UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

WILLIAMBERG DE ALBUQUERQUE FERREIRA

AGENDEVC:
UM SISTEMA DE AGENDAMENTO PARA PRESTADORES E CONSUMIDORES DE SERVIÇOS DIVERSOS

CAMPINA GRANDE - PB

2024

WILLIAMBERG DE ALBUQUERQUE FERREIRA

AGENDEVC:
UM SISTEMA DE AGENDAMENTO PARA PRESTADORES E CONSUMIDORES DE SERVIÇOS DIVERSOS

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador : Tiago Lima Massoni

CAMPINA GRANDE - PB

2024

WILLIAMBERG DE ALBUQUERQUE FERREIRA

AGENDEVC:
UM SISTEMA DE AGENDAMENTO PARA PRESTADORES E CONSUMIDORES DE SERVIÇOS DIVERSOS

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA:

Tiago Lima Massoni

Orientador – UASC/CEEI/UFCG

Melina Mongiovi Cunha Lima Sabino

Examinador – UASC/CEEI/UFCG

Francisco Vilar Brasileiro

Professor da Disciplina TCC – UASC/CEEI/UFCG

Trabalho aprovado em: 15 de Maio de 2024.

CAMPINA GRANDE - PB

RESUMO

Gerenciar a agenda e fornecer informações ao cliente são tarefas que oneram muito o tempo do pequeno empreendedor, que, em muitos casos, gerencia e opera um negócio sozinho, precisando maximizar seu tempo produzindo para obter melhores resultados e manter o seu empreendimento em atividade. Este trabalho tem como objetivo propor o desenvolvimento de um sistema web que oferece ao empreendedor um meio de fornecer as principais informações sobre seu negócio e visualizar os serviços agendados. O sistema provê ao consumidor dos serviços prestados uma forma de visualizar as informações sobre um determinado serviço e efetuar o agendamento em horário disponível e conveniente para si, dispensando a necessidade de qualquer tipo de contato prévio com o prestador do serviço. A solução foi avaliada por 23 usuários, obtendo um índice de satisfação médio de 4.7 entre os dois grupos de usuários (prestadores de serviços e consumidores), constatando ser uma proposta eficaz na solução do problema.

AGENDEVC:
A BOOKING SYSTEM FOR SERVICE PROVIDERS AND CUSTOMERS

ABSTRACT

Schedule management and customer service are tasks that take off a good amount of the working hours from the hands of a small service provider, who in many cases, manages and runs the business on its own. Making more with the available time is crucial to be able to get better results and to keep the business up and running. This work has the goal of proposing the development of a web application which gives the entrepreneur a way to provide information about its business and to maintain a schedule of all the upcoming services and open spots available to its customers. On the customer side, the system offers a way to visualize the necessary information about a certain service and the possibility of booking them by the customer's availability and convenience, without any need of previous contact with the service provider. The solution was rated 4.7 out of 5 by 23 users divided in two separate groups: Service providers and customers. This satisfaction rating demonstrates the effectiveness of the proposed solution.

AgendeVc:
Um Sistema de Agendamento Para Prestadores e Consumidores de Serviços Diversos

Williamberg Ferreira
Universidade Federal de Campina Grande
williamberg.ferreira@ccc.ufcg.edu.br

Tiago Lima Massoni
Universidade Federal de Campina Grande
massoni@computacao.ufcg.edu.br

RESUMO

Gerenciar a agenda e fornecer informações ao cliente são tarefas que oneram muito o tempo do pequeno empreendedor, que, em muitos casos, gerencia e opera um negócio sozinho, precisando maximizar seu tempo produzindo para obter melhores resultados e manter o seu empreendimento em atividade. Este trabalho tem como objetivo propor o desenvolvimento de um sistema web que oferece ao empreendedor um meio de fornecer as principais informações sobre seu negócio e visualizar os serviços agendados. O sistema provê ao consumidor dos serviços prestados uma forma de visualizar as informações sobre um determinado serviço e efetuar o agendamento em horário disponível e conveniente para si, dispensando a necessidade de qualquer tipo de contato prévio com o prestador do serviço. A solução foi avaliada por 23 usuários, obtendo um índice de satisfação médio de 4.7 entre os dois grupos de usuários (prestadores de serviços e consumidores), constatando ser uma proposta eficaz na solução do problema.

Palavras-chave

Agendamento, serviços, sistema web.

Links importantes

https://github.com/william-ferreira/agendevc

https://forms.gle/m8DjjjDHmUV3vAai9

https://www.figma.com/file/8ryG0xmybQhyDnmyiZGMFM/AgendeVc

1. INTRODUÇÃO

O setor de serviços é um dos que mais cresce no Brasil. Apenas em 2022, o setor apresentou um crescimento de 8,3%[1]. Apenas nos últimos dois anos foram abertos 343 mil salões de beleza [2]. Muitos desses empreendimentos são geridos e operados por microempreendedores individuais (MEI) que sozinhos acumulam muitas funções, dentre elas, gerir a agenda do seu negócio, tarefa que demanda tempo e uma metodologia organizacional para executá-la de forma eficiente e livre de erros.

Em virtude do acúmulo de funções e a alta demanda de serviços, a administração da agenda de forma improvisada pode gerar dores de cabeça e problemas organizacionais no negócio que ocasionarão prejuízos ao microempreendedor, ou no mínimo, serão uma barreira que impedirá a progressão dos resultados. Como exemplo podemos ilustrar o caso de um barbeiro que presta atendimento ao cliente, gerencia a agenda e executa os serviços sozinho, sem o auxílio de ninguém. O tempo gasto com o gerenciamento de agenda (marcando e desmarcando serviços com clientes, passando valores de serviços, etc.) poderia ser melhor aproveitado de outras maneiras, com capacitação e implementação de melhorias em seu empreendimento, por exemplo.

Com a finalidade de resolver o 'problema crônico' mencionado anteriormente e auxiliar o empreendedor a melhorar seus resultados e a qualidade organizacional de sua operação, surge o AgendeVc, um sistema que disponibiliza a agenda do empreendedor para que seus próprios clientes agendem os serviços. Ou seja, o próprio consumidor obtém informações sobre os serviços e faz agendamentos sem qualquer tipo de contato prévio com o prestador dos serviços. Para tal, é necessário que o empreendedor forneça apenas algumas informações básicas a respeito do seu negócio, tais como: Horários e dias de funcionamento, tabelas de preço e tempo necessário para execução de cada um dos serviços oferecidos.

Posteriormente à essa fase inicial de configuração das informações do negócio, o sistema irá expor essas informações aos usuários interessados no serviço através de uma página, dando ao usuário a possibilidade de verificar os dias e horários disponíveis para a realização do serviço desejado, podendo também, por conta própria e sem nenhum contato prévio com o prestador de serviço, marcar o período em que deseja ser atendido.

Este documento propõe uma solução que 'remedie as dores' mencionadas anteriormente, através do desenvolvimento de uma aplicação web que fornece um painel de controle para o empreendedor (também chamado de 'prestador de serviços') e um painel exclusivo para agendamentos e consultas para o consumidor dos serviços ofertados.

A solução proposta foi avaliada por 23 usuários, sendo 3 deles prestadores de serviços e os outros 20 potenciais consumidores de serviços ofertados na plataforma, obtendo um índice de satisfação de 4.6 de 5 entre os prestadores de serviços, e 4.8 de 5 entre os usuários do tipo consumidor.

2. ARQUITETURA

Nesta seção estão descritas todas as decisões arquiteturais orquestradas para propor uma solução ao problema anteriormente apresentado. A proposta consiste em unir prestadores de serviços com seu público consumidor, inserindo o menor nível possível de fricção. Sendo assim, o objetivo é entregar uma solução simples e extremamente prática em sua utilização, sem funcionalidades ou

processos desnecessários que possam onerar ainda mais os usuários da aplicação.

Todas as escolhas aqui descritas visam garantir que a proposta seja adequada, satisfaça as necessidades de ambos os públicos e não adicione camadas extras de complexidade. Simultaneamente, o conjunto das decisões opera em favor da oferta de uma aplicação altamente disponível, escalável e com performance suficiente para assegurar o adequado funcionamento do sistema proposto.

2.1 Visão geral

A arquitetura do sistema divide-se em duas camadas: um cliente, responsável por ser a camada de interação com os usuários do sistema, e uma API[3] REST[4] sem servidor. Ambas as camadas estão hospedadas em serviços AWS (Amazon Web Services)[5], garantindo alta disponibilidade, elasticidade e escalabilidade sob demanda, além de um alto grau de confiabilidade e segurança.

O cliente está hospedado em uma instância de armazenamento AWS S3[6], que é um serviço de armazenamento de objetos. A arquitetura propõe o estabelecimento da comunicação entre as duas principais camadas do sistema através de uma API REST sem servidor que encapsula os seguintes serviços:

- AWS API Gateway[7]: Serviço responsável por permitir a criação, manutenção e monitoramento de APIs, fornecendo seus recursos de acordo com o tipo de API especificado.
- AWS Lambda[8]: Serviço de computação sem servidor que possibilita a execução de código sem a necessidade de gerenciar infraestrutura. Sem preocupações com provisionamento ou gerenciamento de servidores.
- AWS DynamoDB[9]: Banco de dados não relacional (de tipo chave-valor) auto escalável e de baixa latência. Fornece um modelo de dados flexível, alta disponibilidade e segurança dos dados.

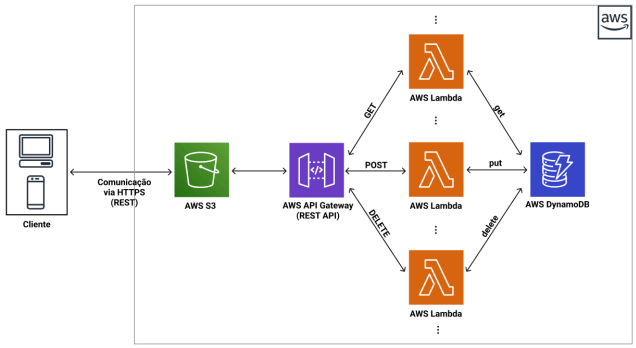

Figura 1: Arquitetura da solução

Nesta proposta arquitetural (figura 1), o cliente comunica-se com a camada lógica da aplicação através do API Gateway, que estabelece a comunicação via requisições HTTP, trafegando objetos JSON[10].

Por sua vez, o API Gateway executa funções no AWS Lambda de acordo com o recurso e tipo de requisição HTTP. O AWS Lambda é responsável por executar o código da API, sendo o único módulo do sistema que se comunica com a base de dados não relacional mantida no DynamoDB.

2.2 Cliente

O cliente da aplicação (Popularmente chamado de front-end) foi desenvolvido utilizando React[11], um framework JavaScript[12] popular na indústria para o desenvolvimento de aplicações web que permite a construção de interfaces de usuário combinando as tecnologias necessárias para desenvolvimento executável em navegadores de internet: JavaScript, HTML[13] e CSS[14]. O cliente da aplicação possui duas frentes distintas: Um painel de controle para usuários do tipo prestador de serviços e um pequeno conjunto de telas apropriadas para dispositivos móveis para usuários do tipo consumidor. Esta decisão visa proporcionar comodidade ao usuário que fará agendamentos na plataforma, não sendo necessária a instalação de nenhum tipo de aplicativo. Levando a decisão anteriormente mencionada em consideração, a proposta da solução envolve a construção de uma aplicação web progressiva *(Do inglês, PWA, Progressive Web App)*[15] por possibilitar o acesso da aplicação independente do tipo de dispositivo utilizado para tal.

Para garantir uma interface intuitiva e amigável ao usuário que ofereça uma boa experiência de usabilidade, um protótipo de todas as telas e componentes da aplicação foi planejado e desenvolvido na ferramenta de design e prototipagem Figma[16]. A ferramenta possui muitos recursos para o desenho de interfaces de usuário, além de fornecer inúmeros templates de componentes em virtude de sua vasta comunidade de usuários que disponibilizam esses conteúdos de forma gratuita.

A execução do protótipo foi fundamental para nortear o desenvolvimento do cliente da aplicação, possibilitando melhor visualização das funcionalidades do sistema a fim de analisar a sua usabilidade e fazer ajustes na interface ainda na fase de prototipagem. Esta decisão ajudou a mitigar a possibilidade de modificações após o início da implementação em código, tarefa bem mais onerosa quando comparada a modificar um simples protótipo.

Após a prototipagem, a proposta do front-end da aplicação foi desenvolvida de forma a promover a reutilização dos componentes e dos temas de cores da aplicação. Tornando o código extensível e aplicável em diferentes contextos com o intuito de acelerar o desenvolvimento e tornar o código modular e organizado. Camadas de serviços e modelos foram pensadas para, respectivamente, receber os dados da camada lógica da aplicação e tratá-los de maneira adequada para os utilizar entre os componentes do front-end.

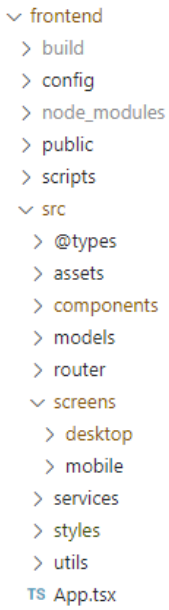

Figura 2 - Estrutura de diretórios do cliente

As telas da aplicação são organizadas entre desktop, que contém as telas para visualização em computador via navegador, e mobile, que visa encapsular as telas para visualização em dispositivos móveis, também via navegador de internet. De forma análoga estão organizados os componentes da aplicação, entre os que são específicos para computador, para celular, e os componentes que são comuns a ambos os tipos de dispositivos. A figura 2 apresenta em mais detalhes a estrutura de pastas do projeto de front-end da aplicação.

2.3 API sem servidor

A parte lógica da aplicação (popularmente chamada de back-end) constitui-se de uma API REST sem servidor que envolve três principais componentes: Um AWS API Gateway, múltiplas instâncias AWS Lambda, sendo uma para cada recurso especificado no gateway de API, e um banco de dados não relacional DynamoDB. Esta subseção visa fornecer alguns detalhes de implementação interessantes além da visão geral fornecida em uma seção anterior deste documento.

As arquiteturas sem servidor oferecem inúmeras vantagens ao desenvolvedor, sendo a principal delas a exclusão da preocupação com questões de infraestrutura. Além de trazer o foco das atividades a questões de desenvolvimento propriamente dito, soluções sem servidor na AWS particularmente possuem integração com todos os serviços a poucos cliques (ou comandos) de distância. Também é válido mencionar que em um cenário real de uma aplicação rodando em um ambiente de produção, os custos serão gerados por demanda. Ou seja, não será necessário pagar para manter uma infraestrutura de uma aplicação que não está sendo consumida. Você paga o que consome e com isso garante-se que não há desperdício de recursos [17]. Ou seja, uma instância AWS Lambda só gera custos enquanto estiver rodando. Para aplicações de propósitos mais simples como a solução aqui proposta, onde não há grande volume de dados e grandes cargas de processamento, uma arquitetura sem servidor é uma ótima escolha. Esta arquitetura também é adequada para a aplicação proposta, por garantir escalabilidade elástica, reduzindo e maximizando o provisionamento da aplicação de acordo com a demanda sem a necessidade de fazer esses ajustes manualmente. O próprio setor de serviços tem seus 'picos e vales' com relação a demanda, tornando a escalabilidade elástica uma característica ainda mais atrativa no contexto da aplicação.

Para estabelecer todos os serviços AWS utilizados de forma automática, o Serverless[18] framework foi utilizado por tornar isso possível de forma rápida e simples. Com apenas alguns comandos é possível criar e disponibilizar aplicações sem servidor na AWS com todos os serviços configurados e todas as comunicações necessárias já estabelecidas. Para tal, é necessário adicionar algumas informações em um arquivo de configuração gerado automaticamente após a criação de um projeto Serverless, tais como políticas de acesso a recursos e outros parâmetros importantes como a região AWS em que sua aplicação irá ser provisionada.

```yaml
org: agendevc
app: agendevc
service: agendevc-api
frameworkVersion: '3'

provider:
  name: aws
  runtime: nodejs18.x
  region: sa-east-1
  iamRoleStatements:
    - Effect: "Allow"
      Action:
        - "dynamodb:PutItem"
        - "dynamodb:Get^"
        - "dynamodb:Scan*"
        - "dynamodb:GetItem"
        - "dynamodb:UpdateItem"
        - "dynamodb:DeleteItem"
      Resource: arn:aws:dynamodb:${aws
```

Figura 3 - Trecho do arquivo de configuração serverless.yml

A figura 3 mostra algumas das configurações necessárias. A região escolhida para o provisionamento da aplicação foi a "*sa-east-1*" localizada em São Paulo. Ainda na mesma figura, estão descritas algumas permissões para manipular dados no DynamoDB. Todas as tabelas de dados, instâncias lambda e recursos no API gateway são estabelecidos de forma automática a partir das configurações presentes neste arquivo. No mesmo arquivo de configuração é necessário inserir informações sobre as funções que irão compor a API gerada automaticamente, como nome da função, tipo de método e caminho (rota) para realizar a chamada a uma função em específico (figura 4).

```
getAgendamentosById:
  handler: functions/agendamento/getAgendamentoById.getAgendamentoById
  events:
    - httpApi:
        path: /agendamento/{id}
        method: get
getAllAgendamentos:
  handler: functions/agendamento/getAllAgendamentos.getAllAgendamentos
  events:
    - httpApi:
        path: /getAgendamentos
        method: get
deleteAgendamentoById:
  handler: functions/agendamento/deleteAgendamentoById.deleteAgendamentoById
  events:
    - httpApi:
        path: /agendamento/{id}
        method: delete
```

Figura 4 - Definição de recursos a serem disponibilizados via API

O AWS API Gateway é a porta de entrada da camada lógica da aplicação, sendo o único serviço responsável por comunicar-se diretamente com o cliente (front-end) do sistema. A comunicação é feita através da disponibilização de recursos previamente definidos de acordo com a funcionalidade que cada um desses recursos deverá desempenhar, como um método HTTP GET ou POST, por exemplo (figura 5).

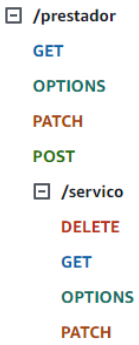

Figura 5 - Ilustração de recursos em um AWS API Gateway

As instâncias AWS Lambda serão responsáveis por executar através de funções todo o código referente a parte lógica da aplicação todas as vezes que uma chamada for feita a algum dos recursos disponíveis no API gateway. A figura 6 exemplifica de forma gráfica a integração de instâncias AWS Lambda e um API gateway.

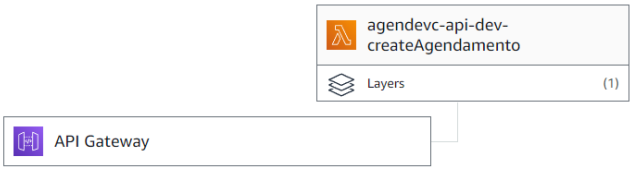

Figura 6 - Função AWS Lambda conectada a um AWS API Gateway

O código da camada lógica da aplicação foi desenvolvido em Node JS[19]. A escolha justifica-se por oferecer melhor desempenho em inicialização[20] (*Do inglês, cold start*) quando comparado a outras linguagens de programação, além de ser amplamente utilizado em arquiteturas sem servidor, facilitando a busca por materiais de orientação durante o desenvolvimento. Quanto à arquitetura proposta para o back-end da aplicação, o código foi todo organizado em funções, que são disponibilizadas uma a uma em uma respectiva instância AWS Lambda. A figura 7 mostra em detalhes a estrutura de diretórios do projeto de back-end da aplicação.

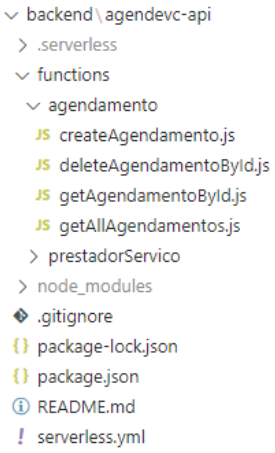

Figura 7 - Estrutura do back-end da aplicação

Uma tabela no DynamoDB foi utilizada para armazenar os dados da aplicação. Por se tratar de um banco de dados extremamente flexível, não foi necessária uma etapa de modelagem de dados e relacionamentos entre tabelas no ciclo de desenvolvimento da proposta de solução. Foi realizada uma breve análise dos tipos de dados que seriam armazenados na tabela, a fim de se estruturar esses dados de forma adequada e facilitar as abstrações necessárias durante o desenvolvimento. O banco de dados comunica-se apenas com as instâncias AWS Lambda, escrevendo, atualizando e removendo dados.

Por se tratar de um banco de dados não relacional, os dados são costumeiramente armazenados em forma de arquivos JSON. A figura 8 mostra um trecho da tabela de dados estruturada para a proposta da solução já populada com alguns itens.


```
{
 "PrestadorServicoId": "0fefcf5f-446d-4441-843a-8f6eccde0631",
 "AgendamentosPrestadorServico": [
  {
   "PrestadorServicoId": "0fefcf5f-446d-4441-843a-8f6eccde0631",
   "AgendamentoId": "f3b9a808-1d23-45d6-961c-883beff054f2",
   "ClienteId": "1",
   "Data": "2024-04-15",
   "FormaPagamentoId": "3",
   "Horario": "08:00",
   "ServicoId": "4"
  },
  {
   "PrestadorServicoId": "0fefcf5f-446d-4441-843a-8f6eccde0631",
   "AgendamentoId": "3ca3e0e1-d6cb-4603-9fb2-fd8457e0b94e",
   "ClienteId": "2",
   "Data": "2024-04-17",
   "FormaPagamentoId": "1",
   "Horario": "16:00",
   "ServicoId": "2"
  },
  {
```


Figura 8 - Visualização JSON da tabela de dados no DynamoDB

3. FUNCIONALIDADES

Nesta seção serão descritas as funcionalidades do sistema para os dois tipos de usuários disponíveis: prestadores de serviço e consumidores. Para exemplificar o uso do sistema, serão percorridos dois fluxos de uso, um para cada tipo de usuário, cobrindo os principais cenários de utilização da ferramenta.

3.1 Prestador de Serviços

O painel de controle do prestador de serviços contém três seções principais: *"Agenda"*, *"Meu Negócio"* e *"Meus Serviços"*, que encapsulam todas as funcionalidades do sistema. Há também um fluxo excepcional para cadastro do negócio que é acionado apenas no primeiro acesso ao sistema. Todas as funcionalidades serão descritas com imagens nas subseções a seguir.

3.1.1 Cadastro e acesso

Para cadastrar-se no sistema como um prestador de serviços, basta inserir informações básicas de cadastro: nome de usuário, e-mail e senha para acessar o sistema (figura 9).

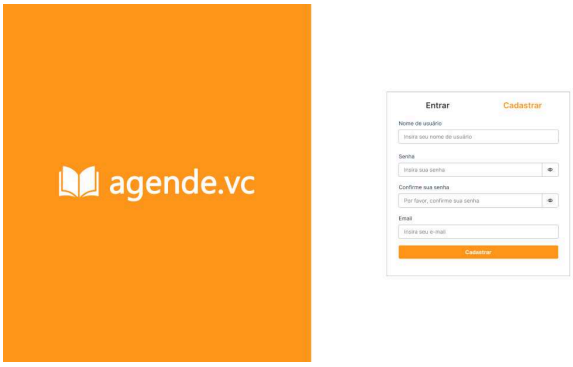

Figura 9 - Cadastro de usuário prestador de serviços

O acesso ao sistema é simplificado através da inserção de nome de usuário (ou e-mail) e senha de acesso.

3.1.2 Primeiro acesso: cadastro de negócio

Ao acessar o sistema pela primeira vez, o usuário prestador de serviços tem o auxílio de um fluxo de cadastro que visa cadastrar e configurar o negócio para deixá-lo em condições ideais para a utilização do sistema.

Na primeira tela (figura 10), o usuário irá inserir as informações básicas do seu negócio, sendo elas: Nome do negócio, telefone celular (*vinculado ao WhatsApp, visando funcionalidades futuras descritas na seção 4.3*) e descrição do negócio.

Figura 10 - Cadastro de negócio: informações básicas

Na segunda tela do assistente de cadastro (figura 11), o usuário irá definir os horários de funcionamento de seu empreendimento, podendo definir opções alternativas de horário para os fins de semana. Também é possível definir horários com um intervalo de inatividade separando dois blocos de horários, opção muito útil para negócios que fecham em horário de almoço, por exemplo.

Figura 11 - Cadastro de negócio: horários de funcionamento

Na terceira das quatro etapas de cadastro (figura 12), o usuário precisará escolher quais os métodos de pagamento que o seu empreendimento aceita, sendo as opções disponíveis: Pix, cartões de débito e crédito, dinheiro e cheques.

Figura 12 - Cadastro de negócio: formas de pagamento

Por fim, o usuário precisa cadastrar os serviços que serão disponibilizados para agendamento. As informações necessárias para o cadastro de um novo serviço são: Nome, valor, duração de execução e descrição do serviço (figura 13). Nesta versão da proposta de solução, a duração válida para os serviços está na unidade de horas, para simplificar questões de implementação, considerando o caráter do sistema como sendo um mínimo produto viável[21].

Figura 13 - Cadastro de negócio: Adicionar serviços

Ao concluir todos os passos de cadastro, o usuário é redirecionado para o painel do prestador de serviços e não será mais direcionado para este assistente de cadastro em seu próximo acesso ao sistema.

3.1.3 Minha agenda

A agenda é a seção principal do painel de controle (figura 14). Nesta seção, o prestador de serviços consegue visualizar todos os agendamentos da semana, podendo avançar ou retroceder semanalmente interagindo com os botões no canto superior direito, ou selecionar uma semana em específico através do calendário no canto superior esquerdo.

Figura 14 - Painel de controle: Minha Agenda

Ao clicar em um dos agendamentos, é possível expandir as informações daquele item em específico (Figura 15), tornando possível também a exclusão do agendamento por parte do prestador de serviços, sob confirmação do mesmo.

Figura 15 - Componente de informações de agendamento

3.1.4 Meu negócio

Na seção de *"Meu Negócio"*(figura 16) é possível visualizar as informações de cadastro descritas no item 3.1.2: Informações básicas, horários de funcionamento e formas de pagamento. É possível editar essas informações caso necessário, como remover ou adicionar uma nova forma de pagamento, por exemplo.

Figura 16 - Painel de controle: Meu Negócio

3.1.5 Meus serviços

Na seção de *"Meus Serviços"*(figura 17) é possível visualizar todos os serviços cadastrados, sendo também possível cadastrar novos serviços, além de editar ou excluir serviços existentes.

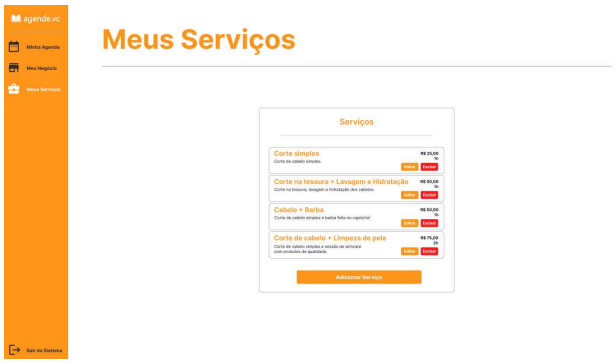

Figura 17 - Painel de controle: Meus Serviços

Por se tratar de uma ação crítica, ao tentar excluir um serviço o usuário recebe um pedido de confirmação para a execução da ação, exibindo mais uma vez as informações do serviço que está sendo supostamente removido da carta de serviços.

3.2 Consumidor

Pensando no maior nível de comodidade e menor resistência possível para utilização da ferramenta, as telas do usuário foram projetadas para serem visualizadas em dispositivos móveis. Aqui o objetivo principal é executar todas as ações necessárias ao consumidor com o menor número possível de telas e componentes, sem comprometer a experiência do usuário.

Nesta versão da proposta, o usuário executa apenas duas ações: agenda serviços e pode também consultar os agendamentos realizados caso deseje. Ambas as funcionalidades serão descritas com mais detalhes nas subseções a seguir.

Antes disso, é válido mencionar como os usuários terão acesso a uma agenda e menu de serviços de um determinado negócio.

Basicamente, os usuários irão acessar a plataforma através de um link nas redes sociais do prestador de serviços (figura 18) que irão o direcionar para a tela principal da aplicação móvel, onde o usuário terá a primeira vista os serviços oferecidos por aquele e somente aquele prestador de serviços. Ou seja, cada prestador de serviços terá um link próprio para o seu negócio.

Figura 18 - Exemplificando o acesso à plataforma para consumidores

3.2.1 Agendamento

Para agendar um serviço, o usuário irá trafegar por um total de quatro telas, realizando ações rápidas, além de ser direcionado para uma última tela que apenas exibe a confirmação do agendamento acompanhado de suas respectivas informações.

Na primeira tela (figura 19), o usuário irá selecionar o serviço desejado. Caso já tenha feito um agendamento anteriormente e desejar consultá-lo, estará disponível na mesma tela um componente que o direciona para o painel de consultas descrito na subseção a seguir.

Figura 19 - Agendamento 1 de 4: selecionar serviço

Na segunda tela (figura 20), o usuário vai escolher a data e o horário em que deseja agendar o serviço escolhido no passo anterior. Apenas os dias e horários disponíveis serão exibidos para escolha.

Figura 20 - Agendamento 2 de 4: selecionar data e horário

Na terceira tela (figura 21), o usuário irá inserir suas informações básicas: nome completo e número de celular (*vinculado ao WhatsApp, visando funcionalidades futuras descritas na seção 4.3*), além de selecionar a forma de pagamento desejada.

Figura 21 - Agendamento 3 de 4: Inserção dos dados pessoais e seleção da forma de pagamento

Por se tratar de uma ação que após efetuada não poderá ser desfeita por parte do consumidor, a quarta tela (figura 22) exibe novamente todas as informações sobre o agendamento para que o usuário possa visualizá-las mais uma vez antes de confirmar o

agendamento, podendo retornar às seções anteriores para corrigir informações caso necessário.

É válido mencionar que na versão atual da solução proposta apenas o prestador de serviços pode cancelar um agendamento. Para que um consumidor cancele um agendamento será necessário solicitar ao prestador do serviço agendado.

Figura 22 - Agendamento 4 de 4: Revisão das informações

Após efetuar um agendamento, o usuário é direcionado para uma tela de confirmação (figura 23), onde é possível também efetuar um novo agendamento caso o usuário deseje.

Figura 23 - Confirmação de agendamento

3.2.2 Consulta de agendamentos

Pensando na facilidade de utilização do sistema, o usuário do tipo consumidor não precisa instalar nenhum aplicativo, muito menos realizar algum tipo de cadastro para utilizar o sistema. Um painel apenas para consultas é uma solução rápida e prática para que o usuário possa visualizar todos os seus agendamentos realizados com o respectivo prestador de serviços. Para acessar o painel de "*Consultar Agendamentos*" (figura 24), nesta versão de proposta da solução, o usuário pode o fazer inserindo apenas o número de celular utilizado para realizar agendamentos na plataforma.

Figura 24 - Acesso ao painel de consultas de agendamentos do consumidor

Após inserir corretamente o número de seu telefone celular acompanhado de seu respectivo DDD, o usuário é direcionado a uma tela contendo todos os agendamentos realizados (figura 25).

Na versão atual da solução proposta, o usuário consegue visualizar apenas os agendamentos efetuados com o prestador de serviços proprietário do link de acesso em que o usuário consumidor está trafegando. Por exemplo, caso o usuário tenha acessado o painel de consultas através do link proprietário do estabelecimento "Joãozinho Barbeiro", o usuário conseguirá visualizar apenas os agendamentos efetuados com o mesmo prestador de serviços.

Figura 25 - Painel para consultas de agendamentos

4. AVALIAÇÃO

Nesta seção está descrita a metodologia utilizada para avaliação da solução proposta, com o objetivo de coletar dados e concluir o nível de satisfação do público alvo com relação à ferramenta.

A solução proposta foi avaliada por três prestadores de serviço e vinte potenciais consumidores de serviços através da plataforma.

Os respondentes foram recrutados e orientados individualmente. Para a realização das avaliações e coleta de dados das respostas dos avaliadores do sistema, uma variação do PSSUQ[22] (*Do inglês, Post-Study System Usability Questionnaire*) foi replicada em um formulário. Este questionário de usabilidade de sistemas é largamente utilizado e prova ser eficaz na coleta de informações que podem ser úteis para aprimorar um sistema. Foram utilizadas dez questões que abordam aspectos de usabilidade do sistema, desde o nível de satisfação quanto a sua interface gráfica, até a eficácia das informações fornecidas pelo sistema para a execução de tarefas.

Por se tratarem de dois subsistemas com conjuntos distintos de funcionalidades, o mesmo questionário foi aplicado para os dois grupos de usuários, prestadores de serviços e consumidores, porém, utilizamos formulários distintos para analisar as respostas separadamente de acordo com o grupo respondente.

Para cada uma das questões, utilizamos a escala Likert[23] com cinco pontos, representando de um a cinco, respectivamente, “discordo totalmente” a “concordo totalmente”.

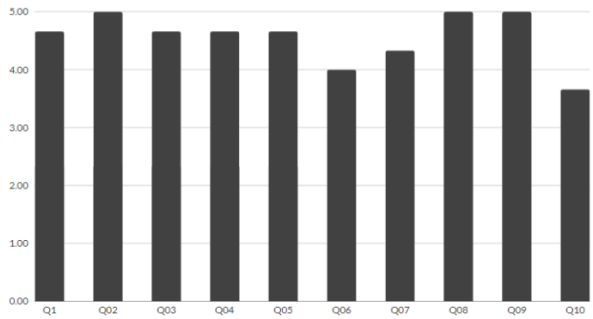

Figura 26 - Resultados do questionário baseado no PSSUQ com respostas dos prestadores de serviço

Os resultados exibidos na figura 26 representam a média dos valores obtidos em cada item do questionário pelos três prestadores de serviço que responderam ao formulário. Foi obtida uma média geral de satisfação de 4.6 de 5, o que representa um alto nível de satisfação com relação ao sistema em uma perspectiva do prestador de serviços.

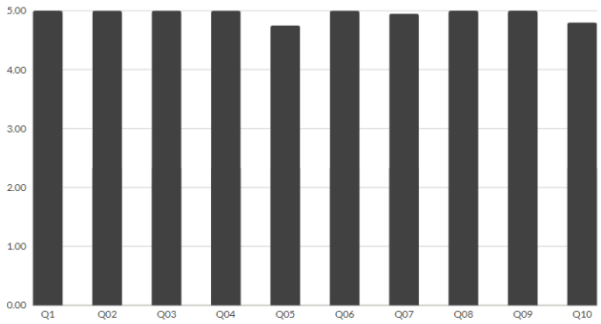

Figura 27 - Resultados do questionário baseado no PSSUQ com respostas de potenciais consumidores

Na figura 27, podemos observar as médias dos resultados de cada uma das questões, desta vez na perspectiva de potenciais consumidores de serviços oferecidos na plataforma. A média geral obtida foi de 4.8 de 5, indicativo de altíssima satisfação dos consumidores com relação à solução proposta.

Concluindo sobre os dados coletados nas perguntas do questionário, ambos os tipos de usuários sentem-se extremamente satisfeitos e confortáveis com a interface de usuário e o nível de dificuldade de utilização do sistema. No entanto, os prestadores de serviços acreditam que um pouco mais de informação nas telas possam tornar a sua experiência melhor ao utilizar a ferramenta.

Retornando à análise da avaliação dos prestadores de serviços, é possível concluir que a ferramenta precisa de mais funcionalidades para suprir de forma mais abrangente suas necessidades, sendo este o ponto pior avaliado entre o público alvo, obtendo 3.66 de 5. Ainda que possa ser considerada uma avaliação positiva, a discrepância nos resultados desta questão em comparação às outras, indica um ponto de atenção que será tratado na subseção de trabalhos futuros deste mesmo documento.

5. EXPERIÊNCIAS

Nesta seção está descrito o processo de desenvolvimento da solução proposta, bem como os desafios encontrados ao longo do processo e as perspectivas para o futuro da plataforma AgendeVc.

5.1 Processo de desenvolvimento

Por se tratar de um projeto com os requisitos bem definidos, o processo de desenvolvimento utilizado foi baseado no modelo de castata, que divide o projeto em fases bem delimitadas e as executa sequencialmente.

A seguir, cada uma das etapas, na ordem em que foram executadas:

- Concepção da ideia;
- Levantamento de requisitos;
- Protótipo de interface gráfica;
- Definição da arquitetura da solução;
- Estruturação dos dados;
- Desenvolvimento do cliente da aplicação (Frontend);;
- Desenvolvimento de lógica da aplicação (Backend);
- Avaliação da solução;

Apesar de se tratar de uma proposta de solução, todos os itens acima foram trabalhados com o objetivo de validar em sua totalidade o que foi proposto, constatando a viabilidade de execução do projeto em um cenário real.

5.2 Desafios

Ao mencionar os desafios enfrentados no desenvolvimento da solução, é possível enxergá-los em duas frentes, como sendo dois tipos de obstáculos: tecnológicos e de produto.

Em uma perspectiva de produto, é sempre tarefa delicada entender com precisão os requisitos do cliente. Construir apenas o necessário e que irá de fato entregar valor na solução do problema é uma missão difícil e que requer bastante análise, além de comunicação com as partes interessadas. A solução foi planejada e posteriormente refinada com sugestões de alguns prestadores de serviços que foram considerados potenciais usuários da plataforma, atingindo um significativo grau de satisfação, como descrito no item anterior deste documento.

No ponto de vista técnico, o principal desafio foi a elaboração de uma arquitetura sem servidor. É uma abordagem diferente, relativamente nova em relação ao clássico modelo

cliente-servidor, o que torna o desenvolvimento bastante desafiador em diversos momentos. Após análise de várias estratégias e modelos de arquitetura, empiricamente foi encontrado um modelo que supriu as expectativas com relação ao que foi proposto.

A estruturação dos dados em um banco de dados não relacional também foi um desafio considerável no desenvolvimento, foi necessário analisar estratégias adequadas de indexação e estruturas de dados utilizadas em alguns dados internos à tabela principal, com o objetivo de evitar operações desnecessárias que possam ocasionar custos adicionais e perda de performance. Dentre as estratégias adotadas, é válido mencionar a utilização de índices secundários globais (*GSI, do inglês, Global Secondary Indexes*)[24] para algumas estruturas internas como os agendamentos em uma tabela de prestador de serviços, com o objetivo de evitar operações desnecessárias e habilitar formas mais diretas de requisitar alguns dados que são acessados frequentemente, mantendo um bom nível de performance no acesso aos dados.

A prototipagem de uma interface de usuário que proporcione uma experiência de usuário decente foi outro desafio observado no processo de concepção da solução. Arquitetar a ordem das telas, os tipos de componentes utilizados e o posicionamento de todos os elementos gráficos de forma que não confunda o usuário é uma tarefa complexa que requer muito planejamento e análise. Alcançar o resultado final foi possível através da coleta de sugestões de melhoria por parte de potenciais usuários da plataforma e o aprimoramento contínuo da interface de usuário com base nessas sugestões.

5.3 TRABALHOS FUTUROS

A plataforma AgendeVc visa tornar o mais simples possível o gerenciamento da agenda de um pequeno empreendedor, simplificando o seu dia a dia, além de agilizar ao máximo a tarefa de realizar agendamentos e obter as informações necessárias sobre um determinado serviço, facilitando também todo o processo para o consumidor. Atualmente, a proposta e o desenvolvimento realizado tem como alvo atingir o critério mínimo de funcionamento da ferramenta, tornando-a um produto mínimo viável.

A próxima fase de desenvolvimento consistirá em implementar algumas funcionalidades sugeridas na sessão de avaliação da solução proposta que foram consideradas boas adições ao conjunto de funcionalidades oferecidas. Sendo:

- Módulo de disparo de mensagens via WhatsApp que visam notificar tanto o prestador de serviços quanto o consumidor sobre seus respectivos agendamentos;
- Integração com a API do Google Calendar que visa replicar a agenda do prestador de serviços no seu calendário atrelado à sua conta Google;
- Atendimento automatizado via WhatsApp através de um chatbot construído com um LLM[25] (*Large Language Model*) personalizado para cada negócio;
- Funcionalidade para adicionar mais de um colaborador no mesmo negócio, com uma agenda para cada, com o objetivo de atender negócios onde há mais de um profissional prestando serviços;
- Módulo de pagamento integrado à aplicação que possibilite o consumidor pagar antecipadamente pelos serviços agendados;
- Painel de finanças simplificado para o prestador de serviços, que o possibilite obter uma visão geral de seus ganhos;
- Redesign da interface de usuário com objetivo de torná-la ainda mais fácil e intuitiva em sua utilização, proporcionando uma melhor experiência para o usuário.

O conjunto de novas funcionalidades anteriormente descritas tem grande potencial de inovação e valor agregado no projeto, expandindo ainda mais o poder da plataforma AgendeVc ao seu público alvo, empoderando o micro e pequeno empreendedor com uma tecnologia que entrega disponibilidade, performance e funcionalidades de forma rápida e descomplicada.

6. REFERÊNCIAS

[1] Exame, "Setor de serviços atinge patamar recorde e fecha 2022 com alta de 8,3%", disponível em: https://exame.com/economia/setor-de-servicos-atinge-patamar-recorde-e-fecha-2022-com-alta-de-83/

[2] Terra, "Em 2 anos, Brasil abriu 343 mil salões de beleza", disponível em: https://www.terra.com.br/economia/em-2-anos-brasil-abriu-343-mil-saloes-de-beleza,42910cf3702f60fd6efbef4757ecd705iy9mylt3.html

[3] AWS, "What is an API?", disponível em: https://aws.amazon.com/pt/what-is/api/

[4] Red Hat, "What is a REST API?", disponível em: https://www.redhat.com/pt-br/topics/api/what-is-a-rest-api

[5] AWS, "What is AWS?", disponível em: https://aws.amazon.com/pt/what-is-aws/

[6] Amazon S3 Developer Guide, disponível em: https://docs.aws.amazon.com/pt_br/AmazonS3/latest/userguide/Welcome.html

[7] AWS, "Amazon API Gateway", disponível em: https://aws.amazon.com/pt/api-gateway/

[8] AWS, Lambda Developer Guide, disponível em: https://docs.aws.amazon.com/pt_br/lambda/latest/dg/welcome.html

[9] Amazon DynamoDB Developer Guide: "Introduction", disponível em: https://docs.aws.amazon.com/pt_br/amazondynamodb/latest/developerguide/Introduction.html

[10] Json Official Website: "Introducing JSON", disponível em: https://www.json.org/json-en.html

[11] React: "React homepage", disponível em: https://pt-br.legacy.reactjs.org/

[12] Geeks For Geeks, “Introduction to JavaScript”, disponível em: https://www.geeksforgeeks.org/introduction-to-javascript/

[13] HTML, “What Is HTML?”, disponível em: https://html.com/#What_is_HTML

[14] MDN Web Docs, “CSS: Cascading Style Sheets”, disponível em: https://developer.mozilla.org/en-US/docs/Web/CSS

[15] Microsoft Learn, “Overview of Progressive Web Apps (PWAs)”, disponível em:

https://learn.microsoft.com/en-us/microsoft-edge/progressive-web-apps-chromium/

[16] UX Academy, “What is Figma and its Advantages?”, disponível em: https://myuxacademy.com/what-is-figma/

[17] Lumigo, “Save Money On Serverless: common costly mistakes and how to avoid them”, disponível em: https://lumigo.io/blog/save-money-on-serverless-common-costly-mistakes-and-how-to-avoid-them/

[18] Serverless: "How It works", disponível em: https://www.serverless.com/#How-It-works

[19] Node.js home page, disponível em: https://nodejs.org/en/about

[20] BetterDev, "AWS Lambda runtime: Node.js vs Python", disponível em: https://betterdev.blog/aws-lambda-runtime-nodejs-python/

[21] InfoMoney, "O que é e como fazer um MVP (Mínimo Produto Viável)", disponível em: https://www.infomoney.com.br/guias/o-que-e-como-fazer-mvp-produto-viavel-minimo/

[22] UI/UX Trend, "PSSUQ (Post-Study System Usability Questionnaire)", disponível em: https://uiuxtrend.com/pssuq-post-study-system-usability-questionnaire/

[23] Survey Monkey, “O que é uma escala Likert?”, disponível em: https://pt.surveymonkey.com/mp/likert-scale/

[24] AWS, DynamoDB Developer Guide: "GSI (Global Secondary Index)", disponível em: https://docs.aws.amazon.com/pt_br/amazondynamodb/latest/developerguide/GSI.html

[25] Triggo AI Blog, "O que é Large Language Model (LLM)?", disponível em: https://triggo.ai/blog/o-que-e-large-language-models-ou-llm/