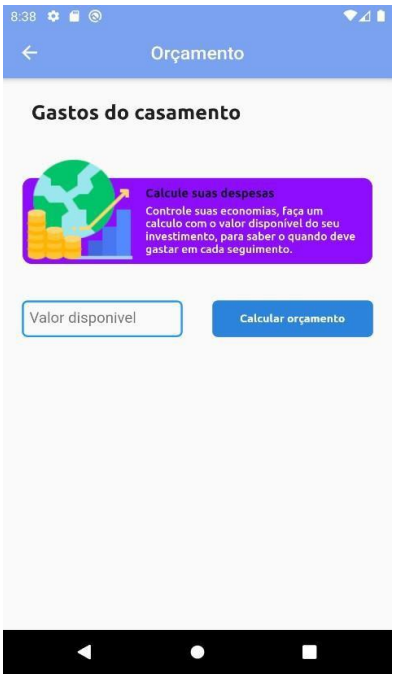


Figura 34: Telas de valores 1

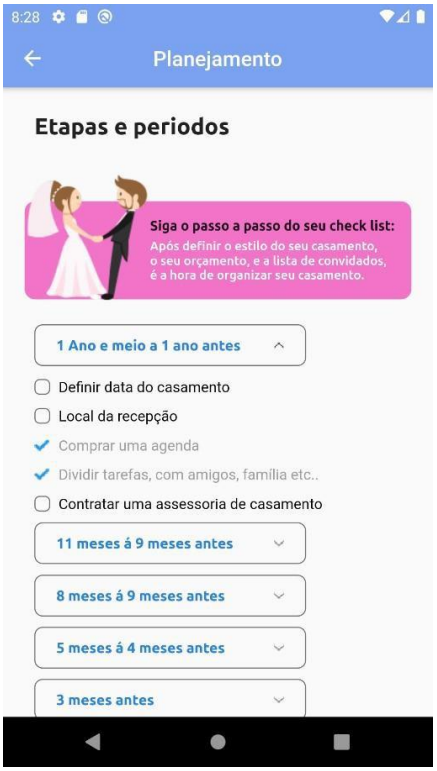
Fonte: Autores, 2020.

Figura 35: Telas de valores 2

Fonte: Autores, 2020.

Figura 36: Tela de planejamento 1

Figura 37: Tela de planejamento 2



Fonte: Autores, 2020.

Figura 38: Tela de lista de convidados

Fonte: Autores, 2020.

7.3 Validação de campos

Neste capítulo serão apresentados alguns exemplos de validação de campos no sistema, para impedir que dados sejam enviados de maneira equivocada, gerando mais segurança para o sistema, impedindo que o funcionamento seja prejudicado, atrapalhando a experiência do usuário e impedindo problemas futuros para a cerimonialista.

A partir da tela de criação de usuário, com a necessidade de inserção de CPF e e-mail, é necessário que tenha o máximo de verificação possível, a fim de evitar que CPF inválido seja inserido, ou até mesmo que e-mail sem “@” ou “.” sejam salvos no banco. Para isso foi criado as validações, como demonstrado na figura 42 e 43.

Na figura 44, é possível ver o momento em que essas validações são feitas, mas especificamente, no momento de criação do usuário, no *front-end*, onde é impedido de mandar CPF e e-mail para o *back-end*, travando a função de cadastrar usuário.

Figura 42: Validação CPF

```
export function validaCpf(val) {
  console.log('entrou')
  if (val.length == 11) {
    var cpf = val.trim();

    var v1 = 0;
    var v2 = 0;
    var aux = false;

    for (var i = 1; cpf.length > i; i++) {
      if (cpf[i - 1] != cpf[i]) {
        aux = true;
      }
    }

    if (aux == false) {
      return false;
    }

    for (var i = 0, p = 10; (cpf.length - 2) > i; i++, p--) {
      v1 += cpf[i] * p;
    }
```

Fonte: Autores, 2020.

Figura 43: Validação e-mail

```
export function verificarEmail(email) {
  if (email == ""
    || email.indexOf('@') == -1
    || email.indexOf('.') == -1) {
    return false;
  }
  return true;
}
```

Fonte: Autores, 2020.

Figura 44: Verificação antes da criação de usuário

```
CreateUsers = async () => {

  const { nome_noiva, nome_noivo, cpf_noiva, cpf_noivo, email_noiva, email_noivo, telefone, data } = this.state;

  if (!nome_noiva || !nome_noivo || !cpf_noiva || !cpf_noivo || !email_noiva || !email_noivo || !telefone || !data)
   return alert('Dados em branco')

    const validatorCpf = validaCpf(cpf_noiva) && validaCpf(cpf_noivo);
    const validatorEmail = verificarEmail(email_noivo) && verificarEmail(email_noiva);


  if (!validatorCpf) return alert('CPF Inválido');
  if(!validatorEmail) return alert('Email Inválido');
```

Fonte: Autores, 2020.

8. Considerações finais

A ideia começou a partir da necessidade de uma cerimonialista em possuir um aplicativo que a auxiliasse em seus planejamentos e ajudasse seus clientes a organizar melhor seus planos e objetivos para atingir uma cerimônia autêntica.

São pouquíssimos aplicativos no mercado que possuem a finalidade de auxiliar esses eventos, dando suporte a cerimonialista a partir dos problemas relatados. Assim, surgiu a ideia desafiadora de um serviço novo para esse mercado, com diferentes funcionalidades conforme apresentado no desenvolvimento e resultado final.

O desenvolvimento desse projeto envolvendo tantas ferramentas para auxiliar na criação da documentação tanto quanto no desenvolvimento do software, foi uma experiência muito gratificante. Antes havia apenas uma noção conceitual de tudo, sobre a tecnologia que foram usadas e as possibilidades que elas viabilizam. A primeira versão do projeto foi pensada a partir da criação de apenas um aplicativo para auxiliar o cerimonialista no acompanhamento das atividades dos noivos, porém conforme o desenvolvimento, foram surgindo novas ideias, e adicionando novas funcionalidade e características diferentes, uma delas foi a criação do web, para o cerimonialista integrar fornecedores e noivos.

Outra funcionalidade que foi adicionada também ao decorrer do projeto foi, a “Tela de detalhes do profissional”, a ideia principal dessa tela era apenas mostrar os dados do fornecedor, então adicionamos outros campos com funcionalidades essenciais, como apresentado no capítulo de “Resultados”.

Desenvolver um software para casamento, envolve muitos detalhes, e pode ser pensado através de diversas funcionalidades para auxiliar o cerimonialista e os noivos, em vários aspectos, seja no sentido de ergonomia ou de acessibilidade. Além disso, é possível cogitar a integração de novas funcionalidades futuras para este projeto.

O produto final alcançou as expectativas deste artigo, uma vez que ele supera todas as expectativas esperadas, embora a lamentação de não ter cumprido com todas as funcionalidades do aplicativo, como por exemplo “chat”, apesar de ser importante para o sistema, não era o foco principal de imediato, tendo em vista que a necessidade da criação da ferramenta era para atender outros fins.

Outro fator que dificultou o desenvolvimento, foi a utilização de linguagens diferentes para criar o site e o aplicativo, apesar que isso nos proporcionou um grande conhecimento de ótimas ferramentas para projetos futuros e formação profissional, a dificuldade de interagir um sistema com JavaScript e Dart ainda é alto.

Concluímos que este projeto tem muito a expandir, por ser desenvolvido a uma área que demanda muitos cuidados, recursos e organização, além disso é importante ressaltar que a cerimônia de casamento passou por diversas mudanças e continua a evoluir se tornando um evento que exige novos elementos sempre, por isso exibe uma grande extensão de possíveis funcionalidades que com certeza serão desenvolvidas futuramente.

REFERÊNCIAS

A BRIEF history of Node.js. Node. Disponível em: https://nodejs.dev/learn/a-briefhistory-of-nodejs. Acesso em 10 abr. 2020

ABRAHÃO, Henrique de Paula. TCC Planejamento casamento. Github, 2020. Disponível em: https://github.com/henriqueabrahao/TCC_Planejamento_casamento.

ABRANCHES, Júnior. Conhecendo um pouco mais do Flutter. Disponível em: https://imasters.com.br/framework/conhecendo-um-pouco-mais-flutter. Acesso: 8 de maio 2020

BARBOSA, Ellen Francine et al. Introdução ao Teste de Software. Disponível em: http://www.duguay.com.br/uploads/arquivos/apostilaUSP_Teste_de_Software.pdf. Acesso: 8 de maio 2020.

BARBOSA, Ellen Francine et al. Introdução ao Teste de Software. Disponível em:http://www.duguay.com.br/uploads/arquivos/apostilaUSP_Teste_de_Software.pdf . Acesso: 8 de maio. 2020.

CAMPOS, Fernando Papa de. Conceito de Casamento. Dubbio, 2018. Disponível em: https://www.dubbio.com.br/artigo/599-conceito-de-casamento. Acesso em: 5 de abr. 2020.

CHYAROMONT, Allan. O que são requisitos de software, e como facilitam o planejamento do seu projeto de computação. Fluxo: blog de engenharia, 2016. Disponível em: https://fluxoconsultoria.poli.ufrj.br/blog/tecnologiainformacao/requisitos-desoftware/?gclid=CjwKCAjwnK36BRBVEiwAsMT8WGiD8Nc9klKuP57wRDsLtwh7lgq LeAMHASEV8NqUYmwXg-h5vfturxoCS5YQAvD_BwE. Acesso: 6 de maio 2020.

GUIA de casamentos. A Importância da Assessoria de Casamento. Disponível em: https://www.guiadecasamento.com.br/planejamento/assessoriacerimonial/assessoria-de-casamento. Acesso: 8 de maio. 2020.

HOLANDA, Timóteo. Dart: primeiros passos com a linguagem. DISPONÍVEL EM: https://www.alura.com.br/conteudo/dart-primeiro-passos. Acesso: 07 de maio 2020

INTRODUZINDO JSX. React. Disponível em: https://ptbr.reactjs.org/docs/introducing-jsx.html. Acesso em 10 abr. 2020

LADEIRA. Luiz. Introdução a testes de Software, 2014. Disponível em:https://luizladeira.wordpress.com/2014/02/11/introducao-a-teste-desoftware/. Acesso em: 7 maio 2020

LENON. Node.js – O que é, como funciona e quais as vantagens. Disponível em: https://www.opus-software.com.br/node-js/. Acesso: 07 de maio. 2020.

MAGALHÃES, Túlio. Flutter: tudo sobre o queridinho do google. Disponível em: https://www.zup.com.br/blog/flutter. Acesso: 07 de maio 2020.

MDN web docs. MVC. Disponível em: https://developer.mozilla.org/enUS/docs/Glossary/MVC. Acesso em 10 abr. 2020

O QUE faz a cerimonialista de casamento. Yes I Do, 2016. Disponível em: https://www.yesidowedding.com.br/O-que-faz-a-cerimonialista-decasamento.html. Acesso em: 5 de abr. 2020.

PEREIRA, Anderson de Sousa; PETRUCELLI, Erick Eduardo. Análise das semelhanças e diferenças entre utilizar JSX ou JAVASSCRIPT puro ao construir interfaces com react. Disponível em: https://webcache.googleusercontent.com/search?q=cache:5pMuBqwDg0IJ:https://revista.fatectq.edu.br/index.php/interfacetecnologica/article/download/569/353/+&cd=1&hl=pt-BR&ct=clnk&gl=br. Acesso em 10 abr. 2020

RESEARCHGATE, Neto. Introdução a testes de software, 2008. Disponível em: https://www.researchgate.net/profile/Arilo_Neto/publication/266356473_Introducao_a_Teste_de_Software/links/5554ee6408ae6fd2d821ba3a/Introducao-aTeste-deSoftware.pdf Acesso em: 7 maio 2020

RODRIGUEZ, Maitê. A história dos aplicativos: Quem usa e quem vive de desenvolver. Tribuna da Imprensa Livre, 2019. Disponível em: https://tribunadaimprensalivre.com/a-historia-dos-aplicativos-quem-usa-equem-vivede-desenvolver/. Acesso em: 6 abr. 2020

ROSA, Arthur. Como organizar um casamento. Arthur Rosa, 2020. Disponível em: https://arthurrosa.com/como-organizar-um-casamento/. Acesso em: 15 maio 2020

SOMMERVILLE, Ian. Engenharia de Software. 9o Edição. 2011.

SOUZA, Alden. Os principais erros das noivas na organização do seu casamento. Alden Souza, 2019. Disponível em: https://www.aldensouza.com.br/post/27571-naorganizacao-do-seu-casamento. Acesso em: 15 maio 2020

WHAT Is SQLite?. SQLite, 2020. Disponível em: https://www.sqlite.org/index.html 2020. Acesso: 8 de maio. 2020.

YASMIN. O que é software? Para que serve? Quais os tipos? Disponível em: https://definicao.net/software/https://definicao.net/software/. Acesso: 6 de maio 2020.