

UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Cassiano Andrei Dias Da Silveira Schneider

**ARQUITETURA PARA DISTRIBUIÇÃO DE CONTEÚDOS
CONSIDERANDO O CONTEXTO DO USUÁRIO**

Santa Maria, RS
2019

Cassiano Andrei Dias Da Silveira Schneider

**ARQUITETURA PARA DISTRIBUIÇÃO DE CONTEÚDOS CONSIDERANDO O
CONTEXTO DO USUÁRIO**

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de **Bacharel em Ciência da Computação**.

ORIENTADOR: Prof. Celio Trois

TG 466
Santa Maria, RS
2019

Cassiano Andrei Dias Da Silveira Schneider

**ARQUITETURA PARA DISTRIBUIÇÃO DE CONTEÚDOS CONSIDERANDO O
CONTEXTO DO USUÁRIO**

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de **Bacharel em Ciência da Computação**.

Aprovado em 4 de dezembro de 2019:

Celio Trois, Dr. (UFSM)
(Presidente/Orientador)

João Vicente Ferreira Lima, Dr. (UFSM)

João Carlos Damasceno Lima, Dr. (UFSM)

Santa Maria, RS
2019

RESUMO

ARQUITETURA PARA DISTRIBUIÇÃO DE CONTEÚDOS CONSIDERANDO O CONTEXTO DO USUÁRIO

AUTOR: Cassiano Andrei Dias Da Silveira Schneider

ORIENTADOR: Celio Trois

Nos últimos anos, houve um aumento considerável no número de aplicativos instalados nos celulares das pessoas, exigindo mais da capacidade de armazenamento de cada dispositivo e tirando o espaço das empresas na área de aplicativos para outros ramos como rede sociais, entretenimento e aplicativos de mensagens. Este projeto objetiva melhorar a experiência de relacionamento entre empresa e cliente através da tecnologia, implementando uma arquitetura que possibilita pessoas, sem conhecimento técnico, construírem websites PWA, que apresentam a mesma experiência de um aplicativo, entretanto sem depender de instalação. Ademais, os conteúdos tornam-se sensíveis ao contexto, fornecendo aos usuários informações precisas. Essas características eliminam dificuldades que empresas têm de emplacar digitalmente em seus clientes.

Palavras-chave: Metodologia de Desenvolvimento. Desenvolvimento Web. Progressive Web Apps. Framework Django. Aplicativos Sensíveis ao Contexto. Distribuição de conteúdos.

ABSTRACT

CONTENT DISTRIBUTION ARCHITECTURE CONSIDERING THE USER CONTEXT

AUTHOR: Cassiano Andrei Dias Da Silveira Schneider

ADVISOR: Celio Trois

In recent years, there has been a considerable increase in the number of apps installed on people's mobile phones, demanding more from the storage capacity of each device and taking corporate space off of native apps for other industries like social networking, entertainment and messaging apps. This project aims to enhance the business-customer relationship experience through technology by implementing an architecture that enables people without technical knowledge to build PWA websites that have the same experience as an application, but without relying on installation. In addition, content becomes context sensitive, providing users with accurate information. These features eliminate the difficulties companies have to digitally deal with their customers.

Keywords: Development Methodology. Web development. Progressive Web Apps. Framework Django. Context Sensitive Applications. Distribution of contents.

LISTA DE FIGURAS

Figura 1.1 – Perda de usuários nas etapas de utilização de aplicativo nativo.	9
Figura 2.1 – Arquitetura PWA	12
Figura 2.2 – Service Worker API	13
Figura 3.1 – Arquitetura ACDC	17
Figura 3.2 – Atores da Arquitetura	18
Figura 3.3 – Seleção do conteúdo a partir do contexto.	19
Figura 3.4 – Diagrama de entidades do banco de dados.	21
Figura 3.5 – Fluxograma da Requisição	22
Figura 4.1 – Formulário de cadastro para criação do website.	24
Figura 4.2 – Formulário de cadastro de conteúdo.	25
Figura 4.3 – Formulário de cadastro de condições de contexto.	26
Figura 4.4 – Cadastro de conteúdo relacionado.	26
Figura 4.5 – Exibição do website.	27

LISTA DE TABELAS

Tabela 2.1 – Comparação entre as tecnologias web, PWA, híbrido e nativo.	11
--	----

LISTA DE ABREVIATURAS E SIGLAS

<i>API</i>	<i>Application programming interface</i>
<i>DRY</i>	<i>Don't Repeat Yourself</i>
<i>GPS</i>	<i>Global Positioning System</i>
<i>HTML</i>	<i>HyperText Markup Language</i>
<i>HTTPS</i>	<i>Hypertext Transfer Protocol Secure</i>
<i>KISS</i>	<i>Keep It Simple Stupid</i>
<i>PWA</i>	<i>Progressive Web Application</i>
<i>RAD</i>	<i>Rapid Application Development</i>
<i>SGBDR</i>	<i>Sistema Gerenciador de Base de Dados Relacional</i>
<i>SQL</i>	<i>Structured Query Language</i>
<i>ORM</i>	<i>Structured Query Language</i>
<i>URL</i>	<i>Uniform Resource Locator</i>

SUMÁRIO

1	INTRODUÇÃO	8
2	FUNDAMENTAÇÃO CONCEITUAL E FERRAMENTAS	11
2.1	PROGRESSIVE WEB APP	11
2.2	CAPTURANDO CONTEXTO DO AMBIENTE	13
2.3	FRAMEWORK DJANGO	14
2.4	TRABALHOS CORRELATOS	15
3	ARQUITETURA CONSIDERANDO CONTEXTO PARA DISTRIBUIÇÃO DE CONTEÚDOS	17
3.1	MÓDULOS E ATORES	17
3.1.1	Conteúdos	18
3.1.2	Condições de Contexto	19
3.1.3	Banco de Dados	20
3.2	APLICAÇÃO DO PWA	21
4	DISCUSSÃO DOS RESULTADOS	23
4.1	INTERFACES	23
4.1.1	Interface de cadastro do website	23
4.1.2	Interface de cadastro de conteúdo	23
4.1.3	Publicação	25
4.2	COMPORTAMENTO DO SERVICE WORKER	27
5	CONCLUSÃO	29
	REFERÊNCIAS BIBLIOGRÁFICAS	30

1 INTRODUÇÃO

A Internet tem visto grandes inovações no passado recente, com os websites ficando intuitivos e responsivos, onde o usuário é capaz de obter um conteúdo semelhante, independentemente do dispositivo usado. No entanto, com a introdução dos *smartphones*, os consumidores começaram a migrar da Web para esses dispositivos móveis, passando a usar aplicativos nativos. As pessoas estão conversando, jogando, assistindo vídeos, ouvindo música, fazendo diversas tarefas e realizando quase todos os tipos de atividades possíveis por meio de aplicativos instalados em seus *smartphones*. Os aplicativos nativos já representam mais da metade do tempo gasto de cada pessoa na própria vida digital (STELER, 2017).

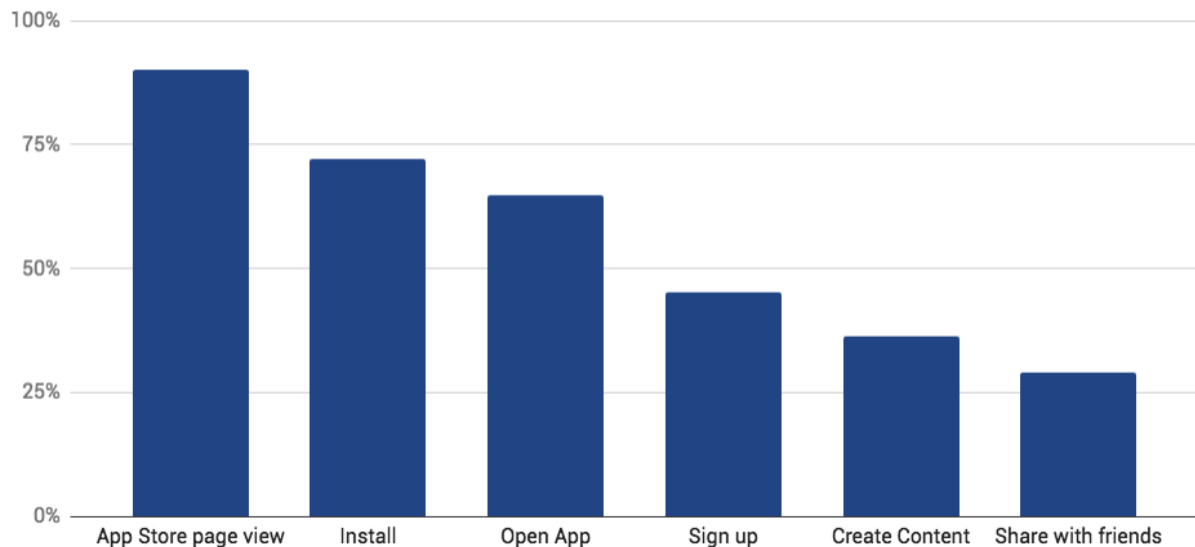
Os aplicativos nativos oferecem uma melhor experiência aos consumidores, pois os mesmos podem aproveitar todos os recursos do hardware dos dispositivos móveis, algo que não era possível nos websites executados nos celulares. No entanto, além de ser mais caro desenvolver um aplicativo nativo, o usuário muitas vezes não está disposto a baixar e instalar no seu celular, visto que custa tempo e ocupa memória de armazenamento do dispositivo. As aplicações Web, apesar de serem a alternativa mais fácil, apresentam limitações em seus recursos funcionais e dependem totalmente da conexão com a Internet, que torna-se um obstáculo devido a restrição de acesso em muitas regiões. Entretanto, com os avanços da tecnologia, uma nova abordagem está diminuindo a lacuna entre os aplicativos nativos e a Web para dispositivos móveis. Essa nova abordagem de desenvolvimento da Web é conhecida como Progressive Web Apps (PWA) (ANTUNES, 2019).

Progressive Web App é uma abordagem, proposta pela Google, para desenvolver aplicações que combinam recursos da tecnologia de aplicações Web e nativas (DEVELOPERS, 2019). Essas aplicações são inicialmente apresentadas como uma aplicação Web utilizando *design* responsivo, porém, considerando as interações contínuas do usuário, tornam-se mais completas e parecidas com aplicações nativas, ganhando acesso aos recursos do dispositivo. Além disso, uma aplicação que atende todos requisitos da arquitetura PWA passa maior confiabilidade, pois o sistema carrega instantaneamente, não apresentando ao usuário o aviso que está sem conexão, mesmo em casos de instabilidade na rede. Através de uso de *cache* local e mecanismos de sincronização a aplicação responderá rapidamente às interações do usuário (BIØRN-HANSEN; MAJCHRZAK; GRØNLI, 2017).

A arquitetura PWA evita que usuários desistam de instalar a aplicação, sem essa arquitetura integrada, um usuário que deseja experimentar o aplicativo, precisa passar por algumas etapas antes de começar a utilizar as informações: buscar, instalar, abrir e se cadastrar. A Figura 1.1 mostra, em porcentagem, a quantidade de usuá-

rios em cada uma dessas etapas no caso do aplicativo DrawChat¹, exibindo uma perda de aproximadamente 20% dos usuários para cada uma dessas etapas (CSELLE, 2018). O mesmo não acontece na Web, visto que, basta acessar um link e poucos segundos depois pode-se experimentar o produto. Ou seja, o usuário não precisa se comprometer e perder tempo instalando um aplicativo para avaliar sua funcionalidade.

Figura 1.1 – Perda de usuários nas etapas de utilização de aplicativo nativo.



Fonte: (CSELLE, 2018)

Como o PWA permite que aplicações Web acessem os recursos nativos dos dispositivos, faz-se possível a utilização dos dados do contexto em que o usuário se encontra. Algumas aplicações, como as detalhado na Seção 2.4, aplicam funcionalidades sensíveis ao contexto. Essas aplicações conseguem entender o contexto na qual estão sendo executadas e gerar informações significativas para um determinado fim (SILVA et al., 2017). A sensibilidade ao contexto pode ser entendida como a capacidade do sistema de inferir o contexto do usuário para decidir como agir e como realizar uma ação, de modo a oferecer um serviço de alta qualidade ao usuário (KIM et al., 2004). Logo, uma aplicação sensível ao contexto consiste em uma aplicação que consegue compreender o que está ocorrendo ao redor do usuário e busca, a partir dessa caracterização do contexto, proporcionar algum benefício para o usuário naquele momento. No entanto, as arquiteturas existentes não facilitam a criação de aplicações que usem PWA e simultaneamente sejam sensíveis ao contexto.

Este trabalho tem como objetivo reunir recursos para implementar uma arquitetura chamada ACDC (**A**rquitetura considerando **C**ontexto para **D**istribuição de **C**onteúdos), a principal finalidade dela é permitir a criação de websites com características PWA, que possam adaptar o conteúdo apresentado em função de dados de

¹Disponível em: <<https://draw.chat/>>

contexto do usuário (*context-aware*). Essa arquitetura, permitirá melhorar a precisão das informações que o usuário procura nos websites, ou seja, o conteúdo será exibido ao usuário de acordo com o contexto em que ele se encontra.

Um exemplo prático dessa tecnologia seria ao entrar no restaurante e conectar no website do estabelecimento, o cliente receberá na tela do seu celular o cardápio de acordo com o horário (café da manhã, almoço ou janta) e também produtos de acordo com o tempo em que se encontra no local (aperitivos, pratos principais ou sobremesas). Além disso, se o cliente tem menos de 18 anos, serão removidos produtos alcoólicos do cardápio. Além do restaurante, vários tipos de organizações podem usufruir dessas funções, como por exemplo, um indivíduo que ao participar de um congresso vai conseguir visualizar as palestras que estão acontecendo no horário atual, e ainda ser identificado em qual sala está presente para interagir com a palestra utilizando perguntas e comentários.

O *back-end*² da arquitetura foi implementada utilizando o *framework* de desenvolvimento Web Django (FORCIER; BISSEX; CHUN, 2008) e todos os dados são armazenados no banco de dados PostgreSQL³. O Django é responsável por conduzir o procedimento de criação dos websites, cadastro de conteúdos e gerenciamento das funções que filtrarão e exibirão os dados dependendo do contexto do usuário. Essa filtragem de informações é exercida pelas funções que validam se o contexto que o usuário se encontra é o mesmo exigido pelo conteúdo; por exemplo, (1) o usuário se encontra no turno da noite? (2) Na cidade de Santa Maria? E (3) está conectado na Internet? Se todas essas condições forem validadas pela arquitetura, as opções de compra com tele-entrega podem ser exibidas para o usuário.

Cada website cadastrado na ACDC é personalizado, ou seja, os conteúdos são distribuídos de forma especificada para atender as exigências do proprietário; por exemplo, um cardápio de restaurante precisa de uma estrutura de organização dos produtos de uma forma diferente que um estabelecimento de serviços de beleza. Ademais, todos websites gerados funcionam dentro dos moldes do PWA, são indexados no buscador da Google e apresentam os metadados organizados para para o conteúdo ser compartilhado na maioria das redes sociais.

Na sequência do texto constam a revisão de literatura (Capítulo 2), que é organizada de forma a fornecer o conhecimento necessário para o entendimento das tecnologias aplicadas dentro da arquitetura. No Capítulo 3 encontram-se os métodos usados para desenvolver a arquitetura, enquanto que os resultados obtidos no protótipo desenvolvido são relatados no Capítulo 4. Por fim, constam as considerações finais e trabalhos futuros são apresentados no Capítulo 5.

²Parte da arquitetura responsável pelo processamento dos dados no lado servidor.

³Disponível em: <<https://www.postgresql.org/>>

2 FUNDAMENTAÇÃO CONCEITUAL E FERRAMENTAS

Este capítulo aborda conceitos essenciais para o desenvolvimento de uma arquitetura para distribuição de conteúdos e recursos sensíveis ao contexto. Primeiramente será apresentada a técnica PWA, após, são exploradas diferentes formas de capturar o contexto do usuário através dos recursos disponíveis nos dispositivos móveis. Na sequência é sucintamente apresentado o *framework* que sustenta o *back-end* do trabalho e por fim, são descritos trabalhos relacionados que propõem soluções semelhantes a esta proposta.

2.1 PROGRESSIVE WEB APP

PWA é uma técnica de desenvolvimento que busca trazer para as soluções *mobile* e *desktop*, a mesma experiência vivenciada em um aplicativo nativo, porém consumindo menos dados. A proposta do PWA consiste em melhorar a experiência do usuário em sistemas Web, fazendo com que os websites possuam as características dos aplicativos nativos, como funcionamento offline, ícone na tela principal do dispositivo usado, atualização automática e notificações *push*, tudo isso sem a necessidade de que o mesmo instale o aplicativo no seu celular.

Tecnicamente PWA é um aperfeiçoamento das tecnologias Web. Essa tecnologia também pode ser visto como uma combinação ou uma evolução híbrida entre websites e aplicativos nativos, unindo as principais características: Baixo custo de desenvolvimento, funcionamento offline, instalável, acesso ao hardware e multiplataforma (ANTUNES, 2019). A Tabela 2.1 faz uma comparação entre os recursos disponíveis para cada tipo de aplicação: Web, PWA, híbrido e nativo.

Tabela 2.1 – Comparação entre as tecnologias web, PWA, híbrido e nativo.

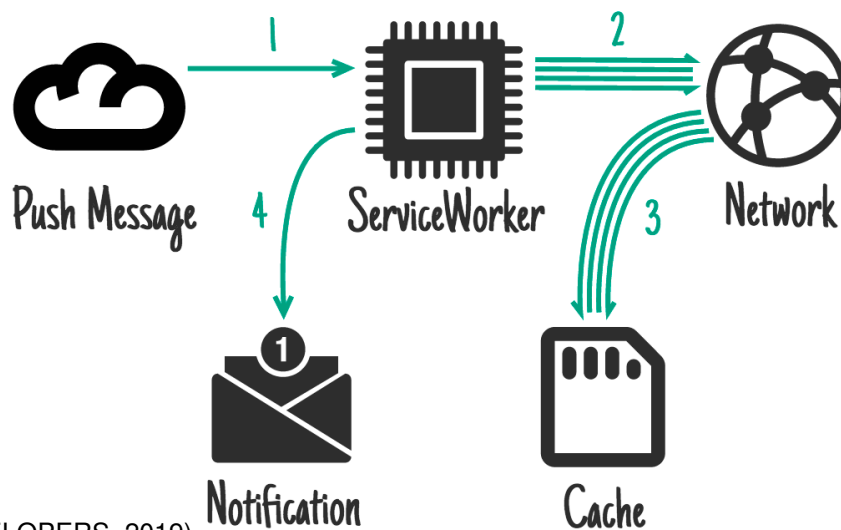
Funcionalidades	Web	PWA	Híbrido	Nativo
Responsividade	Sim	Sim	Sim	Sim
Funciona offline	Não	Sim	Sim	Sim
Atualizações rápidas	Sim	Sim	Não	Não
Indexáveis pelos motores de busca	Sim	Sim	Não	Não
Notificações	Não	Sim	Sim	Sim
Instalável	Não	Sim	Sim	Sim
Multiplataforma	Sim	Sim	Sim	Não
Presença em lojas virtuais	Não	Não	Sim	Sim
Acesso ao hardware	Não	Parcial	Sim	Sim

Fonte: (CECONI, 2019)

Aplicações PWA devem apresentar os seguintes requisitos: (i) ser seguras a

ponto de garantir que esses dados e controle de informações não vazem para uma outra aplicação ou fujam para um usuário malicioso, (ii) serem confiáveis carregando instantaneamente e nunca estando indisponíveis, mesmo quando a conexão à internet for precária ou até nula, (iii) serem eficientes respondendo de forma rápida e natural a interações com o usuário, fazendo uso de animações e (iv) serem atraentes, parecendo uma aplicação nativa em qualquer dispositivo (DEVELOPERS, 2019). Para cada um desses requisitos, existem técnicas e tecnologias (Figura 2.1) que são os principais atores, ou seja, são aqueles que fazem com que esses requisitos sejam alcançados. Os itens são: Service Worker, Cache API, Notificações *Push* e *Web App Manifest*.

Figura 2.1 – Arquitetura PWA



Fonte: (DEVELOPERS, 2019)

Service Worker API é um sistema que intercepta as requisições da aplicação guardando os resultados na cache do dispositivo do usuário. Isso permite uma velocidade de consumo de dados eficiente, além de uma aplicação funcional sem conexão (Figura 2.2). Desse modo, esse sistema atua como uma espécie de *cache*, funcionando como uma espécie de memória local para as requisições (MALAVOLTA et al., 2017).

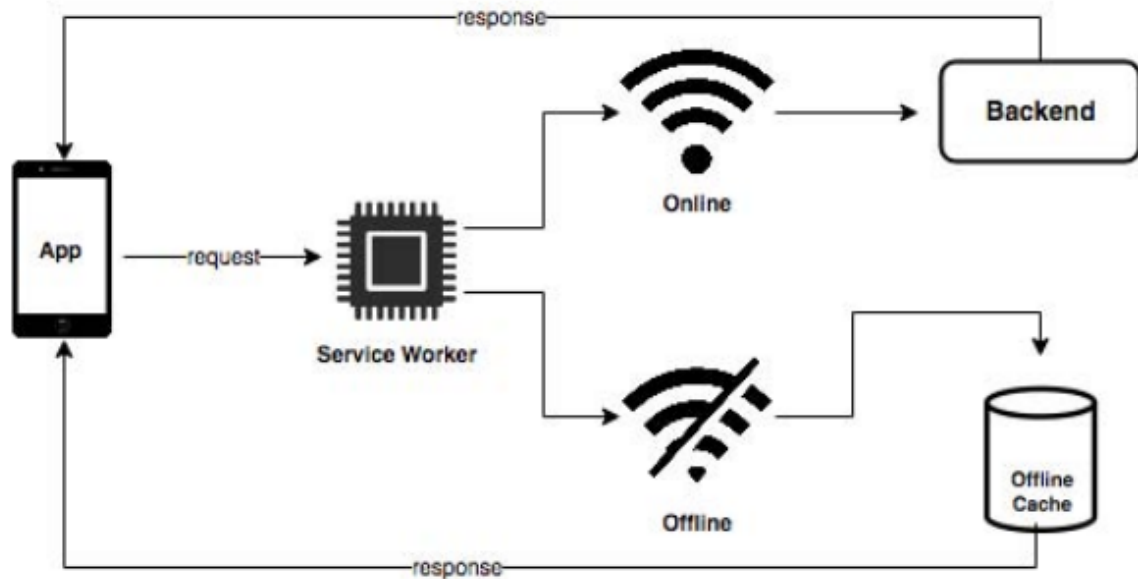
Cache API¹ é uma ferramenta para armazenar requisições localmente, para que quando solicitadas posteriormente, sejam reutilizadas. A API de cache foi criada para permitir que o Service Worker armazene em cache as solicitações de rede para fornecer respostas apropriadas, mesmo estando offline. No entanto, a API também pode ser usada como um mecanismo de armazenamento geral.

Notificações Push² é um recurso utilizado por todos os Aplicativos para alertar sobre atualizações, notícias, sinalizações de falta de uso do aplicativo e demais

¹Disponível em: <<https://developers.google.com/web/fundamentals/instant-and-offline/web-storage/cache-api>>

²Disponível em: <<https://developers.google.com/web/fundamentals/codelabs/push-notifications>>

Figura 2.2 – Service Worker API



Fonte: (Gambhir; Raj, 2018)

informações. As notificações *Push* são usadas para notificar os usuários de determinados serviços sobre diversas funcionalidades. Um exemplo dessa tecnologia são as mensagens do aplicativo WhatsApp que são notificadas na tela do celular. Recentemente, esta tecnologia vem sendo implementada em websites, buscando criar uma maior interação com o público, sem que tenham a necessidade de estar utilizando um aplicativo nativo.

Web App Manifest³ é um documento que tem como finalidade padronizar as aplicações Web, possibilitando que o usuário tenha como adicionar um ícone do website em sua tela inicial, sem ter a necessidade de baixar um aplicativo (DEVELOPERS, 2019). Ao incluir esta tecnologia em seu projeto, o usuário tem acesso mais rápido ao conteúdo, o que faz com que toda a sua experiência seja mais enriquecedora. Embora possa ser usada em qualquer website, o Web App Manifest é obrigatório dentro do PWA (DEVELOPERS, 2019).

2.2 CAPTURANDO CONTEXTO DO AMBIENTE

Um dispositivo móvel contém muitos recursos capazes de detectar o contexto em que o usuário se encontra. Esses recursos são componentes do seu hardware que anteriormente somente os aplicativos nativos tinham acesso, mas que com o sur-

³Disponível em: <<https://developers.google.com/web/fundamentals/web-app-manifest>>

gimento de novas tecnologias modificaram-se. Atualmente websites contam com o acesso a quase todos esses recursos.

O website *What Web Can Do*⁴ reúne várias funcionalidades que permitem que um website interaja com os recursos de hardware do dispositivo, dentre elas podem-se obter dados sensíveis ao contexto a partir de:

1. A *Network Information API* Fornece informações básicas sobre a conexão de rede atual, como a velocidade de conexão.
2. A *WiFi Information API* fornece informações sobre a força do sinal, o nome da rede atual, as redes Wi-Fi disponíveis, e assim por diante.
3. A *Date Javascript API* retorna um objeto que representa a data e hora do dispositivo. Internamente, o tempo é armazenado como o número de milissegundos desde 01 de janeiro de 1970, UTC.
4. A API geográfica (*API Geolocation*) permite que aplicativos autorizados da Web acessem os dados de localização fornecidos pelo dispositivo, obtidos usando GPS ou no ambiente de rede. Além da consulta pontual de local, permite que o aplicativo seja notificado sobre as alterações de local.
5. A *API Web Bluetooth* é uma API de baixo nível que permite que aplicativos da Web acessem os serviços expostos por dispositivos habilitados para Bluetooth nas proximidades.
6. O reconhecimento de fala (*Web Speech API*) permite que aplicativos autorizados da Web acessem o microfone do dispositivo e produza uma transcrição da voz que está sendo gravada. Isso permite que os aplicativos da Web usem a voz como um dos métodos de entrada e controle, semelhante ao toque ou teclado.

Existem outros recursos disponíveis com a finalidade em coletar dados sensíveis ao contexto, o objetivo deste trabalho é desenvolver uma versão protótipo do ACDC utilizando os que apresentam mais utilidade pro usuário.

2.3 FRAMEWORK DJANGO

O *Framework* de desenvolvimento Web chamado Django, encoraja um desenvolvimento de design limpo, rápido e pragmático (SOLORZANO; SCHNEIDER; CHARRAO, 2019). Baseado no padrão MTV (*Model, Template, View*), é um *framework*

⁴<https://whatwebcando.today/>

robusto, utilizado por grandes empresas como Mozilla, Pinterest, Instagram e globoesporte.com, o maior website de esportes da América Latina que, por exemplo, recebe mais de 20 milhões de visitantes por mês. O *framework* utiliza todos os princípios e metodologias da linguagem Python como DRY, RAD e KISS (SILVA; SILVA, 2019). Além disso, através da disponibilização de módulos, o Django facilita o reuso de código em várias tarefas do desenvolvimento web como, autenticações, manuseio de banco de dados, segurança, desenvolvimento de formulários entre outros (FORCIER; BISSEX; CHUN, 2008).

Django possui uma comunidade ativa que incentiva o compartilhamento de pacotes e funções. O website Django Packages⁵ oferece milhares de componentes prontos que podem ser anexados ao seu projeto, exigindo apenas algumas configurações para integrar por completo na sua aplicação. Desde o início do mesmo, o desenvolvedor é influenciado pela comunidade a desenvolver seu projeto com uma estrutura modular, permitindo sempre expandi-lo com pacotes e componentes da comunidade Python e Django (FORCIER; BISSEX; CHUN, 2008).

Django desempenhara o papel de *back-end* da arquitetura, gerenciando o banco de dados que foi integrado, controlando as requisições, renderizando todas as páginas em HTML e gerenciando as APIs que garantem as características PWA e gerenciamento das funções de contexto.

2.4 TRABALHOS CORRELATOS

Nesta seção é apresentado trabalhos que possuem as principais funções e tecnologias relacionados aos recursos da ACDC.

Silva et al. (SILVA et al., 2017) utilizou técnicas de prototipação rápidas para a criação de aplicações sensíveis ao contexto, realizando um estudo comparativo de middlewares com capacidade para fornecer dados em tempo real do contexto. O trabalho explora o uso das técnicas PWA para o desenvolvimento do protótipo. Este estudo de caso, envolve uma aplicação para mobilidade urbana, onde os dados consumidos são do posicionamento GPS (Global Positioning System) em tempo real dos ônibus urbanos da cidade de Uberlândia.

Marinho et al. (MARINHO; JUNIOR; VIANA, 2016) explorou técnicas de sensibilidade ao contexto e criou uma aplicação desenvolvida para a plataforma Android, cuja principal funcionalidade é permitir que o indivíduo possa cadastrar lembretes e ser notificado em contextos definidos pelo mesmo. Os sensores utilizados nessa aplicação são: GPS, relógio e as entradas que o usuário fornece. Além do desenvolvimento

⁵djangopackages.org

do aplicativo, esse trabalho consiste em avaliar sistemas de notificação sensíveis ao contexto e como podem ser bem aceitas pelo público. Durante o desenvolvimento do presente trabalho, tornou-se necessário investigar a aceitação para compreender melhor de que forma sistemas de notificação sensíveis ao contexto podem trazer benefícios.

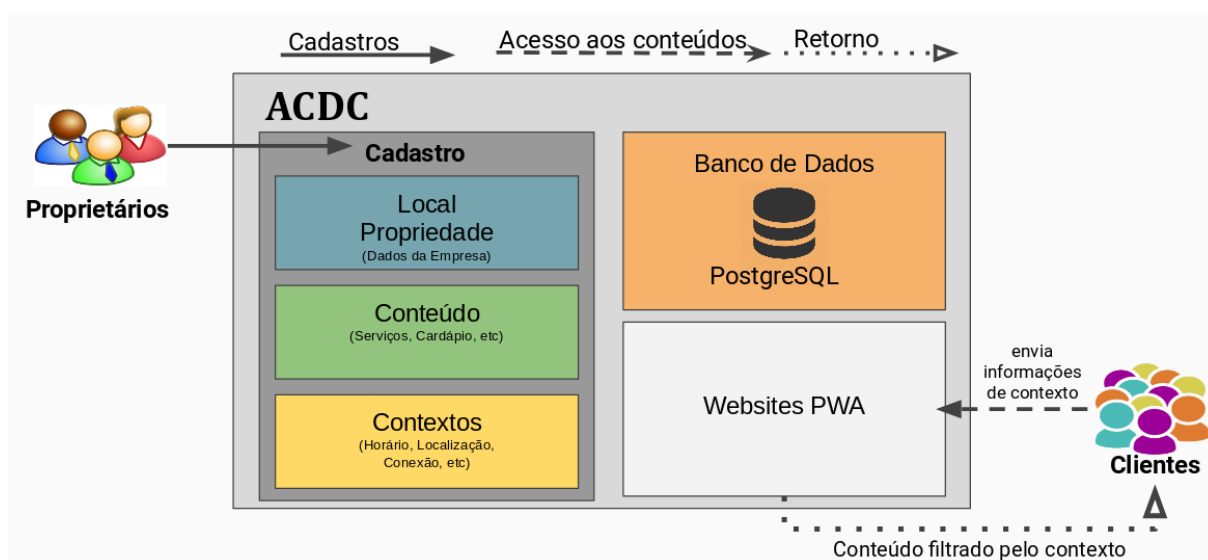
O site GoodBarber (GOODBARBER, 2019) é um construtor de aplicativos criado em 2011, cujo objetivo é tornar a tecnologia disponível tanto para mobile quanto para web, permitindo assim que usuários criem um aplicativo sem necessidade de conhecimento técnico. Para tal ação, as opções são aplicativos de conteúdo e e-commerce, para arquiteturas nativas Android, ios e ademais aplicativos PWA. Quando iniciado, o site construía apenas aplicativos nativos, com a inserção do pwa, o número de aplicativos teve uma significativa ampliação, contando atualmente com mais de 30 mil aplicativos publicados.

Os trabalhos relatados nesta seção exploraram recursos da metodologia de desenvolvimento PWA, aplicações sensíveis ao contexto e distribuição de conteúdos, contudo, nenhum deles aplicou todos os recursos simultaneamente em seus trabalhos. Na próxima seção, será relatado como foi utilizado esses recursos para desenvolver a ACDC.

3 ARQUITETURA CONSIDERANDO CONTEXTO PARA DISTRIBUIÇÃO DE CONTEÚDOS

Este capítulo apresenta as etapas para o desenvolvimento da ACDC (**A**rquitetura considerando **C**ontexto para **D**istribuição de **C**onteúdos). A arquitetura implementa três módulos principais que permitem a (1) criação de websites, (2) cadastro de conteúdos e suas respectivas condições de contexto para ser exibido (ex.: local, data, hora) e (3) estruturas personalizadas para exibição dos conteúdos. A Figura 3.1) apresenta uma visão geral da ACDC e sua interação com os usuários. É importante salientar que as funcionalidades de cadastro são restritas aos proprietários, enquanto que os clientes possuem acesso somente aos conteúdos. As seções seguintes apresentam detalhes da implementação desses módulos.

Figura 3.1 – Arquitetura ACDC



Fonte: Do autor

3.1 MÓDULOS E ATORES

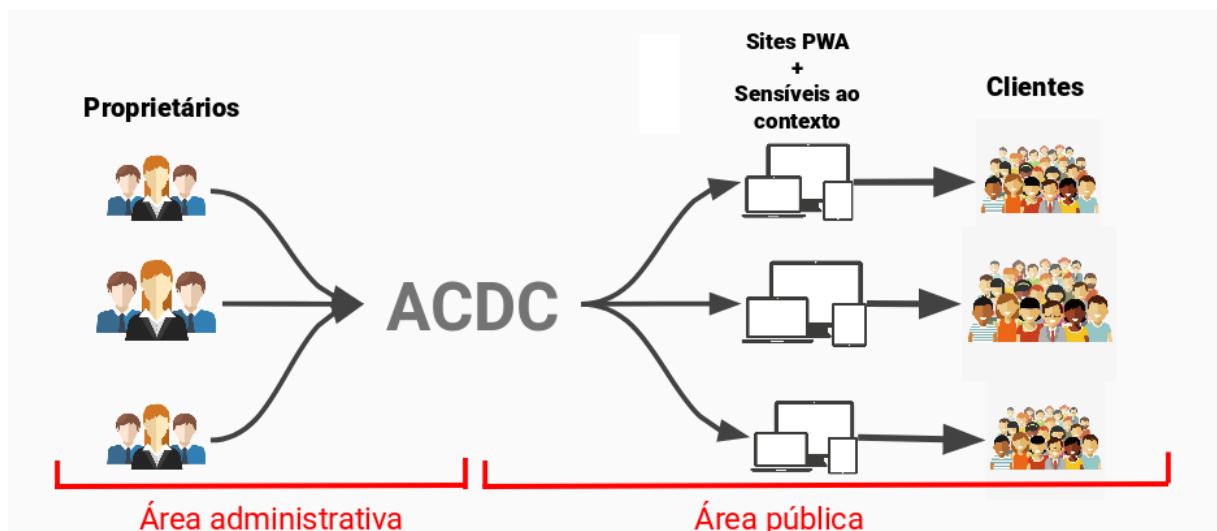
A arquitetura foi dividida em duas áreas principais: (I) Área administrativa, no qual sua principal função é cadastrar dados que posteriormente esses dados serão processados para automaticamente gerar um website e (II) a área pública, onde os clientes irão acessar os websites gerados. A **área administrativa** possui uma interface onde encontram-se os formulários de cadastros para o proprietário criar um

website, criar conteúdos e atribuir condições de contexto para o mesmo. A **área pública** apresenta para cada website uma interface diferente, posicionando os conteúdos conforme foram relacionados no momento do cadastro.

O primeiro formulário da área administrativa a ser preenchido é o que alimenta a tabela `Locais` do banco de dados (Figura 3.4), que representa os dados do local que o proprietário está registrando. Os outros formulários alimentam as outras tabelas restantes da Figura 3.4, o qual têm finalidade gerar conteúdos para o website, ademais atribuem condições de contexto para os mesmos, e por conseguinte, será gerado o website compondo a área pública.

A Figura 3.2 demonstra que a plataforma ACDC é um centralizador dos conteúdos onde páginas são cadastradas pelos proprietários e, posteriormente, renderizadas de forma personalizada para seus clientes.

Figura 3.2 – Atores da Arquitetura



Fonte: Do autor

3.1.1 Conteúdos

Cada *website* receberá uma URL para acessar a página com os conteúdos cadastrados. As páginas geradas pela plataforma, apesar de serem personalizadas, todas possuem recursos padrões, a primeira é que todas páginas dispõem os requisitos mínimos exigidos da arquitetura PWA¹. Os requisitos mínimos são: website em HTTPS, páginas responsivas, todas URLs carregam sem conexão com a Internet, possível adicionar atalho à tela inicial do celular, primeiro carregamento rápido (mesmo

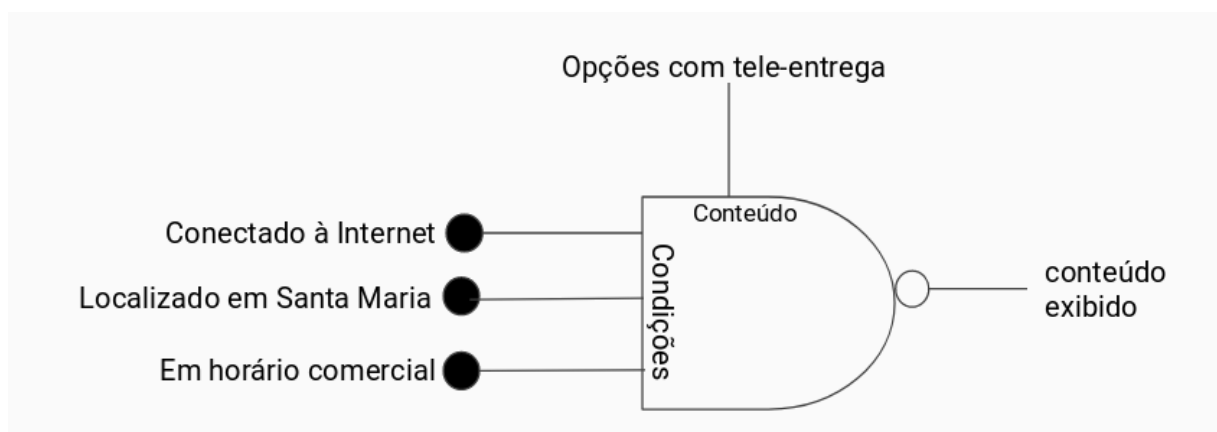
¹Disponível em: <<https://developers.google.com/web/progressive-web-apps/checklist>>

em 3G), website funciona em vários navegadores, transições ágeis entre páginas e cada página possui uma URL. Além dos requisitos mínimos, a ACDC cumpre com recursos como: Todos conteúdos são indexados no Google e todas as páginas geram metadados, que ajudam na indexação do seu website nos mecanismos de pesquisa, permitindo também citar os conteúdos em plataformas de compartilhamento, como Twitter, LinkedIn, Facebook, etc.

3.1.2 Condições de Contexto

As páginas dos websites exibem conteúdos precisos, que são informações que respondem ao que o usuário procura naquele momento, isso se deve ao fato que o conteúdo foi filtrado antes de ser apresentado na tela, essa filtragem é o proprietário que define quando está cadastrando os conteúdos do seu website. A ACDC permite atribuir condições que serão validadas ao obter o contexto do usuário numa requisição, caso todas condições forem válidas, o conteúdo é exibido. Cada conteúdo pode ter diferentes condições atribuídas, mas caso um conteúdo não detenha nenhuma condição atribuída, ele sempre será exibido. Seguindo um exemplo prático conforme a Figura 3.3, o conteúdo (opções com tele-entrega), será exibido caso as três condições do contexto forem verdadeiras.

Figura 3.3 – Seleção do conteúdo a partir do contexto.



Fonte: Do autor

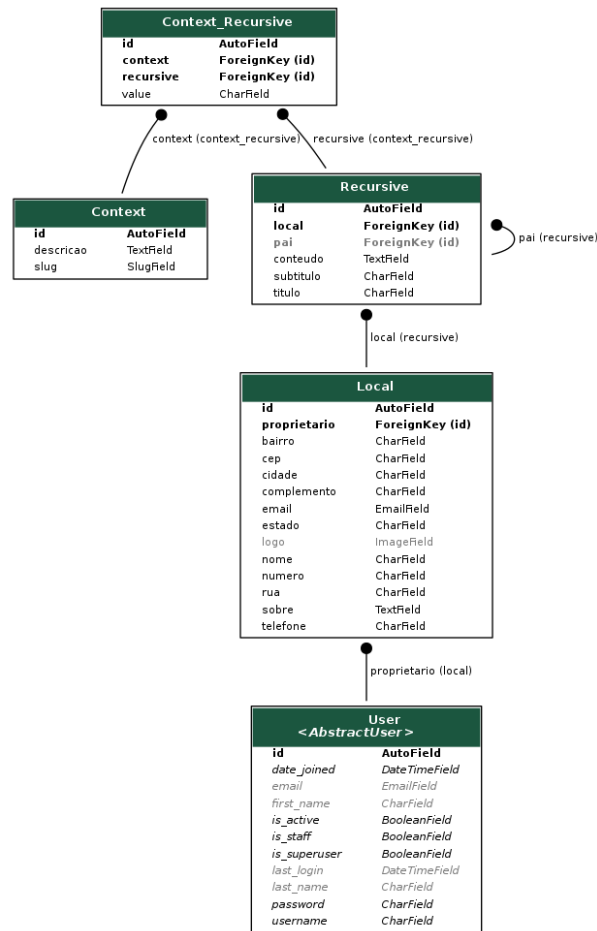
3.1.3 Banco de Dados

O *framework* Django tem suporte para vários banco de dados, entre eles foi escolhido o PostgreSQL (AMARAL; LIZARDO; SOUZA,), que é um SGBDR em software livre que permite a utilização da linguagem SQL, ademais é um banco de dados de alta performance e de fácil utilização em projetos. Para uma plataforma ser capaz de gerar websites personalizados, é necessário um banco de dados relacional estruturado de forma que qualquer website consiga organizar seu conteúdo. O modelo relacional (Figura 3.4) foi codificado dentro do Django, visto que o *framework* possui características ORM que são conjuntos de classes que permite que você desenvolva sem precisar escrever códigos de conexão com o banco (códigos da linguagem SQL) a todo momento, preservando as características de orientação a objetos da linguagem.

As entidades relacionadas permitem que um mesmo usuário consiga cadastrar vários `Locais`, esses locais podem ser especificados como empresas, pequenos negócios e organizações em geral. O banco está preparado para receber informações como descrição, localização, contato e redes sociais. O proprietário pode cadastrar várias redes sociais do local, por exemplo, a adição de vários perfis do Instagram do mesmo local, mas com idiomas diferentes; nesse caso, o conteúdo será exibido de acordo com a nacionalidade do usuário.

Para informações mais complexas, como cardápio, serviços, produtos, o usuário proprietário do website, deve usar a tabela `Recursive`. As entidades relacionadas no banco de dados da ACDC permitem que o usuário cadastre conteúdos recursivamente, podendo organizar grupos e subgrupos para sistematizar as informações em sua página personalizada. Isso permite, por exemplo, que o conteúdo de um cardápio seja cadastrado e exibido de forma organizada e classificada (ex.: Pratos Principais → Temakis → Temaki Filadélfia Sem Arroz). Um objeto da tabela `Recursive` pode ter vários objetos `Context`, onde serão aplicadas funcionalidades que permitam identificar o contexto, para filtrar os dados antes de exibir na tela; essas funcionalidade foram detalhadas na Seção 2.2.

Figura 3.4 – Diagrama de entidades do banco de dados.

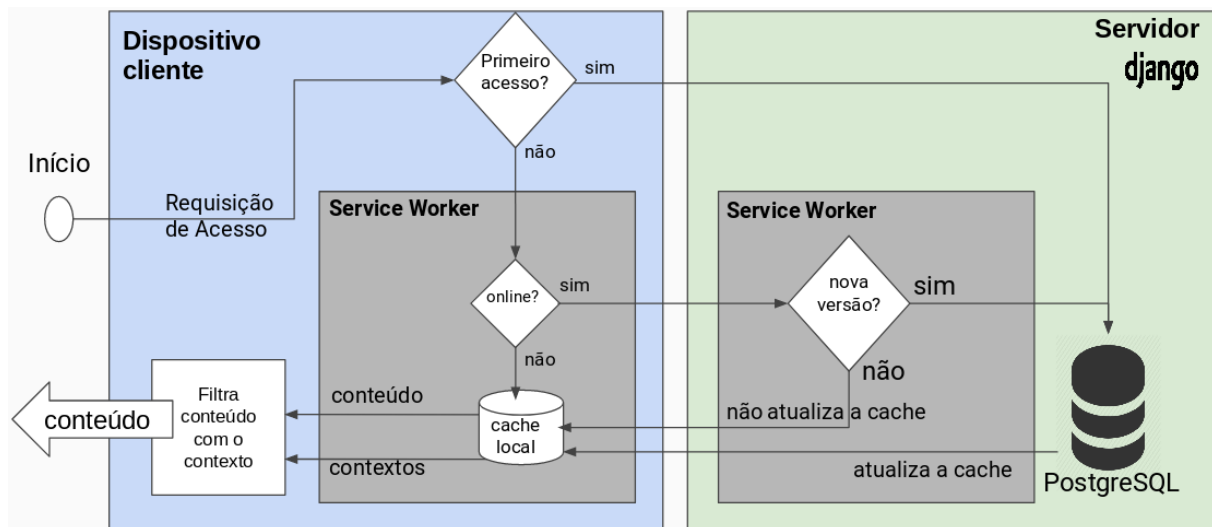


Fonte: Do autor

3.2 APLICAÇÃO DO PWA

A API *Service Worker* é o principal recurso do PWA, ela permite requisições mesmo com a conexão ruim ou inexistente. Esse recurso está dividido entre o dispositivo do cliente e o *framework* no servidor, por isso é preciso que o cliente faça o primeiro acesso para iniciar a instalação do *Service Worker*. Esse processo, desde a requisição do usuário até o retorno dos dados para seu dispositivo está representado na Figura 3.5.

Figura 3.5 – Fluxograma da Requisição



Fonte: Do autor

No primeiro acesso do cliente ao conteúdo, a API do *Service Worker* será instalada no dispositivo e todos os conteúdos sensíveis ao contexto serão salvos na *cache* do cliente para depois serem filtrados de acordo com o contexto em que o cliente se encontra. Com o *Service Worker* instalado no cliente, no próximo acesso que o cliente requisitar, o *Service Worker* irá testar sua conexão com a Internet. Caso não exista conexão, os dados da *cache* serão retornados. É importante salientar que os dados continuarão sensíveis ao contexto do cliente pois, no primeiro acesso, além de todos conteúdos, as regras de contexto também foram armazenados na *cache*.

Caso o dispositivo esteja conectado, o *Service Worker* verificará se existe alguma versão atualizada dos conteúdos para modificar na *cache* antes de filtrar e exibir o conteúdo. Porém, enquanto é verificada a versão dos dados, os dados da *cache* são exibidos ao cliente, para que esse não fique esperando a resposta do servidor. Caso existir uma nova versão, somente os dados alterados serão atualizados.

Os métodos apresentados neste capítulo permitiram o desenvolvimento da arquitetura ACDC, que por sua vez, encontra-se na versão protótipo com interface visual simples. No próximo capítulo serão apresentados os detalhes do processo para a criação de um website, detalhes do funcionamento dos conteúdos e do processo interativo com os usuários.

4 DISCUSSÃO DOS RESULTADOS

Este capítulo tem como objetivo apresentar o protótipo desenvolvido utilizando metodologia descrita no capítulo anterior. Em uma primeira etapa, são expostas as telas da interface onde proprietários cadastram os dados de seus websites, em sequência é apresentada a interface para cadastrar os conteúdos que serão exibidos, juntamente com a interface para associar funções sensíveis ao contexto naquele conteúdo recém cadastrado. Por fim descreve-se como os conteúdos serão disponibilizados aos usuários conforme o contexto e sua conexão com a Internet.

4.1 INTERFACES

Para o protótipo da ACDC, foi implementada uma interface simples e intuitiva, usando a classe Forms¹ do Django para cadastrar todas informações de um local, junto com as informações, o usuário pode opcionalmente aplicar funcionalidades sensíveis ao contexto se for necessário exibir a informações em condições específicas.

4.1.1 Interface de cadastro do website

Para o website ser criado, a primeira etapa consiste no proprietário cadastrar os dados básicos do seu negócio: nome, endereço, funções, características, imagens e contato. A Figura 4.1 apresenta a tela do formulário para iniciar o processo de criação do website.

Após o cadastro inicial, o proprietário será redirecionado para um painel de controle que contém informações sobre seu website e uma interface para cadastrar o conteúdo de seus serviços ou produtos.

4.1.2 Interface de cadastro de conteúdo

De acordo com o modelo relacional do banco de dados (Figura 3.4), os conteúdos podem ser organizados e estruturados de acordo com a necessidade do proprietário, graças ao relacionamento recursivo da tabela onde os conteúdos são

¹ Documentação disponível em: <<https://docs.djangoproject.com/pt-br/2.2/topics/forms/>>

Figura 4.1 – Formulário de cadastro para criação do website.

Cadastrar local

Nome do Local:	<input type="text" value="Restaurante Italiano"/>
Telefone de contato:	<input type="text" value="(55)9-9999-9999"/>
Email de contato:	<input type="text" value="email@operadora.com"/>
Sobre:	<div style="border: 1px solid #ccc; padding: 5px; min-height: 100px;">Restaurante Italiano, com cardápio para almoço e cardápio para janta.</div>
Estado:	<input type="text" value="Rio Grande Do Sul"/>
Cep:	<input type="text" value="97050-400"/>
Cidade:	<input type="text" value="Santa Maria"/>
Bairro:	<input type="text" value="Centro"/>
Endereço (Avenida, Rua, etc):	<input type="text" value="Presidente Vargas"/>
Numero:	<input type="text" value="200"/>
Complemento:	<input type="text" value="Térreo"/>
LogoMarca:	Atualmente: logos/red-banner-and-chef-cap.png <input type="checkbox"/> Limpar Modificar: <input type="button" value="Browse..."/> No file selected.

Fonte: Do autor

cadastrados como apresentado na Seção 3.1.3. A Figura 4.2 mostra a interface para cadastrar conteúdo a ser exibido no website.

No mesmo formulário, o proprietário pode, opcionalmente, inserir funções sensíveis ao contexto, que são condições para permitir que aquele conteúdo seja exibido apenas nos cenários em que as condições são válidas (ex.: mostrar o cardápio apenas aos domingos). Caso opte em não inserir condições, o conteúdo será sempre exibido na tela. Por esse campo permitir testar as variáveis do contexto, é necessário que o usuário analise e planeje com atenção as condições aplicadas nos conteúdos, para evitar que o website resultante tenha pouco ou nenhum conteúdo

Figura 4.2 – Formulário de cadastro de conteúdo.

Adicionar conteúdo

Local: ✎ +

Pai: ✎ +


Título:

Sub-Título:

Conteúdo:

🔍 **B** *I* U ~~X~~ ~~X~~ ~~S~~ ~~B~~ Helvetica 14 A ☰ ☰ ☰ T!

📄 🔗 🖼️ 🎥 ➖ ✖ </> ?



Cardápio

Confira nossas opções de pratos no domingo e aproveite seu feriado com a gente!

Fonte: Do autor

exibido ou, apresente conteúdos redundantes. Depois de um conteúdo cadastrado, o proprietário pode cadastrar subgrupos para aquele conteúdo, por exemplo, o cardápio do almoço é o conteúdo "pai", representado na figura 4.2, e na figura 4.4 temos os pratos principais do almoço, que é subgrupo ou conteúdo "filho" relacionado ao cardápio do almoço.

4.1.3 Publicação

Após finalizar os cadastrados dos conteúdos, o proprietário pode publicar seu website e obter o *link* de acesso (URL), ele opcionalmente pode associar um domínio para essa URL. O conteúdo é agrupado conforme relacionado pelo proprietário (Figura 4.5), os textos e imagens de cada conteúdo são totalmente personalizados na interface de cadastro (Figura 4.2). Quando o cliente acessar o website publicado, serão

Figura 4.3 – Formulário de cadastro de condições de contexto.

CONDIÇÕES DE CONTEXTO

Condição de contexto: #1

Context: Semana ✎ +

Value: Domingo ▾

Condição de contexto: #2

Context: Turno ✎ +

Value: Almoço ▾

+ Adicionar outro(a) Condição De Contexto

Fonte: Do autor

Figura 4.4 – Cadastro de conteúdo relacionado.

Local: Restaurante Italiano ✎ +

Pal: Cardápio Almoço aos Domingos ✎ +

Título: Pratos Principais

Sub-Título: Principais pratos de Domingo

Conteúdo:

Tradicionais

Pão Italiano com manteiga
porção, opcional
7,90

Antepasto
berinjela, pimentão e abobrinha e pão italiano
23,00

Berinjela a Pizzaiola

Fonte: Do autor

exibidos os conteúdos que não possuem regras de contexto e os conteúdos em que o contexto associado está de acordo com o cenário em que o cliente se encontra. Os conteúdos que não condizem com o contexto não serão exibidos na tela.

Figura 4.5 – Exibição do website.



Cardápio

Confira nossas opções de pratos no domingo e aproveite seu feriado com a gente!

Tradicionais

Pão Italiano com manteiga

porção, opcional

7,90

Antepasto

berinjela, pimentão e abobrinha e pão italiano

23,00

Berinjela a Pizzaiola

23,50

Fonte: Do autor

4.2 COMPORTAMENTO DO SERVICE WORKER

No primeiro acesso do cliente ao website, a usabilidade será igual ao de um website normal em relação ao carregamento da página, isso porque o *Service Worker* fará o download de todos os dados da página para a memória do dispositivo do cliente. Por fim, um aviso será emitido ao cliente perguntando se ele gostaria de salvar um atalho do website na tela inicial do seu celular para facilitar seu acesso novamente, depois disso, o website portará as características do PWA para aquele cliente.

Caso o cliente volte a utilizar o website, no segundo acesso em diante, o com-

portamento seguirá características da arquitetura PWA conforme a Figura 3.5. Se o `cliente` não estiver conectado à Internet, os conteúdos que estão armazenados na *cache* serão filtrados pelo contexto que o dispositivo captar e logo serão exibidos; ou seja, um conteúdo que não foi apresentado no primeiro acesso pode aparecer no segundo acesso, como a *cache* foi alimentada no primeiro acesso com todos conteúdos do website, o conteúdo só depende do contexto que o dispositivo estiver para poder ser exibido ou não.

Toda requisição do `cliente` retorna os dados da *cache* imediatamente, mesmo que desatualizados, evitando que o `cliente` espere o carregamento da página. Porém, se o dispositivo do `cliente` estiver conectado à Internet, a *API Service Worker* enviará uma requisição para o servidor verificando se há uma nova versão dos dados. Em caso negativo, o processo acaba e o `cliente` segue a navegação porém, se houver uma versão nova, os dados serão sincronizados e atualizados na *cache* do `cliente`.

O objetivo desse comportamento é garantir que o usuário `cliente` desfrute o website com a mesma experiência de um aplicativo nativo, mesmo que em algumas situações os dados estejam desatualizados: é importante que todas requisições retornem alguma informação sem precisar esperar carregar os dados.

O propósito desse capítulo foi apresentar o protótipo desenvolvido da arquitetura ACDC, detalhando suas interfaces e comportamento dos dados. Esse capítulo buscou validar todas tecnologias aplicadas na arquitetura e demonstrar uma nova abordagem de desenvolvimento Web. O capítulo seguinte apresenta as considerações finais sobre o trabalho desenvolvido, estabelecendo sua contribuição e apontando melhorias que possam a vir a complementar em forma de trabalhos futuros.

5 CONCLUSÃO

Este trabalho apresentou a ACDC, uma arquitetura que integra a API *Service Worker* e as outras tecnologias do PWA dentro do *Framework* de desenvolvimento web Django para auxiliar na criação de websites com conteúdos sensíveis ao contexto e que respondem mesmo com o dispositivo do cliente *offline*. O objetivo do ACDC é que os clientes dos websites possam interagir em qualquer situações do cotidiano considerando a conexão com a internet, o ambiente e o tipo de dispositivo. Ademais da acessibilidade, a arquitetura ajuda os proprietários a disponibilizar conteúdos diferenciados, dependendo do contexto que o cliente se encontra, facilitando encontrar as informações precisas relacionadas ao que o cliente está buscando no website, funcionando como um filtro de informações sensível ao contexto.

Observou-se que com a arquitetura mais acessível com a tecnologia PWA, eliminou-se muitas limitações que os usuários tinham na hora de acessar os aplicativos, como por exemplo: convencer um cliente a baixar e instalar um aplicativo sem testar antes e convencer o cliente a manter o aplicativo instalado no celular, ademais, eliminou a necessidade de conexão com a Internet como existe nos websites tradicionais.

Como trabalhos futuros objetiva-se, primeiramente, expandir a implementação desta plataforma, melhorando a interface, além de adicionar mais possibilidades de personalizar o *layout* de cada website e adicionar novas funcionalidades para ampliar as formas de exibir as informações e atender as necessidades que um proprietário de website precisa para compartilhar seus serviços e produtos com seus clientes.

REFERÊNCIAS BIBLIOGRÁFICAS

AMARAL, H. R.; LIZARDO, L. E. O.; SOUZA, A. C. V. de. Postgresql: uma alternativa para sistemas gerenciadores de banco de dados de código aberto. In: **Anais do Congresso Nacional Universidade, EAD e Software Livre**. [S.l.: s.n.]. v. 2, n. 2.

ANTUNES, A. **PWA: O que são progressive web apps e por que usar?** 2019. <<https://gobacklog.com/blog/progressive-web-apps/>>.

BIØRN-HANSEN, A.; MAJCHRZAK, T. A.; GRØNLI, T.-M. Progressive web apps: The possible web-native unifier for mobile development. In: **WEBIST**. [S.l.: s.n.], 2017. p. 344–351.

CECONI, L. Experiência do usuário em progressive web apps. 2019.

CSELLE, G. **Every Step Costs You 20% of Users**. 2018. <<https://medium.com/gabor/every-step-costs-you-20-of-users-b613a804c329>>. [Online; acessado em 06-Agosto-2019].

DEVELOPERS, G. **Progressive Web Apps**. 2019. <<https://developers.google.com/web/progressive-web-apps/>>. [Online; acessado em 01-Outubro-2019].

FORCIER, J.; BISSEX, P.; CHUN, W. J. **Python web development with Django**. [S.l.]: Addison-Wesley Professional, 2008.

Gambhir, A.; Raj, G. Analysis of cache in service worker and performance scoring of progressive web application. In: **2018 International Conference on Advances in Computing and Communication Engineering (ICACCE)**. [S.l.: s.n.], 2018. p. 294–299. ISSN null.

GOODBARBER. **A filosofia GoodBarber**. 2019. <<https://pt.goodbarber.com/about/>>. [Online; acessado em 20-Novembro-2019].

KIM, S. W. et al. Sensible appliances: applying context-awareness to appliance design. **Personal and Ubiquitous Computing**, Springer-Verlag, v. 8, n. 3-4, p. 184–191, 2004.

MALAVOLTA, I. et al. Assessing the impact of service workers on the energy efficiency of progressive web apps. In: **Proceedings of the 4th International Conference on Mobile Software Engineering and Systems**. Piscataway, NJ, USA: IEEE Press, 2017. (MOBILESoft '17), p. 35–45. ISBN 978-1-5386-2669-6. Disponível em: <<https://doi.org/10.1109/MOBILESoft.2017.7>>.

MARINHO, C. S.; JUNIOR, M.; VIANA, W. Notificação sensível ao contexto: uma análise de aceitação. In: SBC. **Anais Estendidos do XXII Simpósio Brasileiro de Sistemas Multimídia e Web**. [S.l.], 2016. p. 65–68.

SILVA, P. H. P. et al. Prototipação rápida de aplicações orientadas à contexto em cidades inteligentes-o caso buzapp. Universidade Federal de Uberlândia, 2017.

SILVA, R. O. da; SILVA, I. R. S. Linguagem de programação python. **TECNOLOGIAS EM PROJEÇÃO**, v. 10, n. 1, p. 55–71, 2019.

SOLORZANO, A.; SCHNEIDER, C.; CHARAO, A. Pratique obi: Um recurso de apoio a treinos para a modalidade iniciação da olimpíada brasileira de informática. In: **Anais do XXVII Workshop sobre Educação em Computação**. Porto Alegre, RS, Brasil: SBC, 2019. p. 453–462. ISSN 2595-6175. Disponível em: <<https://portaldeconteudo.sbc.org.br/index.php/wei/article/view/6650>>.

STELER, F. **O fim da era dos aplicativos: Sua empresa está preparada?** 2017. <<https://computerworld.com.br/2017/01/02/o-fim-da-era-dos-aplicativos-sua-empresa-esta-preparada/>>. [Online; acessado em 08-Outubro-2019].