



**FACULDADE DE TECNOLOGIA DE AMERICANA
CURSO DE ANÁLISE E DESENVOLVIMENTO DE SISTEMAS**

ROGER DE OLIVEIRA TREVIZAN

**DESENVOLVIMENTO DE APLICAÇÕES WEB PARA EVENTUAL
USO OFF-LINE – PROTÓTIPO: CONTROLE DE TREINAMENTO DE
EXERCÍCIOS FÍSICOS**

**Americana, SP
2017**



**FACULDADE DE TECNOLOGIA DE AMERICANA
CURSO DE ANÁLISE E DESENVOLVIMENTO DE SISTEMAS**

ROGER DE OLIVEIRA TREVIZAN
roger.trevizan@gmail.com

**DESENVOLVIMENTO DE APLICAÇÕES WEB PARA EVENTUAL
USO OFF-LINE – PROTÓTIPO: CONTROLE DE TREINAMENTO DE
EXERCÍCIOS FÍSICOS**

Trabalho de Conclusão de Curso desenvolvido em cumprimento à exigência curricular do Curso de Análise e Desenvolvimento de Sistemas, sob a orientação do Prof. MSc. Rossano Pablo Pinto.

Área de concentração: Programação.

**FICHA CATALOGRÁFICA – Biblioteca Fatec Americana - CEETEPS
Dados Internacionais de Catalogação-na-fonte**

- T789d TREVIZAN, Roger de Oliveira
Desenvolvimento de aplicações Web para eventual uso off-line - protótipo:
controle de treinamento de exercícios físicos. / Roger de Oliveira Trevizan. –
Americana, 2017.
72f.
Monografia (Curso de Tecnologia em Análise e Desenvolvimento de
Sistemas) - - Faculdade de Tecnologia de Americana – Centro Estadual de
Educação Tecnológica Paula Souza
Orientador: Prof. Ms. Rossano Pablo Pinto
1 Programação 2. Web – rede de computadores I. PINTO, Rossano Pablo
II. Centro Estadual de Educação Tecnológica Paula Souza – Faculdade de
Tecnologia de Americana
- CDU: 681.3.06
681.519



Faculdade de Tecnologia de Americana

ROGER DE OLIVEIRA TREVIZAN


**DESENVOLVIMENTO DE APLICAÇÕES WEB PARA EVENTUAL
USO OFF-LINE – PROTÓTIPO: CONTROLE DE TREINAMENTO
DE EXERCÍCIOS FÍSICOS**

Trabalho de Conclusão de Curso desenvolvido em cumprimento à exigência curricular do Curso de Análise e Desenvolvimento de Sistemas, sob a orientação do Prof. MSc Rossano Pablo Pinto.


Área de concentração: Programação.

Americana, 11 de dezembro de 2017.

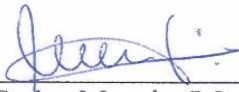
Banca Examinadora:



Rossano Pablo Pinto (Presidente)
Mestre
Faculdade de Tecnologia de Americana - FATEC



Eduardo Antonio Vicentini (Membro)
Mestre
Faculdade de Tecnologia de Americana - FATEC



Francisco Carlos Mancin (Membro)
Mestre
Faculdade de Tecnologia de Americana - FATEC

DEDICATÓRIA

Aos meus pais e minha família, por me acompanharem e me incentivarem durante toda a realização deste trabalho.

AGRADECIMENTOS

Agradeço à Deus, pois tudo que faço, antes e para sempre será Dele. Agradeço aos professores da FATEC – AM por todo o conhecimento propiciado e fornecido com profissionalismo, alegria e prazer aos estudantes. Em especial, agradeço ao mestre Rossano Pablo Pinto, que teve a paciência e, em sua orientação, a sabedoria para me ajudar a tornar realidade esse primeiro trabalho científico na área de TI. Agradeço aos meus amigos e colegas da instituição que partilharam de cada dia de estudos que passamos juntos e com certeza se tornaram valiosos para mim.

RESUMO

Com a inovação tecnológica, surgiram novos dispositivos, entre os quais figuram: dispositivos desktop e mobile. Estes trouxeram inúmeros tipos de novas aplicações para serem utilizadas pelos usuários. A informação gerada por esses novos aplicativos encontra-se mais interconectada e ao mesmo tempo espalhada pela internet como nunca foi presenciado. Tendo em vista que alguns fatores podem interromper o funcionamento da normal da rede, surge a necessidade de preservar tais informações de modo que os usuários ainda possam acessá-las numa eventual indisponibilidade. O presente trabalho objetiva-se a desenvolver um protótipo de uma aplicação Web que funcione on-line e eventualmente off-line (durante uma indisponibilidade de conexão/rede) para o gerenciar e controlar as sessões de treino e os exercícios físicos de um treinamento esportivo. A metodologia utilizada na elaboração teórica deste trabalho foi a pesquisa bibliográfica para documentação e diagramação, além das metodologias práticas sobre desenvolvimento de software e desenvolvimento de aplicativos Web (*Progressive Web Apps*). Utilizando a metodologia citada, foi possível desenvolver um protótipo que identifica a disponibilidade da rede (on-line/off-line) e controla dinamicamente as informações sobre os treinamentos realizados pelo usuário.

Palavras Chave: Programação (Computadores); Aplicações Web.

ABSTRACT

With technological innovation, new devices have emerged, among which are: desktop and mobile devices. These have brought numerous types of new applications to be used by users. The information generated by these new applications is more interconnected and at the same time spread out over the internet like never before. However, some factors may interrupt the normal operation of the network and it is necessary to preserve these information, so the users can still access them in the event of unavailability. The present work aims to develop a prototype of a Web application that works online and eventually offline (during a connection/network unavailability) to manage and control the training sessions and the physical exercises of a sports training. The methodology used in the theoretical elaboration of this work was the bibliographical research for documentation and diagramming, as well as the practical methodologies of software development and web application development (Progressive Web Apps). Using the methodology mentioned, it was possible to develop a prototype that identifies the availability of the network (online/offline) and dynamically controls the information about the trainings performed by the user.

Keywords: Programming; Web applications.

LISTA DE ILUSTRAÇÕES

Figura 1: Esquema de acesso de programas aplicativos às camadas de um sistema de computação.....	17
Figura 2: Desktop vs Mobile: quota no mercado mundial	18
Figura 3: Estrutura cliente-servidor	23
Figura 4: Referenciando o <i>Application cache</i> no código HTML.....	25
Figura 5: Estrutura interna do arquivo <i>Application cache</i>	25
Figura 6: API Web <i>Storage</i> – navegadores suportados e porcentagem de uso.....	26
Figura 7: Exemplo de codificação do objeto <i>local storage</i>	26
Figura 8: Exemplo de codificação do objeto <i>session storage</i>	27
Figura 9: Referenciando o arquivo <i>application manifest</i>	30
Figura 10: Comportamento de uma <i>promise</i>	32
Figura 11: Codificação e sintaxe padrão de uma <i>promise</i>	32
Figura 12: Ciclo de vida do <i>Service Worker</i>	34
Figura 13: Registrando o <i>Service Worker</i> na página Web	34
Figura 14: Evento de instalação do <i>Service Worker</i>	35
Figura 15: <i>Service Worker</i> - evento <i>fetch()</i>	35
Figura 16: Código usado para cachear recursos com o <i>Cache Storage</i> API.....	39
Figura 17: Seção <i>Cache Storage</i> no painel <i>Application</i> no Google Chrome.....	39
Figura 18: Verificar o suporte para o <i>IndexedDB</i> nos navegadores.....	40
Figura 19: Diagrama Caso de Uso do protótipo.....	47
Figura 20: Diagrama de Classe do protótipo – Tipo de Usuário e Usuário	48
Figura 21: Diagrama de Classe do protótipo – Sessão de Treino e Microciclo.....	49
Figura 22: Diagrama de Atividade do protótipo – Acessar sessão de treino	50
Figura 23: Diagrama de Atividade do protótipo – Alterar sessão de treino.....	51
Figura 24: Diagrama de Sequência do protótipo – funcionamento off-line.....	52
Figura 25: Diagrama de Sequência do protótipo – funcionamento on-line	53
Figura 26: Diagrama de Entidade-Relacionamento Tipo Usuário - Usuário.....	54
Figura 27: Diagrama de Entidade-Relacionamento Sessão de Treino e Microciclo	55
Figura 28: Tela Sessões de Treinamento do protótipo.....	57
Figura 29: Código do <i>Service Worker</i> contido no arquivo “ <i>service-worker.js</i> ”	58
Figura 30: Mensagem de instalação do <i>Service Worker</i> no console do navegador	59

Figura 31: <i>Service Worker</i> instalado no protótipo e "tcc-versão1" do cache criada	59
Figura 32: Recursos armazenados pelo <i>Service Worker</i> sendo carregados do cache.....	60
Figura 33: Código do evento <i>fetch</i> do <i>Service Worker</i>	61
Figura 34: "tcc_db" criado para o protótipo.....	61
Figura 35: Codificação para o <i>IndexedDB</i> do protótipo	62
Figura 36: Tela de <i>login</i> do protótipo	69
Figura 37: Tela visualizar sessão de treino do protótipo.....	70
Figura 38: Tela enviar sessão de treino do protótipo	71
Figura 39: Tela de pesquisa das sessões de treino do protótipo.....	72

LISTA DE TABELAS

Tabela 1: Vantagens e desvantagens de uma aplicação nativa	21
Tabela 2: Vantagens e desvantagens de uma aplicação Web.....	21
Tabela 3: Vantagens da utilização de <i>Progressive Web Apps</i>	28
Tabela 4: Comparativo entre APIs para armazenamento de dados off-line.....	37
Tabela 5: Classificação básica de microciclos de treinamento	43

LISTA DE ABREVIATURAS E SIGLAS

API - *Application Programming Interface*

APP – *Application*

CRUD – *Create, Retrieve, Update and Delete*

CSS - *Cascading Style Sheets*

DOM - *Document Object Model*

HTML - *HyperText Markup Language*

HTTP - *Hypertext Transfer Protocol*

IDB – *IndexedDB*

JSON - *JavaScript Object Notation*

PC – *Personal Computer*

PWA - *Progressive Web Apps*

SW – *Service Worker*

URL - *Uniform Resource Locator*

W3C - *World Wide Web Consortium*

WEBAPP – *Web Application*

WHATWG - *Web Hypertext Application Technology Working Group*

SUMÁRIO

1	INTRODUÇÃO	14
1.1	Delimitação do problema	14
1.2	Justificativa da escolha do tema	14
1.3	Objetivo	15
1.4	Metodologia	15
1.5	Organização do trabalho	16
2	REFERÊNCIAL TEÓRICO	17
2.1	O que são aplicativos	17
2.2	Aplicações nativas vs Web - vantagens e desvantagens	20
2.3	Funcionamento tradicional da Web	22
2.4	Tecnologias Web que permitem a utilização off-line	24
2.4.1	Armazenamento de conteúdo Web estático e levemente dinâmico	24
2.4.2	Progressive Web Apps	27
3	TECNOLOGIAS RECOMENDADAS PARA O DESENVOLVIMENTO DE PWAS	29
3.1	HTML5	29
3.1.1	Application Manifest	29
3.2	JAVASCRIPT	30
3.2.1	Programação assíncrona	31
3.2.2	Promises	31
3.2.3	Service Workers	32
3.3	Um pouco mais sobre armazenamento off-line	36
3.3.1	Cache Storage	37
3.3.2	IndexedDB	39
4	PROTÓTIPO DESENVOLVIDO	42
4.1	Treinamento desportivo	42
4.2	Documentação do protótipo da aplicação	44
4.2.1	Requisitos funcionais	45
4.2.2	Requisitos não funcionais	45
4.2.3	Diagrama de Caso de Uso	46
4.2.4	Diagrama de Classe	47

4.2.5 Diagrama de Atividade	49
4.2.6 Diagrama de Sequência	52
4.2.7 Diagrama de Entidade-Relacionamento	54
4.3 Prototipagem da aplicação	56
5 CONSIDERAÇÕES FINAIS	63
REFERÊNCIAS	65
APÊNDICE A - Tela de login do protótipo	69
APÊNDICE B - Tela visualizar sessão de treino do protótipo	70
APÊNDICE C - Tela enviar sessão de treino do protótipo	71
APÊNDICE D - Tela de pesquisa das sessões de treino do protótipo	72

1 INTRODUÇÃO

O advento das tecnologias nas mais diversas áreas do saber humano proporcionou uma grande revolução no mundo. Novos dispositivos e aplicativos surgiram para revolucionar a comunicação das pessoas. O caráter dinâmico dessas aplicações favoreceu o desenvolvimento de tecnologias Web: como *websites*, páginas de sites, aplicativos Web, entre outros. As informações se tornaram cada vez mais interconectadas, acessíveis e disponíveis para os usuários através da Internet.

Contudo, a Internet depende de uma série de fatores para se manter operacional e com tantas variáveis de funcionamento trabalhando juntas, pode ocorrer uma indisponibilidade momentânea na rede e o usuário acabar sem acesso as informações contidas em sua aplicação.

Utilizando os recursos oferecidos pelas novas tecnologias de desenvolvimento Web, aliadas as boas práticas recomendadas pela área de engenharia de software, é possível desenvolver um protótipo de um aplicativo que funcione tanto on-line quanto eventualmente off-line na Web e possibilite o acesso às informações de controle de uma rotina de treinamentos de exercícios físicos.

1.1 Delimitação do problema

A grande adoção na utilização de dispositivos móveis pelos usuários presenciada na atualidade certamente trouxe variedade para o controle e portabilidade de informações.

Aplicativos específicos que foram surgindo, extraem todo potencial desses novos dispositivos, porém fatores adversos (internos e externos) como: falta de cobertura de rede, grande variedade de hardware e versões de sistemas operacionais muito fragmentadas dificultam a disponibilidade de rede para o acesso às informações que os usuários possuem armazenadas nesses aplicativos.

1.2 Justificativa da escolha do tema

A escolha da abordagem do tema desse trabalho é fornecer o acesso para uma aplicação Web, independente da disponibilidade da rede (on-line/off-line). Os benefícios

dessa abordagem para o usuário é que ele sempre possa utilizar o aplicativo, independente de plataforma, dispositivo ou conectividade e, em última instância, através dos recursos pré colocados em cache, economizar o consumo de banda quando acessar a aplicação em um dispositivo móvel.

Esse trabalho também visa proporcionar aos futuros usuários da aplicação o controle e o gerenciamento de suas periodizações de treinamento, sessões de treino e exercícios físicos.

1.3 Objetivo

O objetivo geral desse trabalho é desenvolver um protótipo de um aplicativo Web que funcione on-line e off-line.

Os objetivos específicos deste trabalho são desenvolver um protótipo de uma aplicação Web que funcione on-line e eventualmente off-line (durante uma indisponibilidade de conexão/rede) para o gerenciar e controlar as sessões de treino e os exercícios físicos de um treinamento desportivo, através da utilização de tecnologias como *Service Worker* e *IndexedDB* para gerenciamento de recursos dinâmicos.

Como consequências desse gerenciamento, o sistema controlará aspectos de escolha da atividade desportiva, periodização do treinamento (microciclos) e sessões de treinamento que constituem a parte prática a ser de controlada pelo protótipo.

Este trabalho procura guardar um histórico das sessões de treino praticadas pelos futuros usuários, permitindo que eles possam acompanhar seus resultados mesmo quando eventualmente estiverem utilizando o protótipo off-line.

1.4 Metodologia

A metodologia utilizada na elaboração teórica deste trabalho foi a pesquisa bibliográfica para documentação e diagramação, além das metodologias práticas sobre desenvolvimento de software e desenvolvimento de aplicativos Web.

Foram utilizadas fontes impressas e digitais durante a elaboração desse trabalho.

1.5 Organização do trabalho

Este trabalho é composto por cinco capítulos. O primeiro introduziu o tema, delimitando-o, justificando sua escolha e mostrando o objetivo que motivou a realização desse trabalho, além de incluir informações sobre a metodologia utilizada em sua confecção.

O segundo destina-se ao referencial teórico desse trabalho contendo diferenciações acerca das categorias de aplicativos, vantagens e desvantagens de aplicações nativas vs aplicações Web e fornece uma visão introdutória sobre quais tecnologias permitem a utilização Web off-line. Ele, também, introduz o conceito de *Progressive Web Apps*.

O terceiro capítulo apresenta conceitos sobre as tecnologias recomendadas para o desenvolvimento de *WebApps* que podem funcionar eventualmente off-line. Nesse capítulo noções técnicas sobre HTML5 e JavaScript são explicadas e exemplificadas em código.

No quarto capítulo é apresentado a documentação, modelagem, codificação e interfaces principais do protótipo que foi desenvolvido.

Enquanto o quinto capítulo reserva-se às considerações finais do autor sobre a realização desse trabalho.

2 REFERÊNCIAL TEÓRICO

2.1 O que são aplicativos

Um aplicativo computacional pode ser definido, de acordo com Roger S. Pressman (2011, p. 34) como um: “programa¹ sob medida que soluciona uma necessidade específica”. Um *app*² tem como objetivo, em uma conceitualização inicial, fazer a interação do hardware e do software com os sistemas/usuários que irão utilizá-los para resolver tarefas e trabalhos específicos em um computador. A Figura 1 ilustra essa interação.

Figura 1: Esquema de acesso de programas aplicativos às camadas de um sistema de computação



Fonte: Elaborado pelo autor.

Nesse sentido, Silberschatz, Korth e Sudarshan (2006, p. 4) esclarecem que:

O hardware [...] fornece os recursos básicos de computação. Os programas aplicativos – processadores de texto, planilhas eletrônicas, compiladores e navegadores *Web* – definem as maneiras em que esses recursos são usados para resolver os problemas de computação dos usuários. Pode haver muitos usuários diferentes (pessoas, máquinas, outros computadores) tentando resolver problemas diferentes. Da mesma forma, pode haver muitos programas aplicativos diferentes.

Como foi citado, podem haver usuários e aplicativos com necessidades diferentes e, ainda, consumindo recursos computacionais em ambientes diferentes. Atualmente, os ambientes mais comuns que os usuários encontram aplicativos sendo executados são no **desktop** ou na **Web**.

Aplicações **desktop** são aquelas que, principalmente, rodam instaladas em um sistema operacional (MANIERO, 2017).

¹ Um programa (em inglês, *software*) é uma sequência de instruções específicas que descrevem uma tarefa a ser realizada por um computador, na manipulação, redirecionamento ou modificação de dados, de maneira lógica. Disponível em: <<http://www.ufpa.br/dicas/progra/protipos.htm>>. Acesso em: 13 de nov. 2017.

² *App* é a forma contraída da palavra “*application*” que significa aplicativo em inglês.

Os dispositivos que utilizam as aplicações desktop mais conhecidos são: máquinas servidoras, computadores de mesa (desktop) ou notebooks e as plataformas mais comuns em que podem ser encontradas são no *Windows* da Microsoft e em sistemas operacionais baseados em Unix, como é o caso do Linux, dentre outros.

É importante mencionar que com a incrível expansão da tecnologia móvel observada nos últimos anos, conforme ilustra a Figura 2, novas categorizações para os aplicativos surgiram. Dentre as que merecem destaque estão os *mobile apps*, *native apps*, *WebApps* e *hybrid apps*.

Figura 2: Desktop vs Mobile: quota no mercado mundial



Fonte: Disponível em <<http://gs.statcounter.com/os-market-share/desktop/worldwide/#monthly-201610-201709-bar>>. Acesso em: 25 de out. 2017.

Mobile apps ou aplicativos móveis derivam justamente do surgimento dos dispositivos móveis, conforme ressaltam Laudon e Laudon (2011, p. 110):

“[...]novas plataformas computacionais digitais móveis surgiram como alternativa aos PCs e grandes computadores. Essas plataformas móveis como o próprio nome diz, são capazes de manterem-se operacionais mesmo em movimento.

Os dispositivos móveis mais conhecidos atualmente são smartphones, *netbooks*, dispositivos vestíveis, GPS e leitores digitais e as plataformas³ mais comumente encontradas para estes são o *Android* da Google, *IOS* da Apple e *Windows Phone* da Microsoft⁴.

Visto isso, consegue-se inferir que os aplicativos móveis são aqueles que rodam em cima de suas plataformas para seus respectivos dispositivos móveis. Esse pensamento leva ao entendimento que tanto a aplicação desktop tradicional quanto um *mobile app* funcionam de maneira nativa no dispositivo e na plataforma na qual encontram-se instalados/configurados.

Deste ponto em diante, para evitar confundir o leitor e incorrer em erros de terminologia técnica, o termo *native app* será utilizado neste trabalho para tratar tanto sobre aplicativos desktop quanto para aplicativos *mobile*.

Um *native app*, reiterando toda a ideia anterior, pode ser definido como uma aplicação que rode “direto no sistema operacional e, na maioria dos casos, são escritas nativamente para a plataforma específica...” (LOPES, 2013, p. 14 e 15).

Devido à tamanha especificidade na construção (desenvolvimento e implementação) desse tipo de aplicação, elas conseguem tirar vantagens dos recursos de hardware e software daquele dispositivo em particular, processar e extrair com mais eficácia os dados computacionais daquele ambiente. Por exemplo utilizando GPS e câmera (ROUSE, 2013).

Imediatamente em contraste e/ou conjunto com os aplicativos nativos existe uma categoria de aplicativos que está inerentemente vinculada ao funcionamento Web, Pressman (2011, p. 37) refere-se a eles como *WebApps*. Seguindo seu raciocínio, uma *WebApp* engloba desde uma única página Web até um site completo, ou seja, essa categoria de aplicativos contempla informações da Internet - de uma Intranet ou Extranet - que são entregues ao usuário através da interface de um **navegador Web** (ROUSE, 2011).

O navegador é o núcleo do ambiente Web, isto é, a Web “roda em cima do navegador Web padrão” (MANIERO, 2017).

³ Disponível em <<http://gs.statcounter.com/os-market-share/mobile-tablet/worldwide/#monthly-201610-201709>>. Acesso em: 26 de out. 2017.

⁴ Apple, Google e Microsoft são empresas de Tecnologia da Informação e as três marcas mais valiosas do mundo <<http://epocanegocios.globo.com/Marketing/noticia/2017/09/apple-google-e-microsoft-sao-marcas-mais-valiosas-do-mundo.html>>. Acesso em: 27 de out. 2017.

Existem organizações que fornecem meios de programar aplicativos para Web de forma padronizada, já que (como é visto na seção 2.3), a Web *per se* tem caráter descentralizado e distribuído, numa linguagem mais acessível. Web traduz-se em independência, portabilidade e colaboração em tempo real e com alcance global. Logo, se as aplicações Web têm tamanho dinâmico, é necessário recomendar especificações que adequem essas aplicações para funcionar em qualquer parte sua. As organizações atuais que desenvolvem as especificações são a *World Wide Web Consortium (W3C)*⁵ e a *Web Hypertext Application Technology Working Group (WHATWG)*⁶, enquanto as companhias mais conhecidas que implementam nos navegadores essas especificações são o Google, Mozilla *Foundation*⁷ e a Microsoft.

Outra característica que vale a pena mencionar para uma *WebApp* é que deve obrigatoriamente ser disponibilizada sob demanda pelo menos a primeira vez, via HTTP⁸ (MANIERO, 2017). O primeiro acesso deve invariavelmente fazer uma requisição Web para se caracterizar como uma aplicação Web.

Após essas definições, para uma visualização mais clara, a seção 2.2 evidencia as vantagens e desvantagens de aplicativos ambientados nativamente contra aqueles ambientados na Web.

2.2 Aplicações nativas vs Web - vantagens e desvantagens

A grande diferença entre *native Apps* e *WebApps* que geralmente se discute é que um “App dá melhor acesso e integração ao hardware e à plataforma nativa do aparelho, enquanto que a Web traz independência de plataforma e portabilidade” (LOPES, 2013, p. 13).

Ambos os ambientes possuem suas vantagens e desvantagens, que tornarão o processo de utilização e adoção pelos usuários ou durante criação de aplicativos pelos desenvolvedores de software, mais favorável a um desses lados. A Tabela 1 e a Tabela 2 procuram comparar as vantagens de uma aplicação nativa frente a uma aplicação Web.

⁵ A W3C é a principal organização de padronização da *World Wide Web*.

⁶ O WHATWG é um grupo de trabalho de pessoas interessadas na evolução do HTML e das tecnologias relacionadas.

⁷ A Mozilla *Foundation* é uma organização sem fins lucrativos desenvolvedora do navegador Firefox e de todos os produtos da Mozilla. Disponível em: <<https://www.mozilla.org/pt-BR/foundation/>>. Acesso em: 30 de out. 2017.

⁸ HTTP é um acrônimo para (em inglês) *Hypertext Transfer Protocol*. Um dos protocolos utilizados na Web.

Tabela 1: Vantagens e desvantagens de uma aplicação nativa

NATIVO		
	VANTAGENS	DESVANTAGENS
INTEGRAÇÃO COM HARDWARE E PLATAFORMA	O <i>native app</i> tem acesso direto ao <i>hardware</i> do aparelho e a recursos do sistema operacional. Consegue se integrar com funções avançadas e a outras <i>apps</i> (performance).	-
SEGURANÇA E PRIVACIDADE	-	Em geral estas aplicações tendem ser um pouco menos seguras para o usuário devido ao grande poder de acesso às funcionalidades de hardware e software.
USABILIDADE E VISUAL	Integração visual da <i>app</i> com a plataforma em si. Ela tem a cara da plataforma e é familiar para o usuário.	
INSTALAÇÃO E DISTRIBUIÇÃO	-	Precisa ser instalada e isso, geralmente, envolve uma loja do fabricante onde a <i>app</i> será disponibilizada. Quando há atualização, uma <i>app</i> precisa ser instalada novamente.
MULTIPLATAFORMA	Trabalha off-line.	O desenvolvimento de <i>native apps</i> normalmente é focado nas plataformas que estão em maior evidência (alta).

Fonte: Adaptado de Lopes (2013, p. 13-16) e Maniero (2017).

Tabela 2: Vantagens e desvantagens de uma aplicação Web

WEB		
	VANTAGENS	DESVANTAGENS
INTEGRAÇÃO COM HARDWARE E PLATAFORMA	Uma <i>WebApp</i> roda em cima de um navegador Web.	Por rodar dentro do navegador, o HTML, CSS e JavaScript é interpretado em um processo relativamente mais lento do que no <i>native app</i> (fluidez e performance).
SEGURANÇA E PRIVACIDADE	Restrições de segurança do navegador são fortes e a chance de acontecer algo de ruim é bem pequena. O usuário está mais protegido abrindo um site Web do que instalando uma <i>app</i> em seu aparelho.	-
USABILIDADE E VISUAL	Os sites ou <i>WebApp</i> costumam ter um estilo mais ligado à identidade visual da marca e da empresa, do que da plataforma de acesso.	-
INSTALAÇÃO E DISTRIBUIÇÃO	Enquanto a aplicação Web estiver na rede (on-line) o usuário poderá acessá-la, também em sua versão mais recente quando navega.	-
MULTIPLATAFORMA	O grande apelo da Web é ser independente de plataforma. Navegadores Web estão disponíveis em quase todos os aparelhos computacionais atualmente.	Não trabalha off-line (pelo menos o primeiro acesso deve ser feito on-line)

Fonte: Adaptado de Lopes (2013, p. 13-16) e Maniero (2017).

Após todas essas diferenciações, conforme suscita Lopes (2013, p. 19 e 20):

É importante conhecer bem o escopo do projeto e as expectativas do seu grupo de usuários. Às vezes, por mais que *Web* faça mais sentido, uma *App* vai trazer a experiência final mais apropriada naquele caso. Ou não, muitas vezes as restrições das *Apps* com relação à *Web* são fatores mais determinantes.

De acordo com o que foi comparado nas tabelas e também na citação anterior consegue-se deduzir que o aplicativo **nativo** estará muito mais integrado com o dispositivo e o sistema operacional em que estiver instalado, refletindo em performance e disponibilidade off-line (na maioria das vezes) para o usuário, enquanto a **Web** será tratada como uma aplicação acessível, portátil e segura tendo o auxílio de uma rede universal de acesso.

Para este último fato torna necessário uma explicação de como essa rede funciona, para ajudar a escolher quando será melhor uma aplicação utilizar-se dos mecanismos de funcionamento Web.

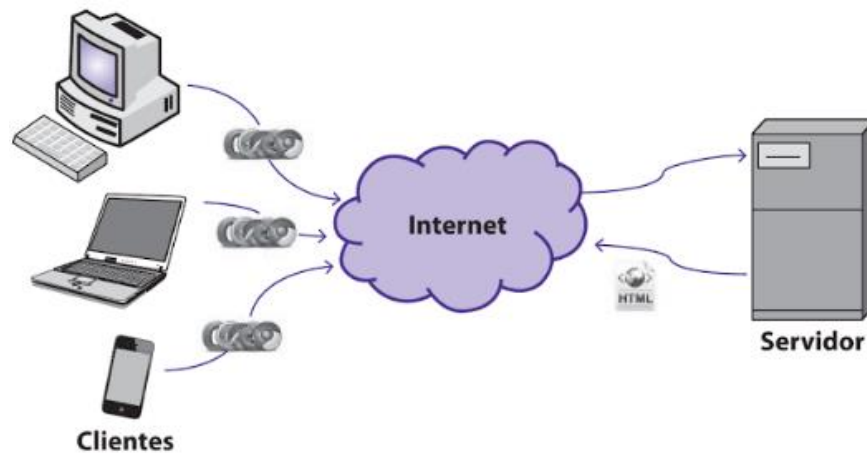
2.3 Funcionamento tradicional da Web

A Web é uma plataforma, global e aberta, de distribuição e navegação de conteúdo. Em suma pode-se dizer que:

A Web é a soma de uma rede em cima de HTTP com a capacidade de se criar links entre páginas, em um formato padronizado, o HTML. Web é estar conectado na Internet e carregar suas páginas de um servidor em qualquer lugar do mundo. E, seguindo links, navegar para todos os cantos (LOPES, 2013, p. 22).

Na prática as páginas Web são baixadas do servidor com todas as suas dependências e arquivos (HTML, CSS e Scripts) de modo a ficarem disponíveis para o acesso dos usuários (LOPES, 2013, p. 15).

Seu funcionamento básico consiste em uma **aplicação** localizada em algum computador remoto que esteja disponível (on-line) e possa ser acessada por usuários através de seus computadores usando um navegador (browser). Teoricamente, se diz que esse modelo é considerado um **sistema distribuído**, pois parte do processamento ocorre no computador do **cliente** e a outra parte acontece numa máquina denominada **servidora** (modelo **cliente-servidor**) (MILETTO e BERTAGNOLLI, 2014, p. 4), conforme a Figura 3:

Figura 3: Estrutura cliente-servidor

Fonte: Miletto e Bertagnolli (2014, p. 5).

Assim, toda concepção de aplicações Web está fundamentada em torno de uma rede que fique disponível às requisições dos usuários⁹.

Porém, devido aos diversos fatores externos, a rede pode ter sua disponibilidade parcial ou totalmente interrompida. Exemplos do cotidiano são: casos de falta de cobertura de área da operadora contratada que fornece os dados móveis para o dispositivo do usuário, situações de trabalho que restrinjam o acesso à internet ou ambientes que, apesar de ter sinal *WiFi* não o disponibilizem a todos. Ainda existem casos de conexão extremamente ruins, infraestrutura de rede precária, entre outros fatores que venham a ser limitantes de conectividade (MAHEMOFF, 2013).

Como fazer quando nestes casos em que a utilização da aplicação pelo usuário não possa (ou em teoria não deveria) ser interrompida e mesmo assim o usuário gostaria de utilizar a aplicação sem instalá-la toda em seu dispositivo?

Dados mostram que muito pouco das aplicações mobile instaladas têm número relevante de usuários ativos¹⁰.

A seção 2.4 reserva-se a mostrar propostas de solução para esse suposto cenário.

⁹ Uma explicação didática mais detalhada pode sobre o funcionamento da Web ser vista em: <https://developer.mozilla.org/pt-BR/docs/Aprender/Getting_started_with_the_web/Como_a_Web_funciona> Acesso em: 31 out. 2017.

¹⁰ Estatísticas sobre os dados de aplicações mobile podem ser vistas no artigo em: <<http://andrewchen.co/new-data-shows-why-losing-80-of-your-mobile-users-is-normal-and-that-the-best-apps-do-much-better/>> Acesso em: 31 out. 2017.

2.4 Tecnologias Web que permitem a utilização off-line

A chegada de aplicativos categorizados como *hybrid app* trouxe a proximidade das linguagens de marcação, estilização e *scripting*, originalmente utilizadas na programação Web, para as aplicações nativas, conforme define Sérgio Lopes (2016, p. 7): “uma aplicação híbrida é uma aplicação normal, apenas escrita em linguagens comuns a desenvolvedores Web. Não é Web. Isso é importante: Web não é HTML, CSS e JavaScript”.

Contudo, ainda que o *hybrid app* possa acessar a internet, continua sendo uma aplicação nativa que deve ser baixada de alguma loja de aplicativos e apenas utiliza-se das tecnologias de desenvolvimento Web para portar a aplicação aos diversos dispositivos, em caráter de multiplataforma.

Frente a questão da utilização de uma aplicação, quando a rede se tornar indisponível (parcial ou totalmente **off-line**), pelos mais diversos fatores, seria interessante uma abordagem que alie a portabilidade da Web e as vantagens do *native app*. Algumas inovações tecnológicas que contêm diversas funcionalidades para construção de aplicações Web que funcionam off-line surgiram. Elas serão mencionadas a seguir antes de se aprofundar naquelas que serão recomendadas para atender o objetivo desse trabalho.

As tecnologias: *Cookies*, *Application Cache*, *Web Storage*, *Web SQL Database* permitem o armazenamento off-line do conteúdo Web estático ou de dados levemente dinâmicos. Estas são tratadas na seção 2.4.1.

2.4.1 Armazenamento de conteúdo Web estático e levemente dinâmico

Cookie: é um pequeno arquivo texto que é armazenado na máquina do usuário pelo browser e é enviado para o servidor à cada requisição, de modo a processar alguma informação relevante. O mecanismo de *cookie* foi inventado em meados de 1997 por Lou Montulli, um funcionário da Netscape, e eles podem ser usados para armazenamento local persistente de pequenas quantidades (4KB) de dados (GOMES, J., 2016, p. 116). *Cookies* são utilizados principalmente para três propósitos: **gerenciamento de sessão**, como por exemplo: armazenamento de informações de *login*, carrinho de compras, pontuação de jogos, etc. **Personalização** como preferências do usuário, temas e outras configurações e no

rastreamento para gravação e análise de comportamento do usuário (MOZILLA DEVELOPER NETWORK, 2017f).

Application cache: é um arquivo manifesto¹¹ do HTML5 que foi recomendado para a W3C pelo W3C *Working Group Note* em 30 de maio de 2008 (VAN; HICKSON, 2008). Esse arquivo, conforme descreve a WHATWG, fornece aos desenvolvedores um mecanismo de cache que lista quais arquivos são necessários para a aplicação Web funcionar off-line e em quais situações o browser do usuário, através da **interface de cache do aplicativo** (*AppCache*), irá manter uma cópia desses arquivos para uso off-line. (WHATWG, 2017).

O *Application cache* é referenciado na página Web através do atributo “*manifest*” do HTML5 (Figura 4).

Figura 4: Referenciando o *Application cache* no código HTML

```

1 <!DOCTYPE html>
2 <html>
3 <body>
4 <!--PÁGINA COM O CONTEÚDO WEB -->
5 </body>
6 </html>

```

Fonte: Elaborado pelo autor.

A estrutura do arquivo manifesto pode ser vista na Figura 5.

Figura 5: Estrutura interna do arquivo *Application cache*

```

1 CACHE MANIFEST
2 #v 01
3 CACHE:
4 css/estilos.css
5 js/lib/jquery.js
6 img/edit.svg
7 NETWORK:
8 *

```

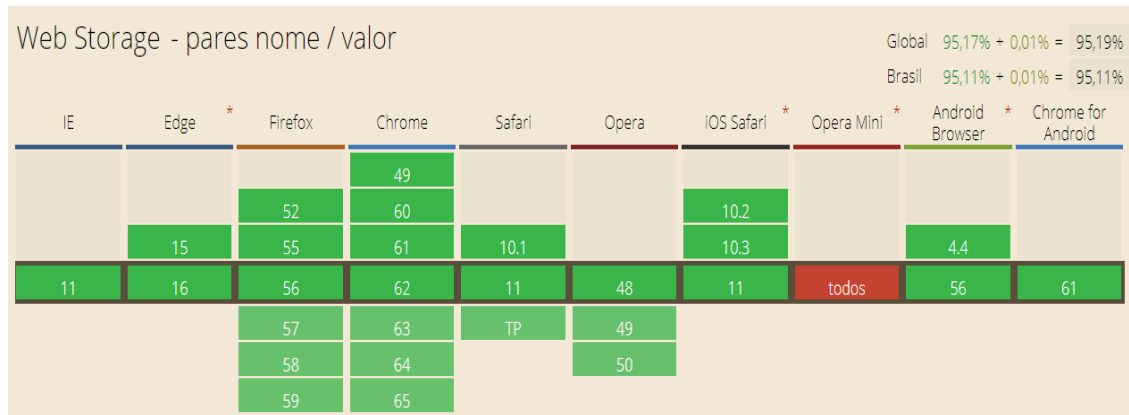
Fonte: Elaborado pelo autor.

O arquivo contém basicamente texto, sempre iniciando a primeira linha com “CACHE MANIFEST”, seguido do número da versão do documento. “CACHE” refere-se aos outros arquivos a serem armazenados localmente no browser (colocados em cache) e “NETWORK” indica o que buscar da internet. No caso da Figura 5, o asterisco após *NETWORK*: representa que o *AppCache* sempre buscará tudo da rede e se não encontrar nada, então buscará do cache anteriormente salvo no navegador do usuário.

¹¹ O arquivo manifesto é um documento do HTML5 que contém uma série de referências a outros arquivos Web estáticos (HTML, CSS e Imagens).

Outra tecnologia que permite que aplicativos Web armazenem dados localmente é a chamada **Web Storage**, uma API¹² que está presente em aproximadamente noventa e cinco por cento dos browsers utilizados na Web, conforme apresentado na Figura 6.

Figura 6: API Web Storage – navegadores suportados e porcentagem de uso



Fonte: Adaptado de Alex Deveria (2017).

Conforme a Figura 6 ilustra, o Web Storage armazena valores baseados em pares de nome/valor e pode ser acessado através de código JavaScript. O padrão atual utilizado dessa especificação teve sua última atualização pela WHATWG no dia 09 de novembro de 2017 (WHATWG, 2017b).

Existem dois objetos da especificação Web Storage que contém os métodos responsáveis pelo armazenamento de dados localmente, eles são o **local storage** e o **session storage**:

- **Local storage**: armazena dados sem uma data de expiração, isto é, os dados permanecerão salvos ainda que o browser seja fechado e estarão disponíveis até que sejam atualizados ou limpos (MOZILLA DEVELOPER NETWORK, 2017c). Um exemplo de uso do Local Storage utilizando JavaScript é visto na Figura 7.

Figura 7: Exemplo de codificação do objeto local storage

```

1 // Armazenamento com Local Storage
2 localStorage.setItem("nome", "Roger");
3 //Recuperar
4 document.getElementById("resultado").innerHTML = localStorage.getItem("nome");

```

Fonte: Elaborado pelo autor.

¹² API é o acrônimo para (em inglês) *Application Programming Interface*, uma lista de rotinas e procedimentos os quais implementam diversas bibliotecas que fornecem recursos adicionais para um aplicativo. Disponível em: <<https://canaltech.com.br/software/o-que-e-api/>>. Acesso: 13 de out. 2017.

- **Session storage:** armazena os dados de forma igual ao objeto anterior, a diferença consiste que os dados têm permanência de uma única sessão, sendo apagados assim que o usuário fecha a aba ativa do browser que utiliza esse método. Um exemplo de uso do *Session Storage* utilizando JavaScript é visto na Figura 8.

Figura 8: Exemplo de codificação do objeto *session storage*

```

1 ▾ if (sessionStorage.contadorDeClique) {
2   sessionStorage.contadorDeClique = Numero(sessionStorage.contadorDeClique) + 1;
3 ▾ } else {
4   sessionStorage.contadorDeClique = 1;
5   }
6 document.getElementById('resultado').innerHTML = "Você clicou o botão " +
7 sessionStorage.contadorDeClique + " vez(es) nessa sessão.";

```

Fonte: Adaptado de W3Schools (2017).

Outro modo de se persistir informações localmente no navegador do usuário é através da **API Web SQL Database** do HTML5. Esta API foi especificada para W3C em 18 de novembro de 2010 (VAN; HICKSON, 2008) e como Mark Pilgrim (2012) diz “fornece uma fina camada em torno de uma base de dados SQL”. Assemelha-se com uma programação de banco de dados *back-end* tradicional, porém para o *WebDB* (como é formalmente conhecida) a programação é via *JavaScript*.

Contudo, apesar de ter processamento assíncrono, o *WebDB* não está tão presente nos navegadores atuais (como é visto na seção 3.3) e, também, essa API não é recomendada para o desenvolvimento da aplicação proposta nesse trabalho.

2.4.2 Progressive Web Apps

Progressive Web Apps (PWAs) são um conjunto de práticas recomendadas para proporcionar aos seus usuários uma melhor experiência usando tecnologias Web (JUAREZ FILHO, 2016).

O termo foi cunhado em 2015, por Frances Berriman e um engenheiro do navegador Google Chrome, Alex Russel Demoed, para descrever aplicativos que tiram vantagem das novas características suportadas pelos navegadores modernos. Essas aplicações não são empacotadas nem são distribuídas através de lojas, elas são apenas *websites* que progressivamente se tornaram *Apps* (DEMOED, 2015).

Com base no exposto, de uma maneira bastante simplista, pode-se dizer que PWAs são *WebApps* que utilizam as mais modernas capacidades dos navegadores para entregar aos usuários experiências similares aos aplicativos nativos.

Elas são construídas baseadas na *Promises API* e na *Fetch API* do *JavaScript* (GOOGLE DEVELOPERS, 2017), além de ser necessário que o navegador utilizado tenha suporte ao *Service Workers* (SWs). A explicação acerca do funcionamento técnico desses itens é abordada no capítulo 3.

A principal **vantagem** que o usuário tem ao utilizar uma PWA é a de não precisar se comprometer a fazer o download de um aplicativo antes mesmo de saber se valerá a pena ou não. A Tabela 3 apresentada outras vantagens referentes à utilização de PWAs.

Tabela 3: Vantagens da utilização de *Progressive Web Apps*

PWA	
VANTAGENS	DESCRIÇÃO
Progressivo	Funciona para qualquer usuário, independentemente do navegador escolhido, pois é criado com aprimoramento progressivo como princípio fundamental.
Responsivo	Se adequa a qualquer formato: desktop, celular, <i>tablet</i> ou o que for inventado a seguir.
Independente de conectividade	Aprimorado com <i>Service Workers</i> para trabalhar off-line ou em redes de baixa qualidade.
Semelhante a aplicativos	Parece com aplicativos para os usuários, com interações e navegação de estilo de aplicativos, pois é compilado no modelo de <i>shell</i> de aplicativo.
Atual	Sempre atualizado graças ao processo de atualização do <i>Service Worker</i> .
Seguro	Fornecido via HTTPS para evitar invasões e garantir que o conteúdo não seja adulterado, exceto para o <i>localhost</i> .
Descobrível	Pode ser identificado como "aplicativo" graças as especificações W3C e ao escopo de registro do <i>Service Worker</i> , que permite que os mecanismos de pesquisa os encontrem.
Reenvolvente	Facilita o reengajamento com recursos como notificações <i>push</i> , sincronização em segundo plano, etc.
Instalável	Permite que os usuários adicionem os aplicativos na tela inicial do dispositivo, sem precisar acessar uma loja de aplicativos.
Linkável	Compartilhado facilmente por URL ¹³ , não requer instalação complexa.

Fonte: Adaptado de Pete Lepage (2017).

Portanto, o capítulo 3 trata sobre as tecnologias que melhor se enquadram para elaboração do proposto nesse trabalho.

¹³ URL é o acrônimo para *Uniform Resource Locator* sendo o formato de atribuição universal para localizar um recurso na Internet. Maiores informações podem ser vistas em: < <http://br.ccm.net/contents/288-o-que-e-um-url>>. Acessado em: 25 de nov. 2017.

3 TECNOLOGIAS RECOMENDADAS PARA O DESENVOLVIMENTO DE PWAS

Conforme mostrado anteriormente, diversas podem ser as tecnologias adotadas para o armazenamento local no browser e persistência de dados off-line. Muitas delas podem e algumas vezes devem ser utilizadas em conjunto. No entanto a maioria se torna inflexível quando tem que lidar com dados e conteúdos que são manipulados dinamicamente nos *websites*.

A recomendação do Mark Cohen (2017) acerca de quais tecnologias utilizar para o armazenamento de *Progressive Web Apps* é a seguinte: “para PWAs, você pode armazenar em cache recursos estáticos, compondo o *shell* do aplicativo (arquivos JS/CSS/HTML) usando a Cache API e preenchendo os dados da página off-line usando o *IndexedDB*.”

Além das anteriormente recomendadas, o recurso do HTML5 *application manifest*, bem como **processamento assíncrono** baseado em **promessas** do *JavaScript*, *Service Workers* e um relato mais aprofundado dos eventos que permitem o armazenamento no cache do dispositivo, compõem o conteúdo técnico desse capítulo.

3.1 HTML5

“HTML, ou *HyperText Markup Language*, é uma linguagem de marcação utilizada para criar páginas acessadas a partir de um navegador. A característica principal dessas páginas é que elas utilizam hipertexto para viabilizar a navegação.” (MILETTO e BERTAGNOLLI, 2014, p. 62).

Seguindo essas proposições, o **HTML** não é uma linguagem de programação, mas sim uma **linguagem de marcação** das páginas que estão na Web. O **hipertexto** é (além dos próprios textos) imagens, tabelas, formulários, vídeos e links que possibilitam a navegação.

O HTML5 trouxe algumas inovações para o conteúdo chamado hipertexto, no sentido de diversas padronizações e especificações que ajudam na portabilidade, independência e compatibilidade entre aplicações e dispositivos (LOPES, 2013, p. 25 e 26). Uma dessas especificações diz respeito ao *Application Manifest* que integra a experiência progressiva no desenvolvimento de aplicações Web.

3.1.1 *Application Manifest*

O manifesto dos aplicativos web é um arquivo JSON que permite controlar como o aplicativo Web ou site é exibido para o usuário em áreas que normalmente se espera ver aplicativos nativos (por exemplo, a tela inicial de um dispositivo), como definir o que o usuário pode inicializar e o visual durante a inicialização (GAUNT; KINLAN, 2017).

Tendo em vista o citado pelos autores, o arquivo JSON¹⁴ tem propriedades que definem o comportamento de inicialização da PWA, tornando-a mais parecida com o comportamento padrão de inicialização de uma *app* nativa. Um exemplo é a possibilidade de adicionar um atalho da PWA diretamente para a tela inicial do dispositivo do usuário. Uma observação técnica é que apesar de ser possível usar um *application manifest* em qualquer site, ele é obrigatório para PWAs. A Figura 9 mostra como o arquivo do *application manifest* é referenciado no código HTML.

Figura 9: Referenciando o arquivo *application manifest*

```
<link rel="manifest" href="/manifest.json">
```

Fonte: Mozilla Developer Network (2017h).

O *application manifest* é implementado em páginas HTML usando uma *tag* de `<link>` no cabeçalho do documento que referencia onde o arquivo está localizado.

3.2 JAVASCRIPT

De acordo com Pedro Remoaldo (2008, p. 35) “a linguagem *JavaScript* é uma linguagem de *scripting* interpretada que está disponível de forma gratuita na maioria dos browsers modernos.”

Segundo o autor afirma o *JavaScript* possui scripts que são uma sequência lógica de comandos de programação e tarefas e, que podem ser **interpretados** e/ou embutidos em outros programas, no caso os browsers usados na internet.

Os scripts dessa linguagem tornam possível o desenvolvimento no modelo chamado de **melhoria progressiva** através de APIs que são construídas pelos desenvolvedores de

¹⁴ JSON é um acrônimo para “*JavaScript Object Notation*” (Notação de Objetos *JavaScript* – em português) é uma formatação leve para troca de dados computacionais.

WebApps, numa filosofia de design que se centra em fornecer as funcionalidades essenciais da aplicação desenvolvida para o maior número possível de usuários. Os usuários que utilizam browsers mais modernos oferecem a melhor experiência possível. A detecção de recursos que determina quais navegadores conseguem lidar com as funcionalidades mais modernas disponíveis (ou não) e trata dos recursos faltantes com a própria codificação *JavaScript* (MOZILLA DEVELOPER NETWORK, 2017g).

Uma característica da melhoria progressiva é o uso de chamadas assíncronas para *websites* com conteúdo dinâmico, para quando o recurso estiver implementado no navegador do usuário.

3.2.1 Programação assíncrona

As chamadas assíncronas “executam em paralelo suas funções sem travar processamento das outras e principalmente sem bloquear o sistema principal”. (PEREIRA, 2014, p. 25).

Isto faz com que uma PWA tenha maior liberdade e dinamismo, principalmente durante o gerenciamento dos vários dados que serão processadas e entregues aos usuários na forma de informações, aumentando assim sua usabilidade. Elas desempenham papel essencial no armazenamento de conteúdo off-line para aplicações.

Funções assíncronas estão ativadas por padrão nos navegadores Google Chrome 55, Microsoft Edge versão 14342 e estão com desenvolvimento ativo no Mozilla Firefox e no Apple Safari (ARCHIBALD, 2017).

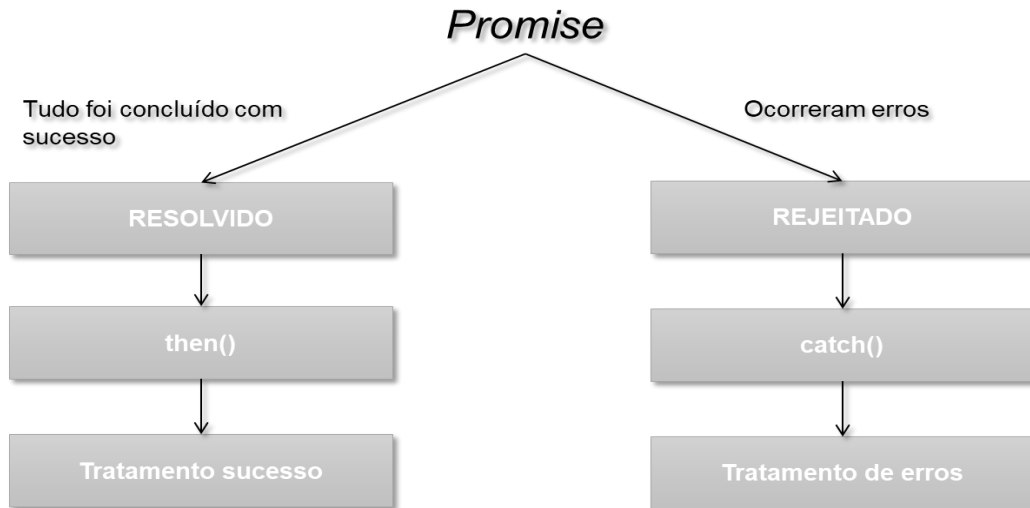
As funções assíncronas fazem forte uso de **promessas** do JavaScript.

3.2.2 Promises

Promises são objetos do JavaScript ES6 que auxiliam a trabalhar com operações assíncronas. “Este tipo de objeto aguarda a operação ser completada e oferece uma resposta positiva (resolvida) para quando realizada com sucesso, ou negativa caso algo tenha ocorrido algum erro no processo (rejeitada)” (PINHO, 2017, p. 161).

As *promises* têm 3 estados: **resolvido e rejeitado** foram explicados pela citação do autor Pinho e o estado **não resolvido** diz respeito à quando uma *promise* aguarda para ser processada. A Figura 10 ilustra seu comportamento.

Figura 10: Comportamento de uma *promise*



Fonte: Adaptado de Pinho (2017, p. 165).

Quando uma *promise* é resolvida, o método *then()* é automaticamente ativado e o desenvolvedor realiza o tratamento para o caso de sucesso, quando a ela é rejeitada entra em ação o método *catch()* deixando aberta a opção para o tratamento de eventuais erros no retorno de seu estado.

Figura 11: Codificação e sintaxe padrão de uma *promise*

```

1 let promise = new Promise(resolve, reject) => {
2   let resultado = true;
3   if (resultado){
4     resolve("deu tudo certo!");
5   } else {
6     reject("deu tudo errado!");
7   }
8 };
  
```

Fonte: Adaptado de Pinho (2017, p. 166).

Elas tiveram suas primeiras implementações nos navegadores a partir das versões Chrome 32, Opera 19, Firefox 29, Safari 8 e Microsoft Edge, estando ativadas por padrão.

3.2.3 Service Workers

Service Worker é um arquivo de script que fica armazenado dentro do projeto de um *website* e é executado pelo browser do usuário.

Service workers agem essencialmente como servidores proxy que ficam entre as aplicações web, o navegador e a rede (quando disponível). Eles têm a intenção (entre outras coisas) de permitir a criação de experiências offline eficazes, interceptando pedidos de rede e tomando medidas apropriadas com base no fato de a rede estar disponível e os ativos atualizados residirem no servidor. Eles também permitem o acesso a *push notifications* e APIs de sincronização de *background* (MOZILLA DEVELOPER NETWORK, 2017d, TRADUÇÃO NOSSA).

De acordo com o que foi exposto pode-se definir que o *Service Worker* permite controlar como as solicitações de rede para aplicações Web são tratadas. Ele é a evolução do *AppCache* (visto na seção 2.4), sendo inteiramente codificável em *JavaScript*. Outras características de seu funcionamento são que ele não tem acesso ao DOM¹⁵, pois é executado fora do escopo da página, além de ser uma melhoria progressiva e fazer uso intensivo de promessas, assuntos discutidos anteriormente.

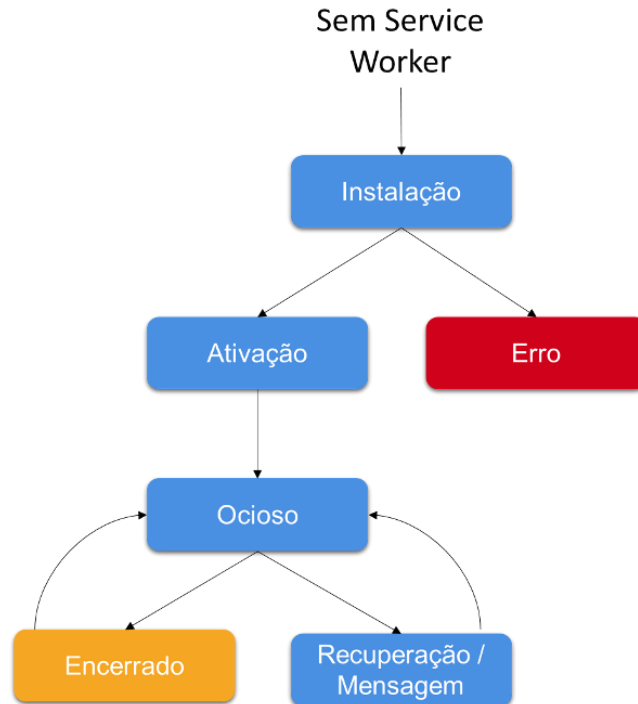
A especificação mais atual, recomendada para W3C, para *Service Workers*, data de 02 de novembro de 2017, tendo como desenvolvedor e editor principal, **Jake Archibald** engenheiro do Google (DEMOED *et al*, 2017).

Por questões de segurança durante a comunicação on-line de uma página Web, o SW só poderá ser implantado em sites que utilizem o protocolo HTTPS, contudo durante o desenvolvimento, poderá ser utilizado por meio do *localhost* em HTTP. Os navegadores que detêm essa tecnologia implementada são o Google Chrome desde sua versão 40, Mozilla Firefox versão 44 em diante e o Opera na versão 24 fornecendo apenas uma implementação básica do que foi especificado (MOZILLA DEVELOPER NETWORK, 2017e).

Resumidamente os *SWs* trazem o que é necessário para a experiência Web off-line e oferecem aos desenvolvedores controle total sobre essa experiência, como será demonstrado a seguir.

Primeiramente é necessário visualizar o ciclo de vida básico do *Service Worker*, lembrando que esse ciclo acontece totalmente separado da página da Web. A Figura 12 ilustra um fluxograma desse ciclo.

¹⁵ DOM é o acrônimo para *Document Object Model*, sendo uma convenção para representação de documentos Web (HTML, XHTML e XML).

Figura 12: Ciclo de vida do *Service Worker*

Fonte: Adaptado de Matt Gaunt (2017, TRADUÇÃO NOSSA).

Antes de utilizar (**instalar**) um SW em um *website* é necessário **registrá-lo** primeiro. A Figura 13 intenta mostrar como é a codificação para o registro do SW em uma página Web, mais especificamente utilizando o Google Chrome como navegador.

Figura 13: Registrando o *Service Worker* na página Web

```

1 if ('serviceWorker' in navigator) {
2   window.addEventListener('load', function() {
3     navigator.serviceWorker.register('/service-worker.js');
4   });
5 }
  
```

Fonte: Matt Gaunt (2017).

O código da Figura 13 verifica se o navegador tem suporte à essa tecnologia e adiciona um “escutador de eventos” na página, para que quando ela carregar a primeira vez, possa executar a função de registro do *Service Worker*, buscando onde o arquivo *JavaScript* foi referenciado.

Feito isso a próxima etapa do ciclo será a **instalação**, onde normalmente alguns ativos estáticos, são colocados em cache.

Se a etapa de registro for executada com êxito, então o SW é instalado. Isso implicitamente denota outro estado do ciclo de vida do *Service Worker*, se a instalação for bem-sucedida, existe a possibilidade de preencher o cache do navegador com os primeiros recursos para executar website off-line, se não, então apresentará um estado de **erro** (MOZILLA DEVELOPER NETWORK, 2017e).

A etapa de instalação pode ser executada conforme o código mostrado na Figura 14.

Figura 14: Evento de instalação do *Service Worker*

```
1 self.addEventListener('install', function(event) {
2   //Executar etapas de instalação
3 });
```

Fonte: Adaptado de Drifty Co (2017).

A próxima etapa é a **ativação**, que ocorre após a instalação ter sido bem-sucedida. No momento em que o usuário acessar uma página diferente ou atualizar a página do website, então o *Service Worker* começa a receber eventos **fetch** (proporcionados pela *Fetch API*). Isto significa que ele passará a controlar todas as páginas dentro de seu escopo (GAUNT, 2017).

O código exibido pela Figura 15 traz um exemplo prático de utilização de recursos pré-cacheados para obtenção de uma experiência off-line.

Figura 15: *Service Worker* - evento *fetch()*

```
1 this.addEventListener('fetch', function(event) {
2   event.respondWith(
3     caches.match(event.request)
4   );
5 });
```

Fonte: Mozilla Developer Network (2017e).

Após corretamente ativado, o SW entra em um de dois estados: “**encerrado**, para economizar memória, ou tratando eventos de **recuperação e mensagem** gerados pela página quando faz uma solicitação ou mensagem de rede.” (GAUNT, 2017).

O exposto relata os outros dois estados finais do ciclo de vida do *Service Worker*, que são reservados à espera de requisições de rede, quando essas acontecem. Essa é o ciclo de vida básico dessa tecnologia que é mais flexível e permite um controle mais fino sobre a experiência de criação de *WebApps* off-line.

No entanto, existem outros eventos que estão contemplados nos SWs. Um evento que é comum na utilização prática dessa tecnologia diz respeito a **atualização**. Esse evento pode ocorrer seja por força de novas implementações em um *website*, como criação de novas funcionalidades, melhorias na segurança, renovação de layout da página, modelo de negócios atualizado, etc.

3.3 Um pouco mais sobre armazenamento off-line

O armazenamento de ativos estáticos e dinâmicos para utilização off-line garante uma série de vantagens para o usuário que utilizar uma aplicação que foi concebida pensando nesse aspecto. Dentre as vantagens pode-se mencionar a redução no consumo de banda (o que implica em menores gastos com internet), possível ganho com velocidade, pois os recursos podem ser carregados dos dados salvos localmente em cache; disponibilidade de utilização, pois o usuário conseguirá transitar entre a utilização on-line/off-line sem comprometer a experiência do usuário para aquele aplicativo, entre diversas mais.

Foi recomendado anteriormente a adoção de *Progressive Web Apps* para criação de aplicativos, pois elas reúnem vantagens que são originárias de *apps* nativos e *WebApps* e fazem uso de *Service Workers*. Para se adequar corretamente ao armazenamento Web off-line, o Mark Cohen (2017) recomenda, além das anteriores citadas, as seguintes tecnologias:

- Para recursos endereçáveis por URL (páginas de sites, arquivos CSS e *JavaScript*, imagens), a *Cache Storage API*, que será vista na seção 3.3.1.
- Para todos os outros dados dinâmicos, a *IndexedDB API*.

Antes da descrição técnica acerca dos mecanismos de funcionamento dessas duas tecnologias recomendadas, se faz necessário uma breve recapitulação de tópicos chave para o entendimento sobre armazenamento de dados.

Modelo de dados: determina como eles são organizados internamente. Dados armazenados em tabelas com campos predefinidos, como é típico em sistemas de gerenciamento de bancos de dados baseados em SQL, são chamados de **dados estruturados**, sendo ideal para consultas flexíveis e dinâmicas. O *IndexedDB* utiliza esse modelo (COHEN, 2017).

Persistência de dados: os dados que são retidos entre sessões e guias/janelas de navegador em um determinado dispositivo têm a persistência chamada de **persistência de dispositivo**. O *Cache Storage* utiliza essa persistência. (COHEN, 2017). A Tabela 4 contrasta algumas variáveis relacionadas ao armazenamento de dados que são proporcionados por APIs específicas desenvolvidas para essa finalidade.

Tabela 4: Comparativo entre APIs para armazenamento de dados off-line

API	Modelo de Dados	Persistência	Compatibilidade de Navegadores %	Transações	Sincronicidade
Sistema de arquivos	<i>Stream de bytes</i>	Dispositivo	52%	Não	Assíncrono
Armazenamento local	Chave-valor	Dispositivo	93%	Não	Síncrono
Armazenamento de sessão	Chave-valor	Sessão	93%	Não	Síncrono
Cookies	Estruturado	Dispositivo	100%	Não	Síncrono
WebSQL	Estruturado	Dispositivo	77%	Sim	Assíncrono
Cache	Chave-valor	Dispositivo	60%	Não	Assíncrono
IndexedDB	Híbrido	Dispositivo	83%	Sim	Assíncrono
Armazenamento em nuvem	<i>Stream de bytes</i>	Global	100%	Não	Ambos

Fonte: Adaptado de Mark Cohen (2017).

Em conjunto com o exposto, os principais navegadores trabalham com uma quantidade de espaço específica destinada ao armazenamento de dados em cache ou em disco. Os navegadores Google Chrome e Mozilla Firefox, destinam até 6% e 10% do espaço livre em disco para armazenamento, respectivamente. No Safari para dispositivos móveis a cota vai até 50 MB, no *desktop* essa cota é ilimitada. Por fim, no Internet Explorer 10 ou posterior o armazenamento pode ser utilizado até 250 MB (KITAMURA, 2014).

Após essas análises, a seção 3.3.2 reserva-se as explicações práticas de utilização das APIs *Cache Storage* e *IndexedDB* para armazenamento de dados off-line.

3.3.1 *Cache Storage*

A *Cache Storage* API é uma tecnologia que está implementada dentro dos *Service Workers* e provê uma interface para o armazenamento de objetos em cache (MOZILLA

DEVELOPER NETWORK, 2017). Os recursos cacheados podem ser vistos no navegador através do nome que foi destinado pelo desenvolvedor para cada cache criado.

Comumente durante o evento de **ativação** do SW ocorre o gerenciamento do cache, isto é, apagar caches antigos para servir-se de ativos mais atualizados para o *website*. A manipulação durante esse evento é vista como uma boa prática de utilização dessa tecnologia.

Atualmente cada desenvolvedor poderá criar seus próprios padrões para cacheamento com os SWs. Porém, citando boas práticas de utilização para o *Cache Storage*, existem diversas metodologias de armazenamento de recursos em cache que são altamente recomendadas durante a implementação.

Jake Archibald, engenheiro do Google e membro da W3C, descreve essas boas práticas como a “**máquina de cache**”, relatando uma série de eventos e situação onde o armazenamento e controle de solicitações pelo SW e *Cache API* podem ser ideais (ARCHIBALD, 2017b).

- **Na instalação - como uma dependência:** todo o conteúdo estático do site (CSS, imagens, fontes, JS, modelos).
- **Na ativação:** para limpeza e migração seria um bom momento para processar migrações de schema no IndexedDB e exclusão de caches não utilizados.
- **Na interação do usuário:** caso não for possível que o site todo fique off-line, o usuário seleciona o conteúdo que quer disponível off-line.
- **Na sincronização em segundo plano:** para atualizações não urgentes e que ocorram com alguma regularidade.
- **Cache e depois retorno para rede:** para aplicações voltadas ao modo off-line, a maioria das solicitações serão processadas.

A Figura 16 exemplifica o código do último item da máquina de cache feito com SWs.

Figura 16: Código usado para cachear recursos com o *Cache Storage* API

```

1 self.addEventListener("fetch", function(event){
2   event.respondWith(
3     caches.match(event.request).then(function(response) {
4       return response || fetch(event.request);
5     })
6   );
7 });

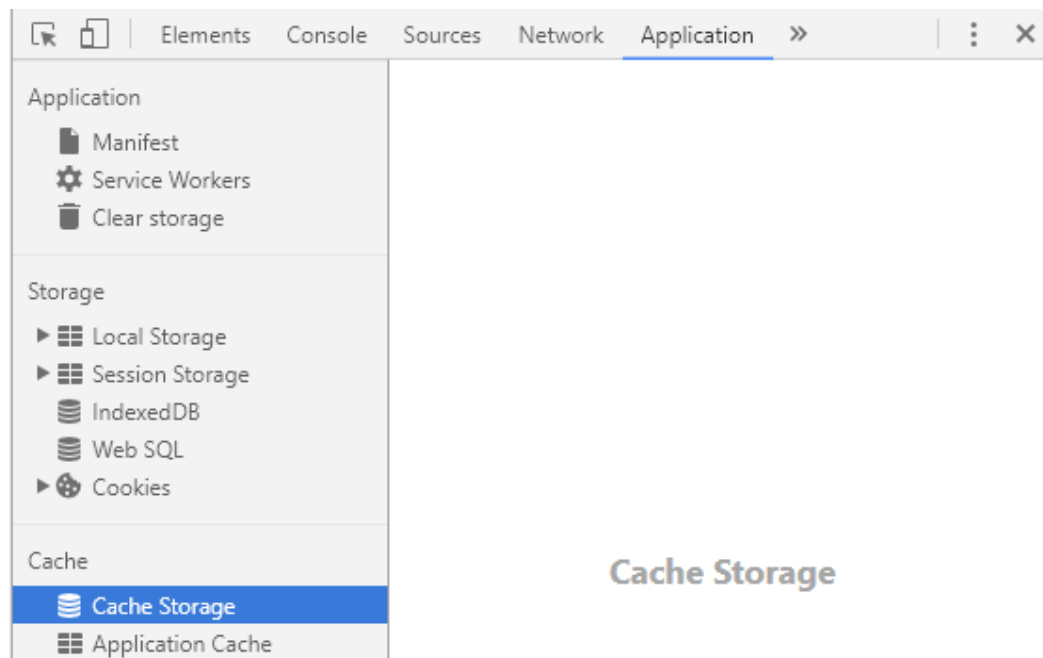
```

Fonte: Jake Archibald (2017b).

O código mostra na prática como situações onde o comportamento para obter os recursos armazenados primeiro no cache funciona. Os recursos **não encontrados** serão **procurados em rede** pelo evento *fetch()*.

O painel de ferramentas para desenvolvedores do navegador Google Chrome, permite inspecionar, modificar, e depurar caches criados com a Cache API (Service Worker), conforme aponta a Figura 17.

Figura 17: Seção *Cache Storage* no painel *Application* no Google Chrome



Fonte: Kayce Basques (2017).

O *Cache Storage* API é suportado no Google Chrome desde a versão 40, no Mozilla Firefox versão 44 em diante e no Microsoft Edge (MOZILLA DEVELOPER NETWORK, 2017).

3.3.2 *IndexedDB*

IndexedDB é uma API para armazenamento *client-side* de quantidades significantes de informações e buscas com alta performance por índices. Enquanto *DOM Storage* é útil para armazenamento de pequenas quantidades de dados, *IndexedDB* é a solução para grande porção de dados estruturados (MOZILLA DEVELOPER NETWORK, 2017b).

Diante da definição anterior, se entende que usando a API de base de dados indexada do navegador será possível manipular dados estruturados no lado do cliente, ou seja, manipular grandes quantidade de dados locais e com velocidade.

Os desenvolvedores da Mozilla, ainda, reforçam que o *IndexedDB* “permite criar aplicativos da Web com habilidades de consulta ricas, independentemente da disponibilidade da rede, esses aplicativos podem trabalhar on-line e off-line.”

Embasado nessa proposição e nas recomendações de uso anteriores, essa tecnologia complementa o desenvolvimento de PWAs, lidando com a parte dinâmica do armazenamento de dados off-line.

O funcionamento de sistemas gerenciadores de banco de dados relacionais tradicionais para operações de pesquisa, recuperação e alteração garante a confiabilidade nas operações sobre os dados. Nesse sentido o *IndexedDB* também suporta transações, que permitem tais operações e garantem o fácil manuseio, migração e adaptação para o desenvolvedor interessado em sua utilização (GOOGLE DEVELOPERS, 2017b).

O código para verificar se o navegador suporta o a API do *IndexedDB*, antes de sua utilização prática, pode ser visualizado na Figura 18.

Figura 18: Verificar o suporte para o *IndexedDB* nos navegadores

```

1 window.indexedDB = window.indexedDB || window.mzIndexedDB
2                   || window.webkitIndexedDB || window.msIndexedDB;
3
4 if(!window.IndexedDB) {
5     console.log("Seu navegador não suporta o recurso IndexedDB.");
6 }

```

Fonte: Adaptado de Macoratti (2015).

Após verificado se o navegador utilizado pelo usuário aceita essa tecnologia, a base de dados indexada tem um padrão básico de funcionamento (MOZILLA DEVELOPER NETWORK, 2017b).

1. Criar o banco de dados;
2. Criar um *ObjectStore*;
3. Iniciar uma transação (requisição de operação no banco);
4. Esperar a operação ser completada;
5. Manipular o resultado da operação (requisição).

Utilizando como exemplo prático, uma nova base de dados indexada pode ser aberta invocando o método *idb.open()* que recebe como parâmetros: o nome da base a ser criada, o número da versão da nova base e um call-back de atualização opcional. Exemplo: *var idbTeste = idb.open('test-db1', 1)*.

As ferramentas para desenvolvedores do Google Chrome permitem inspecionar, modificar e excluir dados do *IndexedDB*. Bem como, gerenciar as diversas bases que poderão ser criadas

Os arquivos referentes à base de dados do *IndexedDB* podem ser encontrados no Windows, para o navegador Google Chrome, no diretório:

➤ *C:\Users\%usuário%\AppData\Local\Google\Chrome\User Data\Default\IndexedDB*

Para o Mozilla Firefox, os arquivos podem ser encontrados no diretório:

➤ *C:\Users\%usuário%\AppData\Roaming\Mozilla\Firefox\Profiles*.default\storage*

A versão candidata à recomendação para W3C data o dia 10 de agosto de 2017 com o nome de *Indexed Database API 2.0* (W3C, 2017).

Uma observação importante a se fazer sobre esse capítulo é que as tecnologias que foram descritas na data de elaboração desse trabalho, por se tratarem de tecnologias que estão começando a ser especificadas, implementadas e utilizadas nos navegadores, as maiores fontes de material bibliográfico sobre o assunto são o Google Developers, a Mozilla Developers Network, WHATWG, W3C e o blog pessoal de Jake Archibald. Portanto, tais tecnologias são as que foram utilizadas no desenvolvimento do protótipo desse trabalho.

4 PROTÓTIPO DESENVOLVIDO

Neste trabalho foi desenvolvido um protótipo de uma aplicação Web para **gerenciar o treinamento dos futuros usuários**. O **principal objetivo** desse protótipo é que ele **funcione on-line** e na eventualidade da falta de rede, funcione também **off-line**, ou seja, sem interrupções para o usuário.

O protótipo funciona on-line fazendo uso do modelo cliente/servidor clássico para operações de CRUD¹⁶ (na prática nenhum dado será deletado, apenas inativado). O funcionamento off-line ocorre armazenando em cache recursos endereçáveis por URL (recursos estáticos ou *shell* do aplicativo), através do *Cache Storage* e utiliza o *IndexedDB* para o controle de dados dinâmicos que necessitem ser persistidos localmente enquanto a rede estiver indisponível.

Para ajudar no entendimento sobre o funcionamento do controle de treinamento de exercícios físicos é necessário fazer uma breve explicação sobre periodização de treinamento (microciclos de treinamento), sessões de treino (parte prática do esporte) e exercícios físicos propriamente ditos, além de ressaltar a importância de seu controle para o objetivo prático desse trabalho.

4.1 Treinamento desportivo

O treinamento desportivo é composto de ciclos que são planejados para cada modalidade de esporte. Esses ciclos, também são denominados pela literatura de **periodização de treino** e auxiliam o praticante a desenvolver seus objetivos dentro do esporte.

Os microciclos constituem a micro estrutura dos sistemas de periodização, e são caracterizados pelo conjunto das sessões de treino que são o elo unificador de todo este processo. Em norma a sua duração é de uma semana, mas podem ser (e são em alguns casos), constituídos por três a quatro dias. São, no essencial, o elemento determinante da qualidade do processo de treino, assumindo funções diversificadas e fundamentais (SORIANO, 1996, p. 22).

Com base em Soriano, entende-se que os microciclos são a menor parte do sistema de treinamento com pequena duração de dias e eles são compostos pelas **sessões de treinamento**

¹⁶ CRUD é o acrônimo em inglês para *create, retrieve, update and delete*, operações realizadas em um gerenciador de banco de dados relacional. Em português traduz-se para: criar, recuperar, atualizar e deletar.

que são o programa **prático-diário** de treinamento, brevemente, uma sessão de treino é composta por três fases: o aquecimento, o treinamento e o esfriamento (BOMPA e CORNACCHIA, 2000, p. 45).

Os microciclos podem ser classificados quanto as tarefas a que se destinam no treinamento e quanto aos aspectos comuns dos treinos, por exemplo a grandeza das cargas, metodologia, etc. (GOMES, A., 2009, p. 179).

A Tabela 6 traz uma possível classificação para os microciclos.

Tabela 5: Classificação básica de microciclos de treinamento

MICROCICLO	CARACTERÍSTICAS
Ordinário	É o mais comumente encontrado no treinamento. Visa provocar as adaptações orgânicas desejáveis, capazes de incrementar o nível de condicionamento do praticante da atividade desportiva. Cargas cerca de 60 a 80% em relação as máximas.
de Choque	Caracteriza-se pela soma das cargas máximas ou próximas as máximas, de 80% a 100%. Exige mobilização máxima do organismo.
Estabilizador	É aplicado com o objetivo de assegurar a estabilidade do estado do organismo do do praticante da atividade desportiva. Cargas variando entre 40% a 60% em relação as máximas.
de Manutenção	Assegura não só a recuperação do praticante, mas reduz consideravelmente os ritmos de perda de preparo devido aqueles sistemas que ficaram num estado reprimido. Cargas utilizadas em relação a máxima entre 30-40%.
Recuperativo	Geralmente aplicada em dias de descanso, visa assegurar a recuperação mais completa e eficiente do praticante da atividade desportiva. As cargas são mínimas em relação as máximas, se estabelecendo entre 10-20%
Controle	Este microciclo visa preparar o do praticante da atividade desportiva para que entre no estado de pré-competição ou competição. O intuito é verificar o nível de preparação que do praticante com partidas de controle e a execução de testes. A carga varia muito e depende do teste.

Fonte: Adaptado de Antonio Carlos Gomes (2009, p. 180-193).

Os dados da Tabela 6 mostram os tipos básicos de microciclos no treinamento em geral, é claro que um pode ser mais adequado que o outro para treinar um praticante de determinada modalidade, porém a despeito disso é possível e recomendado que sejam combinados (periodizados) para que de forma conjunta expandam o potencial do esportista e ele atinja seus objetivos.

Os microciclos também ajudam a evitar que o praticante esteja em permanente estado de cansaço físico ou mental, devido ao volume grande treinos. Eles proporcionam uma alternância entre esforço e recuperação rápida.

Como fora anteriormente mencionado os microciclos são constituídos de sessões de treinamento. Essas, por sua vez, são compostas por exercícios físicos. Exercícios físicos são uma “sequência planejada de movimentos repetidos sistematicamente com o objetivo de elevar o rendimento” do praticante (BARBANTI, 2003, p. 249).

Assim, os exercícios físicos que serão praticados durante o treinamento têm papel fundamental no rendimento do praticante para a atividade esportiva escolhida. Santos e Simões (2012, p. 181) destacam que eles trazem “melhoria da percepção dos indivíduos no que se refere ao bem-estar físico, social e emocional”. Existem inúmeros exemplos de exercícios físicos, que variam de acordo com a modalidade esportiva adotada, tais como: abdominais, corrida, levantamento de peso, dentre outros.

Na prática, em uma possível ordenação: os exercícios físicos estão contidos nas sessões de treino que pertencem aos seus respectivos microciclos, caracterizando uma periodização de treinamento desportivo completa. A importância de controlar-se tais variáveis para o objetivo desse trabalho é a de que o usuário do protótipo consiga acompanhar o histórico de seus treinamentos ao passo que procura melhoria contínua para seu rendimento esportivo.

Após atestadas as vantagens da periodização de treinamento para o desempenho do esportista em face a execução dos exercícios físicos contidos em cada sessão de treino que for realizada, a documentação do protótipo do aplicativo que gerencia essas variáveis está descrita na seção 4.3.

4.2 Documentação do protótipo da aplicação

A documentação gerada nessa seção refere-se aos mecanismos de funcionamento on-line e off-line – que são o núcleo - do protótipo desse trabalho. A prototipagem é definida na seção 4.4.

Antes de iniciar qualquer trabalho técnico, é uma boa ideia aplicar um conjunto de tarefas da engenharia de requisitos. Estas levam a um entendimento de qual será o impacto do software [...] e como os usuários finais irão interagir com o software. [...] O objetivo da engenharia de requisitos é fornecer a todas as partes um entendimento escrito do problema. Isso pode ser alcançado por meio de uma série de artefatos: cenários de uso, listas de funções e características, modelos de análise ou alguma especificação (PRESSMAN, 2011, p. 126).

De acordo com a citação de Roger S. Pressman, a análise de requisitos¹⁷ fornece as ferramentas iniciais necessárias para a documentação da aplicação que será projetada e, mais que isso, levar ao entendimento sobre o funcionamento esperado pela aplicação.

A principal boa prática proveniente da engenharia de requisitos, é justamente o levantamento dos requisitos da aplicação, que segundo Somerville (2007, p. 80), são principalmente classificados em **requisitos funcionais** (seção 4.3.1) e **requisitos não funcionais** (seção 4.3.2).

4.2.1 *Requisitos funcionais*

Os requisitos funcionais dizem respeito às funcionalidades técnicas da aplicação (SOMERVILLE, 2007, p. 81).

- O protótipo deve manter um cadastro e controle de usuários e sessões de treino.
- As sessões de treino do protótipo devem ser persistidas off-line utilizando *Cache Storage* e *IndexedDB*.
- O protótipo deve permitir que os usuários autorizados manipulem dados (inserção, alteração, exibição e inativação).
- O protótipo deve ser consistente em seus dados.
- O protótipo deve funcionar primariamente on-line e depois off-line.

4.2.2 *Requisitos não funcionais*

Os requisitos não funcionais especificam ou restringem propriedades emergentes da aplicação, como por exemplo: confiabilidade, qualidade, segurança, usabilidade, etc.

¹⁷ A engenharia de requisitos é “uma das primeiras fases de um processo de desenvolvimento de software ...” (GUEDES, 2011, p. 21). A engenharia de requisitos pertence à Engenharia de Software.

- **Confiabilidade:** o protótipo tentará se recuperar de falhas sistêmicas e caso não for possível mostrará mensagens claras sobre os erros originados.
- **Portabilidade:** o protótipo será portátil por se tratar de uma *WebApp* e funcionar em navegadores Web.
- **Segurança:** o protótipo não exibirá dados informações sensíveis do usuário e caso seja necessário o envio dessa informação entre as camadas da aplicação, elas serão previamente criptografadas ou tratadas.
- **Usabilidade:** o protótipo irá interoperar entre os estados on-line/off-line sem prejuízo para o usuário, notificando-o dessa mudança quando necessário.
- **Interface:** o protótipo, por se tratar de uma PWA, poderá ser utilizado tanto em dispositivos mobile quanto desktop.
- **Documentação:** o protótipo terá sinalização visual e escrita para os status de suas sessões de treinamento.
- **Instalação:** o protótipo, por se tratar de uma *WebApp*, não é instalável, mas o usuário poderá adicionar um atalho a área de trabalho através do *Application Manifest* para facilitar seu acesso.

O levantamento dos requisitos funcionais e não funcionais ajudam a identificar o que o protótipo deve ou não conter. A próxima etapa é a modelagem do protótipo, que abstrai os requisitos que são relevantes e os documenta na forma de diagramas.

A ferramenta mais conhecida para documentação e diagramação de aplicações é a **UML (*Unified Modeling Language*)**. Cada diagrama da UML mostra a aplicação em um nível diferente de abstração e completa o outro, juntos eles fornecem uma visualização global, uma projeção do aplicativo a ser desenvolvido (GUEDES, 2011, p. 30).

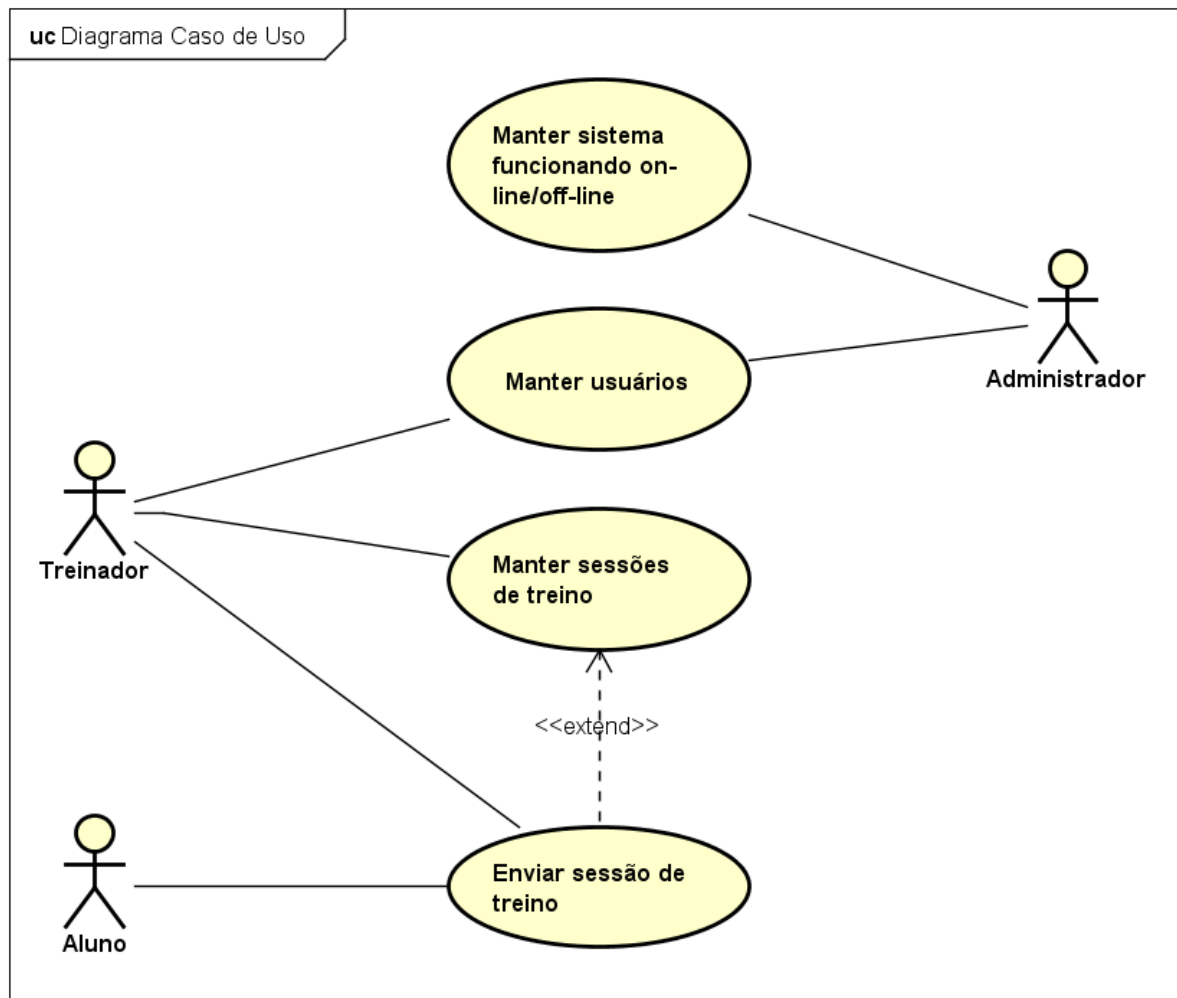
Os diagramas da UML concernentes ao núcleo do protótipo que foram modelados são: **Caso de Uso, Classe, Atividade, Sequência e Diagrama de Entidade-Relacionamento.**

4.2.3 Diagrama de Caso de Uso

O diagrama de Caso de Uso é o mais geral da UML. Apresenta uma ideia de fácil compreensão sobre o comportamento da aplicação e identifica os atores contidos da aplicação (GUEDES, 2011, p. 30).

A Figura 19 representa o diagrama Caso de Uso desse protótipo.

Figura 19: Diagrama Caso de Uso do protótipo



Fonte: Elaborado pelo autor utilizando o software Astah Community.

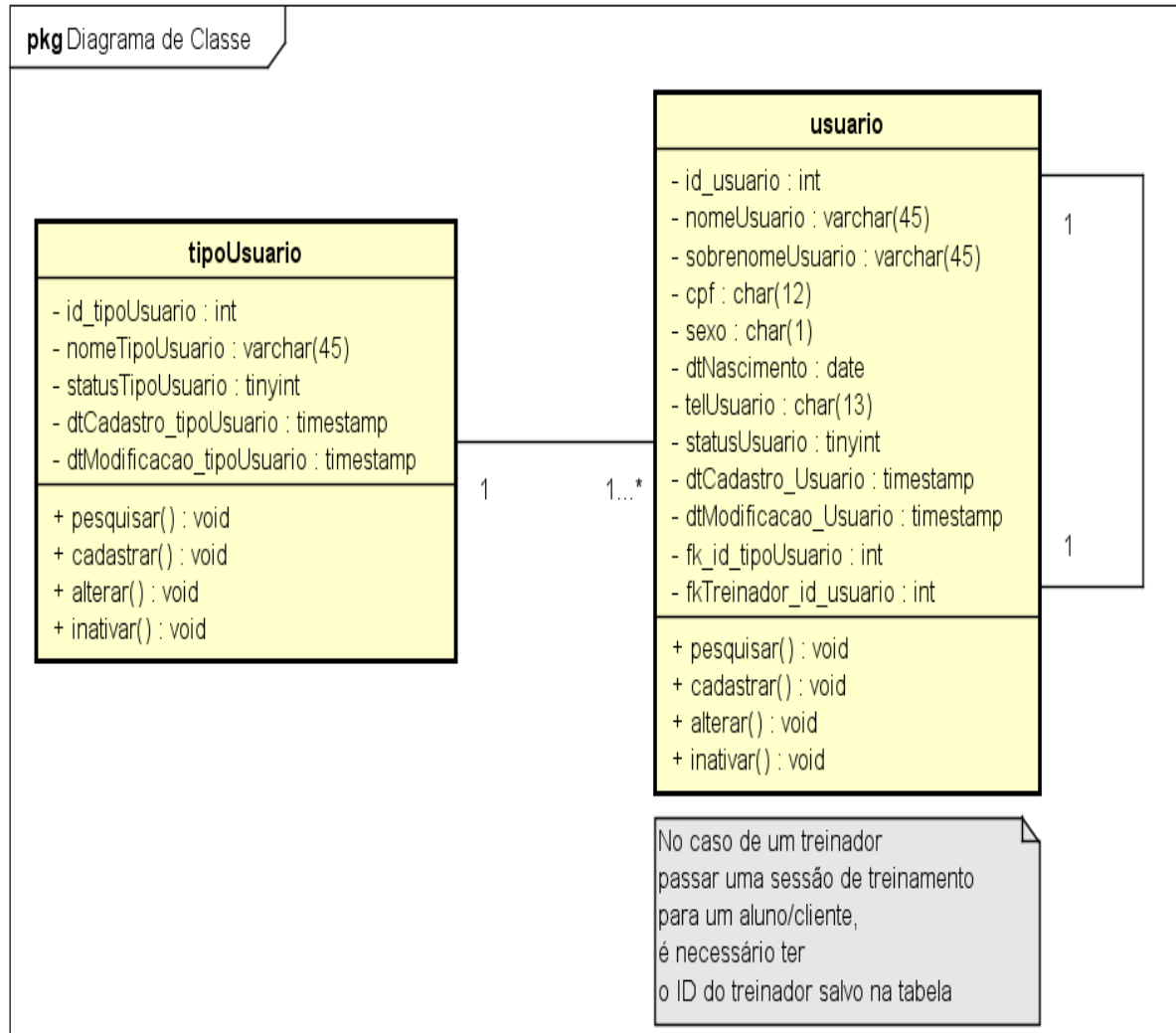
Como pode ser visto no que foi modelado para o protótipo com o diagrama de Caso de Uso, o administrador da aplicação é responsável pela manutenção funcionamento on-line/off-line, bem como gestão de usuários. O treinador é responsável pela gestão das sessões de treinamento e o aluno realiza a sessão de treino e envia o status atualizado.

4.2.4 Diagrama de Classe

O diagrama de Classe mostra, além de propriedades e métodos, como as classes existentes na aplicação se relacionam no que diz respeito à troca de informações (GUEDES, 2011, p. 101).

A Figura 20 representa o primeiro diagrama de Classe desse protótipo.

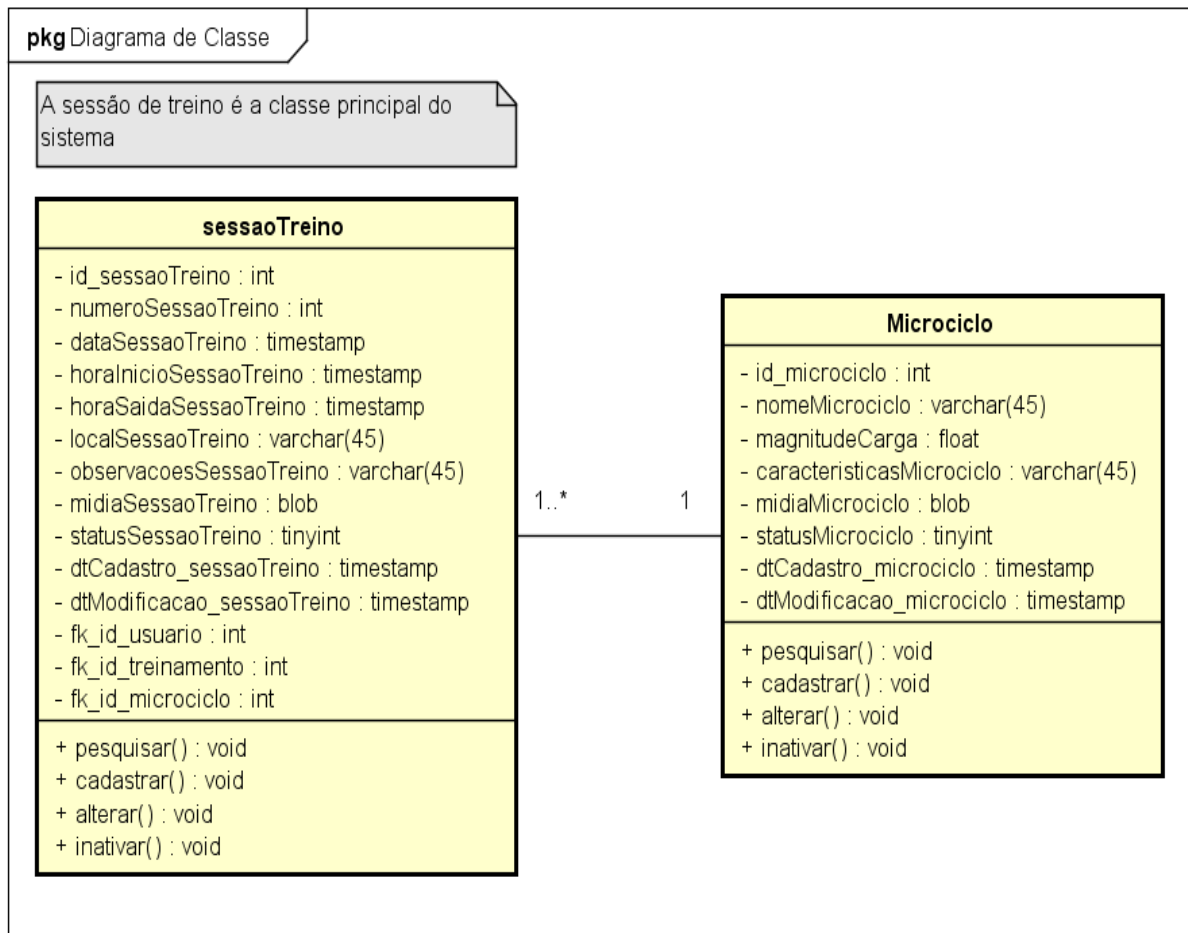
Figura 20: Diagrama de Classe do protótipo – Tipo de Usuário e Usuário



Fonte: Elaborado pelo autor utilizando o software Astah Community.

Como foi modelado na Figura 20, cada classe de **usuário** pertence a um **tipo de usuário** (Administrador, Treinador ou Cliente). A classe usuário recebe um identificador dela mesma, para indicar qual o treinador, que também é um usuário, que passou a sessão de treino para o aluno. Existem mais classes que se relacionam principalmente com a classe usuário, mas que não aparecem nessa documentação, pois não concernem ao núcleo do sistema. Cada classe possui seus atributos e métodos respectivos. A Figura 21 representa o segundo diagrama de Classe desse protótipo.

Figura 21: Diagrama de Classe do protótipo – Sessão de Treino e Microciclo



Fonte: Elaborado pelo autor utilizando o software Astah Community.

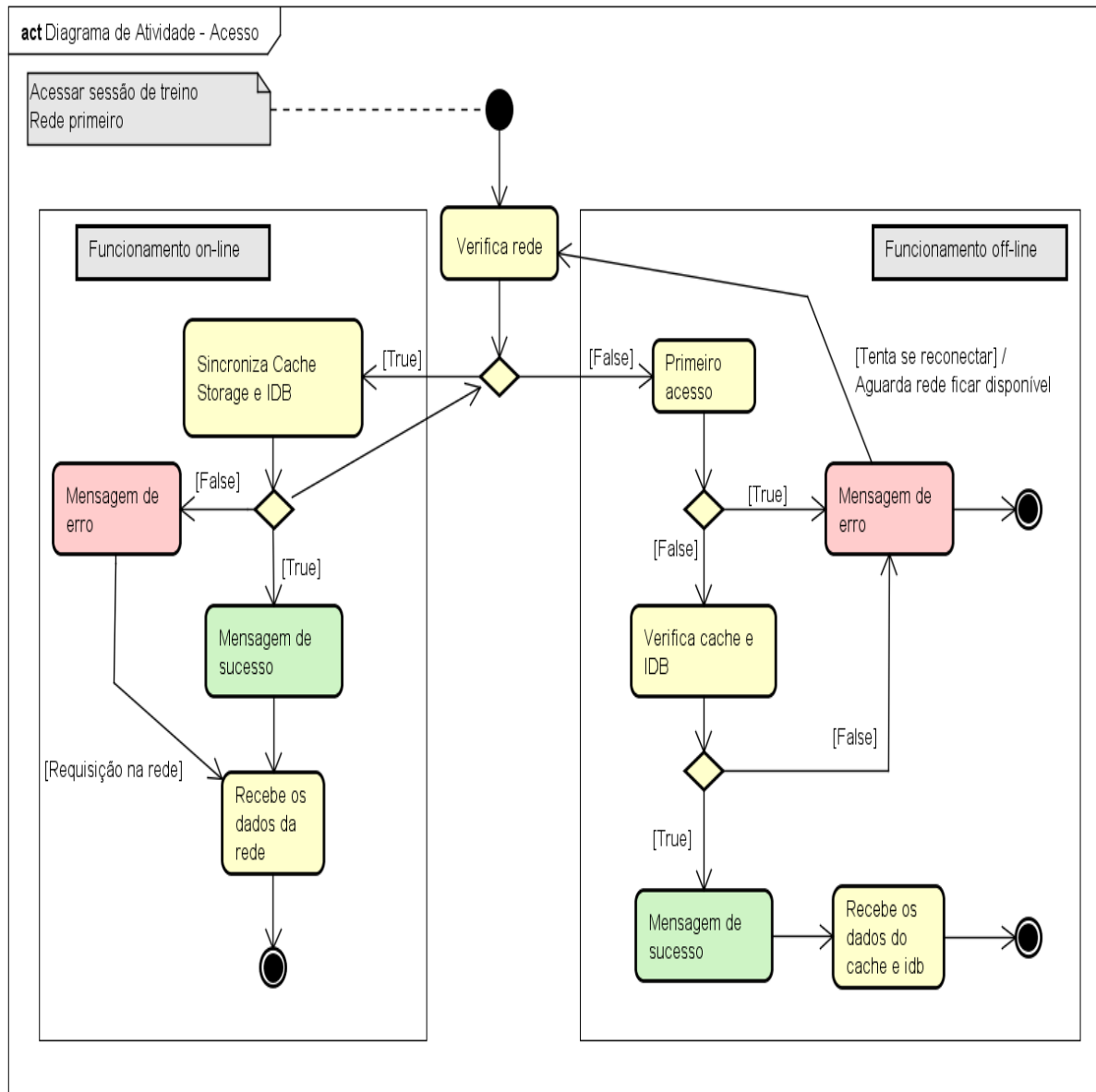
A segunda modelagem, diz respeito às classes **sessão de treino** e **microciclo**. A classe sessão de treino, como é reforçado pela Figura 21, é a principal classe do protótipo, pois ela contém todos os relacionamentos necessários (chaves estrangeiras dos alunos, do treinamento e do microciclo) para o funcionamento da parte prática do protótipo. O diagrama apresenta as classes com seus respectivos atributos e métodos.

4.2.5 Diagrama de Atividade

O diagrama de Atividade, conforme explica Guedes (2011, p. 36) “concentra-se na representação do fluxo de controle da uma atividade específica”. Isso significa que esse diagrama representa um algoritmo completo de uma atividade que a aplicação desempenha.

A Figura 22 representa o primeiro diagrama de Atividade desse protótipo.

Figura 22: Diagrama de Atividade do protótipo – Acessar sessão de treino

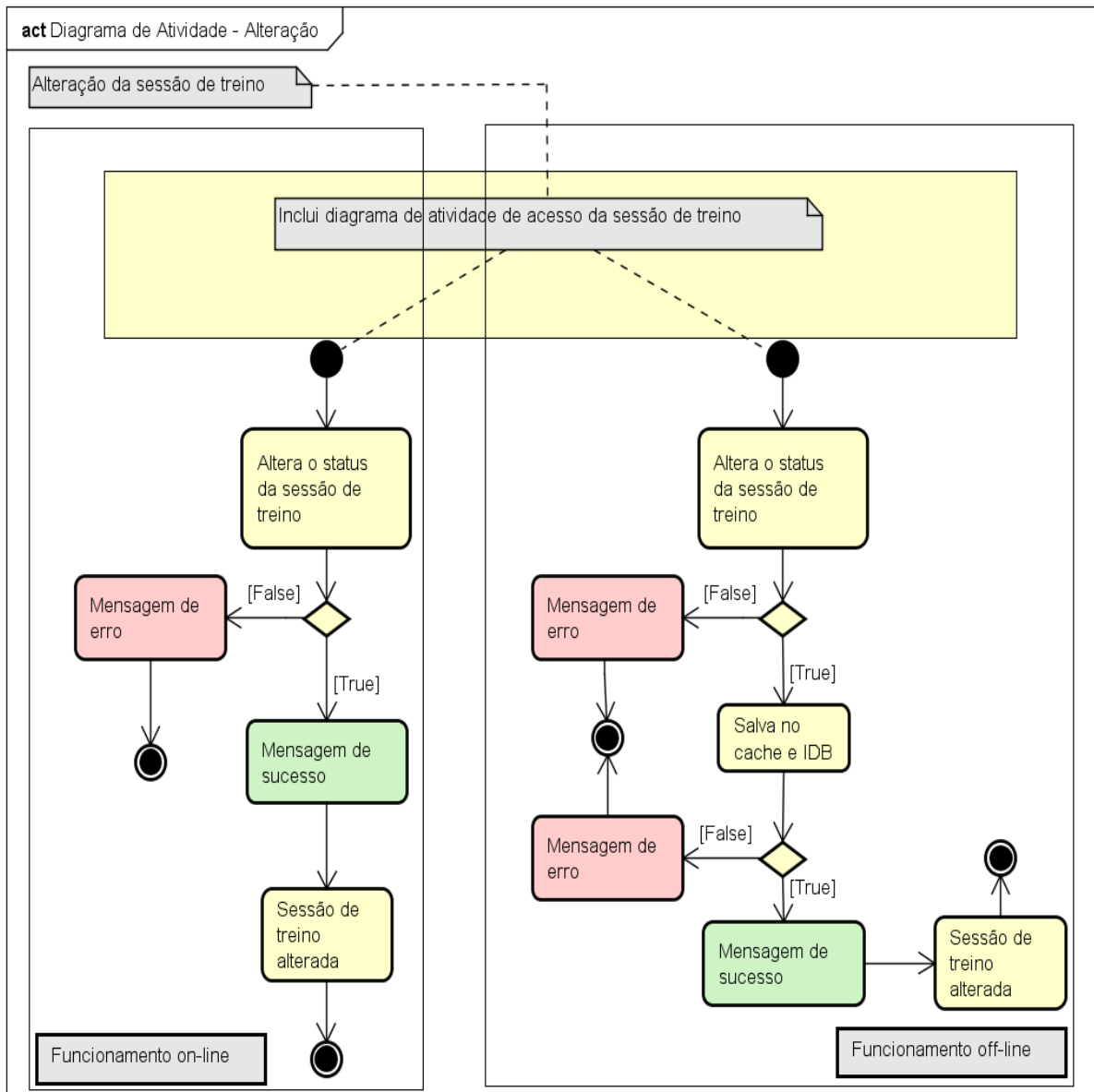


Fonte: Elaborado pelo autor utilizando o software Astah Community.

Nessa modelagem pode ser visto o diagrama de atividade para o **acesso da sessão de treino**. O protótipo sempre tentará inicialmente, acessar a rede. Caso consiga, ocorre o funcionamento on-line tradicional e a página da sessão de treino é exibida, com um adendo da sincronização dos ativos que são armazenados em cache e no IDB para uso posterior off-line. Caso a rede estiver indisponível, ocorre uma verificação para ver se existem dados que foram previamente armazenados localmente no navegador e então a página da sessão de treino é renderizada do armazenamento off-line e exibida.

A Figura 23 representa o segundo diagrama de Atividade desse protótipo.

Figura 23: Diagrama de Atividade do protótipo – Alterar sessão de treino



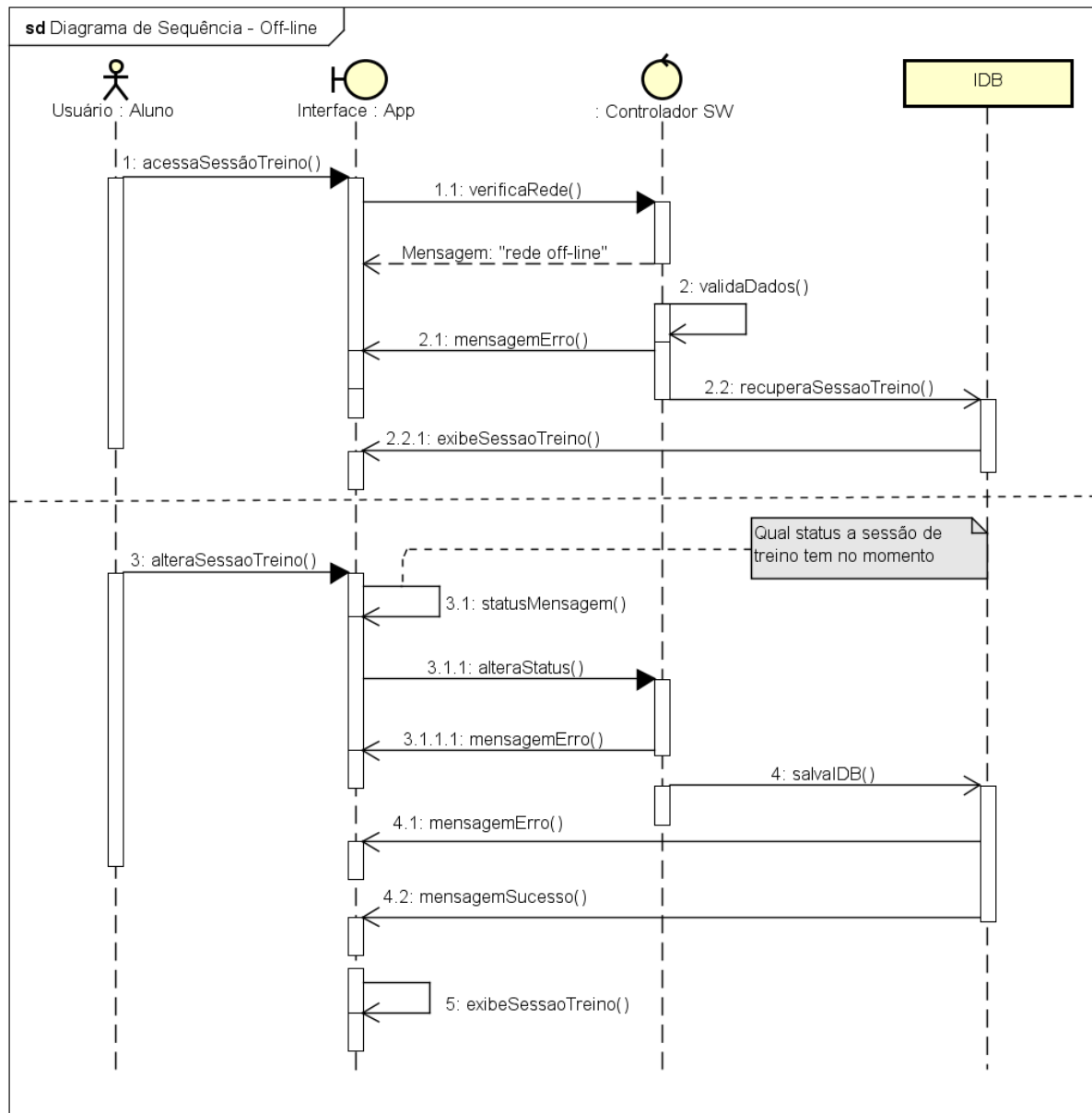
Fonte: Elaborado pelo autor utilizando o software Astah Community.

A modelagem no segundo diagrama mostrado na Figura 23, continua imediatamente após o acesso a sessão de treino, nesse momento o usuário decide se irá **alterá-la**, na prática isso resume-se ao treinador criar uma sessão de treino, ou o treinador passar a sessão de treino para o aluno ou então, o aluno decidir realizar a sessão de treino passada pelo treinador ou não. O funcionamento do protótipo segue dividido em on-line e off-line, sendo que na parte com rede, o fluxo de eventos é quase constante, se não houver erro. O funcionamento off-line depende do sucesso na armazenagem da alteração do status da sessão de treino no cache e no IDB, se tiver algum erro esse é sinalizado, senão os dados são salvos localmente, ficando disponíveis para consulta e posterior sincronização com a base de dados MySQL.

4.2.6 Diagrama de Sequência

O diagrama de Sequência mostra os eventos, as etapas e a ordem de que um determinado processo tem na aplicação (GUEDES, 2011, p. 277). A Figura 24 representa o primeiro diagrama de Sequência desse protótipo.

Figura 24: Diagrama de Sequência do protótipo – funcionamento off-line

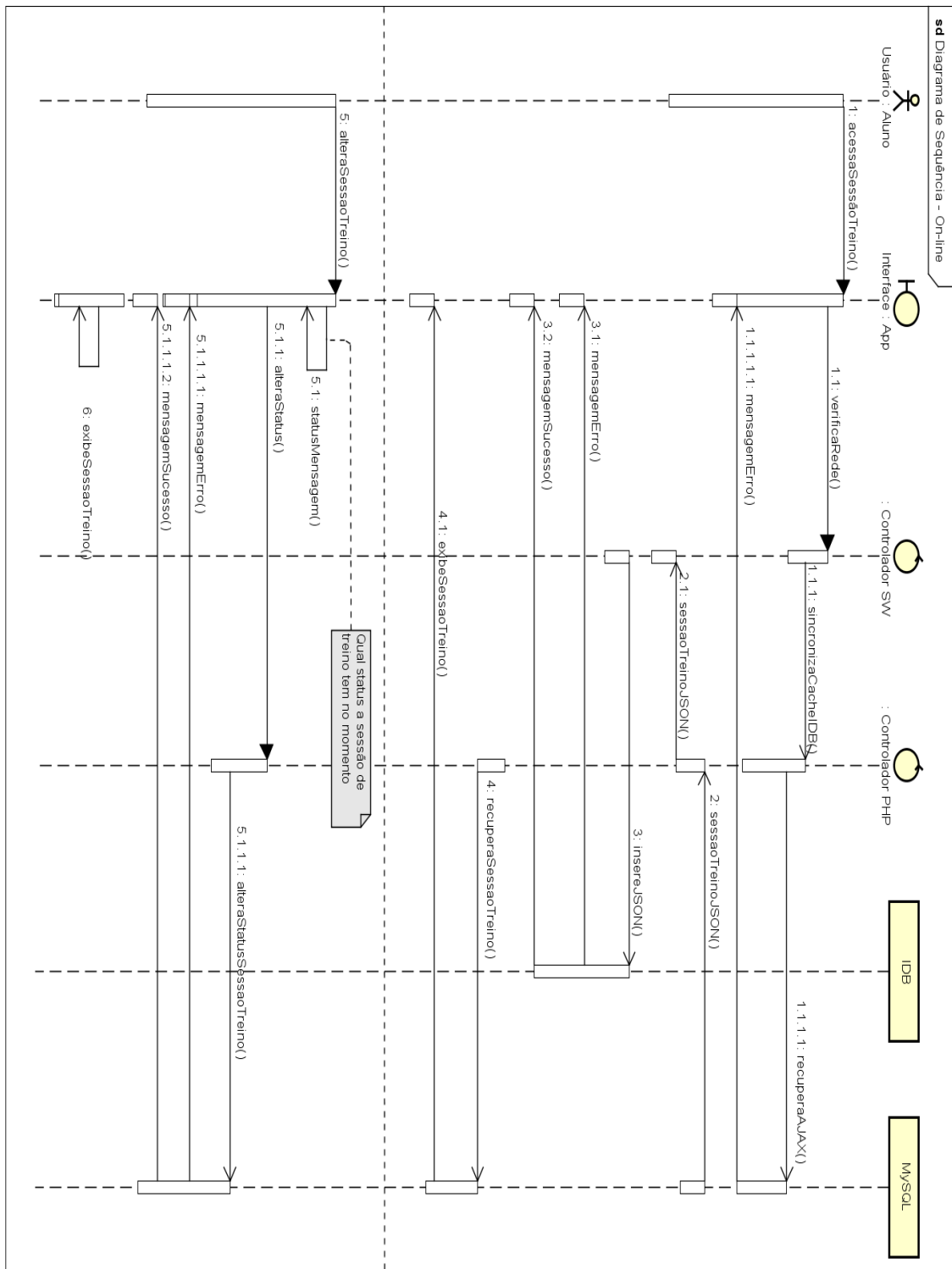


Fonte: Elaborado pelo autor utilizando o software Astah Community.

O diagrama de sequência apresenta uma modelagem mais próxima dos eventos que são disparados em cada etapa para **acessar** ou **alterar a sessão de treino**. A primeira parte

mostra o funcionamento off-line e a Figura 25 mostra o diagrama de seqüência referente ao funcionamento on-line desse protótipo.

Figura 25: Diagrama de Sequência do protótipo – funcionamento on-line



Fonte: Elaborado pelo autor utilizando o software Astah Community.

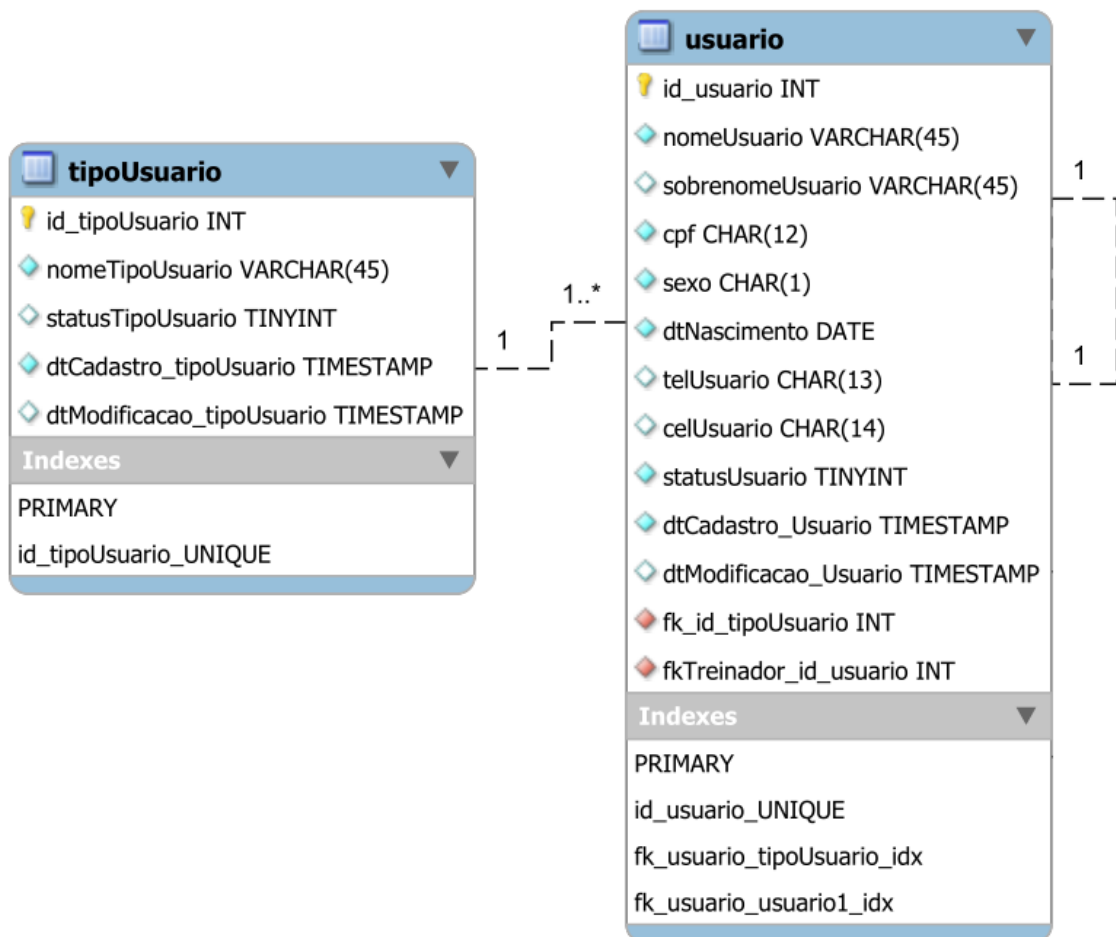
A modelagem apresentada na parte on-line (Figura 25) parece ser um pouco mais complexa, quando na verdade constitui a diferença de buscar pelos recursos que são enviados através da rede pelo *back-end* (API) para o *front-end* (App), enquanto o funcionamento off-line recupera os dados armazenados **localmente** no *browser* e os envia para o *front-end*.

4.2.7 Diagrama de Entidade-Relacionamento

O diagrama de Entidade-Relacionamento mostra por meio de notações gráficas a modelagem para uma aplicação de banco de dados (ELMASRI e NAVATHE, 2011, p. 65).

A Figura 26 representa o diagrama de Entidade-Relacionamento para as entidades **Tipo Usuário** e **Usuário**.

Figura 26: Diagrama de Entidade-Relacionamento Tipo Usuário - Usuário

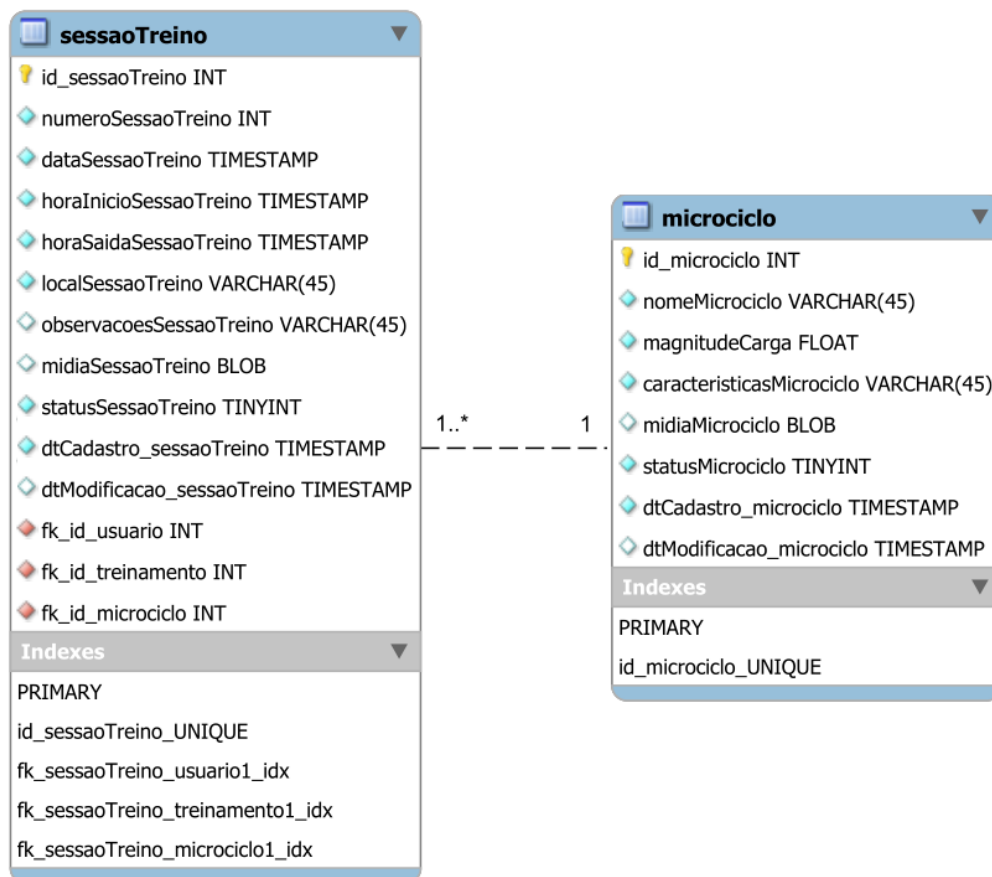


Fonte: Elaborado pelo autor utilizando o software MySQL Workbench.

Como pôde ser visto, a modelagem (apesar de bem parecida com o diagrama de classes) no diagrama de Entidade-Relacionamento o foco é diretamente na projeção do banco de dados.

A Figura 27 representa o diagrama de Entidade-Relacionamento para as entidades Sessão de Treino e Microciclo.

Figura 27: Diagrama de Entidade-Relacionamento Sessão de Treino e Microciclo



Fonte: Elaborado pelo autor utilizando o software MySQL Workbench.

O diagrama da Figura 27 conclui a modelagem do protótipo apresentando a modelagem para o banco de dados da entidade **microciclo** e da entidade principal do protótipo que é a **sessão de treino**. O que pode ser notado é que um microciclo pode possuir um ou mais sessões de treino, o que valida a afirmação da explicação exposta sobre a periodização de treino na seção 4.2.

A seção 4.4 contém o que foi utilizado na prototipação da aplicação, bem como uma noção visual e prática do protótipo desenvolvido.

4.3 Prototipagem da aplicação

Após a documentação do núcleo do sistema, foi construído um protótipo da aplicação que contempla o objetivo desse trabalho. Um protótipo é uma primeira amostra sobre a aplicação desenvolvida, segundo afirma Guedes (2011, p. 24):

[...] a etapa de análise de requisitos deve, obrigatoriamente, produzir um protótipo para demonstrar como se apresentará e comportará o sistema em essência, bem como quais informações deverão ser inseridas [...] e que tipo de informações deverão ser fornecidas pelo software.

Para demonstrar o comportamento projetado para a aplicação, o desenvolvimento do protótipo desse trabalho foi realizado em duas vertentes. Uma API construída usando na linguagem de script PHP, que ficou responsável pelo desenvolvimento tradicional do *back-end* do protótipo e um APP, escrito majoritariamente em *JavaScript* que ficou responsável pelo *front-end*, isto é, por consumir os recursos fornecidos pela API, também por manipular as tecnologias de armazenamento off-line, além de construir as interfaces do protótipo, contando com um *template* inovador que foi implementado. A Figura 28 mostra a tela principal do protótipo visualizada em um navegador mobile.

Figura 28: Tela Sessões de Treinamento do protótipo

SIGMIT | Aluno

Sessões de Treinamento

Listar Treinamentos

10 resultados por página

Pesquisar

Treinador	Sessão	Data	Local	Status
⊖ Treinador	2	2017-11-23	Outdoor	A fazer
⊕ Treinador	1	2017-11-23	Academia FATEC	Realizado
⊕ Treinador	1	2017-11-22	Academia FATEC	Realizado
⊕ Treinador	1	2017-11-23	outdoor - teste	A fazer
⊕ Treinador	4	2017-11-26	Parque	A fazer

Mostrando de 1 até 5 de 5 registros

< 1 >

Home Sessão de Treino Contato

Fonte: Desenvolvido pelo autor.

Como pôde ser visto na Figura 28, as principais informações acerca das sessões de treinamento estão nessa tela, na forma de tabela, disponíveis para a consulta do usuário. As demais telas desenvolvidas para o protótipo podem ser consultadas no **Apêndice**, no final desse trabalho.

Nesse momento, inicia-se a codificação para permitir o armazenamento de recursos em cache, logo após o *Service Worker* ser registrado na página. A Figura 29 traz essa codificação.

Figura 29: Código do *Service Worker* contido no arquivo “*service-worker.js*”

```

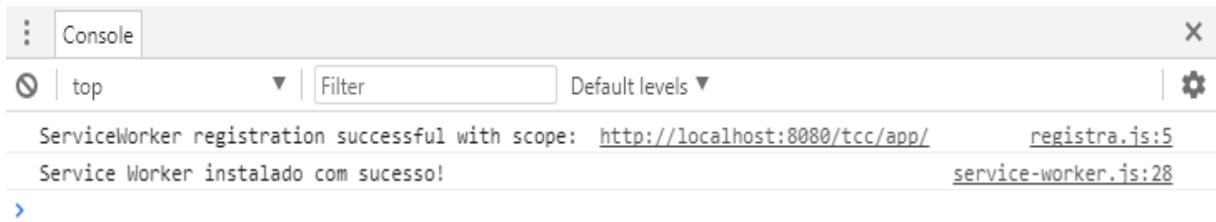
1  let versao = 1
2  var tccCache = 'tcc-cache-versão-' + versao;
3  let arquivos = [
4    '/',
5    'tcc_sw.js',
6
7    'assets/css/bootstrap.min.css',
8    'assets/css/style.css',
9    'assets/css/navbar.css',
10   'assets/css/footer.css',
11   'assets/css/dataTables.bootstrap.min.css',
12   'assets/css/datatables.min.css',
13
14   'assets/js/jquery-3.2.1.min.js',
15   'assets/js/bootstrap.min.js',
16   'assets/js/datatables.all.min.js',
17   'assets/js/bootbox.min.js',
18
19   'L3/menu.html',
20   'L3/navbar.html',
21   'L3/footer.html',
22
23   'L3/app.js',
24   'L3/sessaoTreino/read-sessaoTreino.js',
25   'L3/sessaoTreino/read-one-sessaoTreino.js',
26   'L3/sessaoTreino/read-sessaoTreino-Template.js',
27   'L3/sessaoTreino/update-sessaoTreino.js',
28
29   '../api/sessaoTreino/read.php'
30 ]
31
32 //No evento de instalação
33 self.addEventListener("install", function(event) {
34   console.log("Service Worker instalado com sucesso!");
35   self.skipWaiting();
36
37   event.waitUntil(
38     caches.open(tccCache).then(function(cache) {
39       return cache.addAll(arquivos);
40     })
41   )
42 })

```

Fonte: Desenvolvido pelo autor.

Nesse código mostrado pela Figura 29, o *Service Worker* tentará se registrar na página pelo evento “*install*” e se tudo ocorrer bem, uma mensagem de sucesso no console do navegador usuário será exibida. A Figura 30 mostra a mensagem de instalação bem-sucedida do *Service Worker* no console do navegador do usuário.

Figura 30: Mensagem de instalação do *Service Worker* no console do navegador

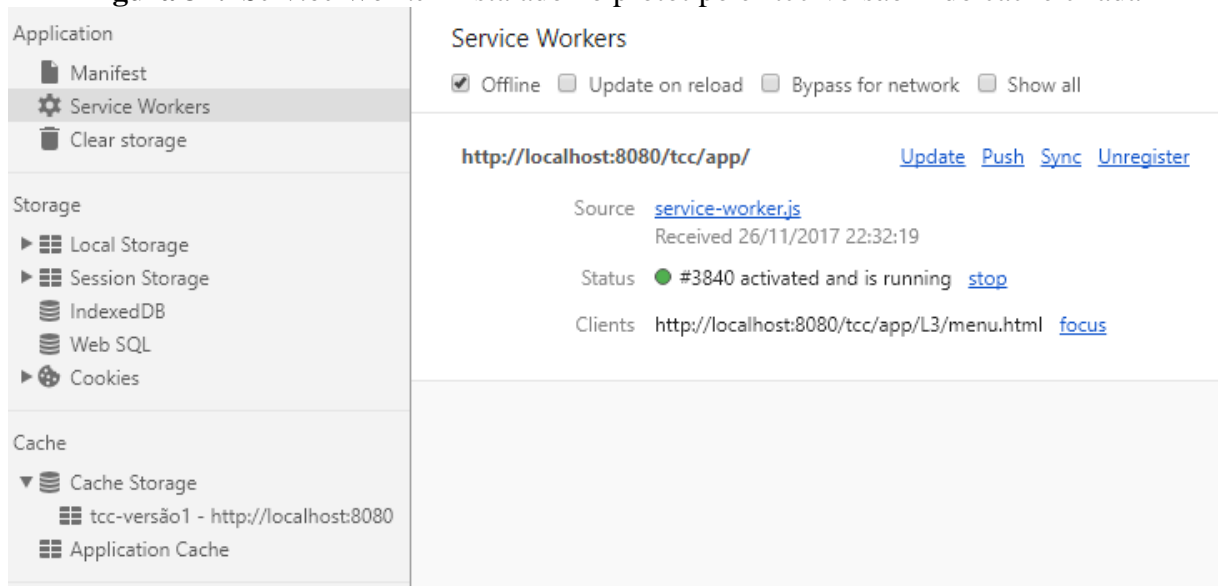


Fonte: Tela do navegador do autor.

Nesse momento o *Service Worker* já está controlando as páginas dentro do escopo em que foi instalado, (escopo do diretório do APP, definido pelo arquivo “*registra.js*”).

Em seguida, é criada a versão do cache (**tcc-versão1**), conforme Figura 31.

Figura 31: *Service Worker* instalado no protótipo e "tcc-versão1" do cache criada

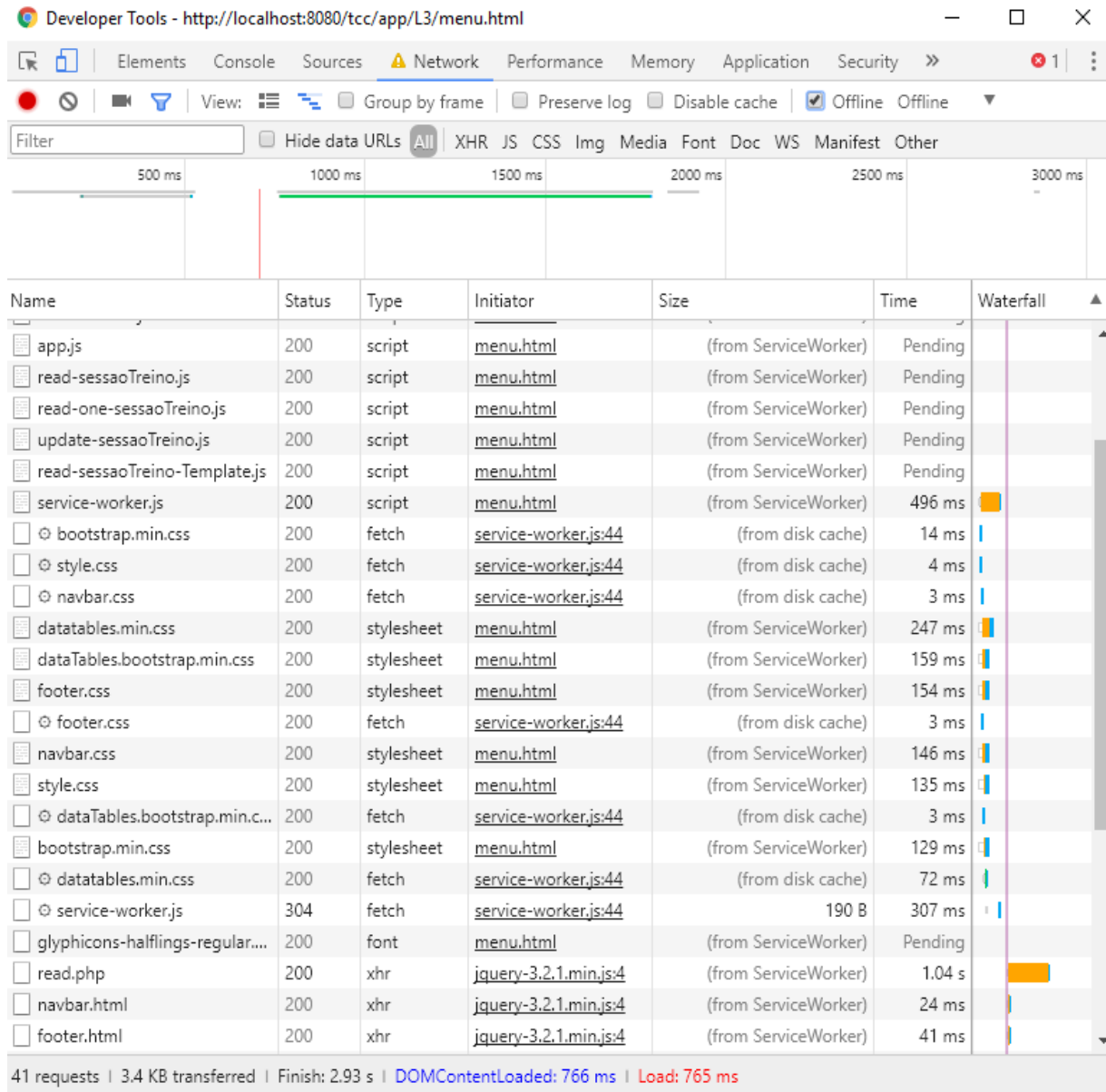


Fonte: Tela *Application* das ferramentas de desenvolvedor presentes no navegador Google Chrome.

A Figura 31 traz algumas informações adicionais sobre localização do escopo do *Service Worker* instalado (**http://localhost:8080/tcc/app/**), assim como número do SW (**3840**), seu status (**ativo e rodando**) e a página que está controlando no momento (**http://localhost:8080/tcc/app/L3/menu.html**).

Com o SW operacionalmente configurado, os ativos armazenados em cache podem ser recuperados e carregados quando a rede estiver off-line. Esse cenário é demonstrado pelas ferramentas para desenvolvedores do navegador Google Chrome (vide Figura 32).

Figura 32: Recursos armazenados pelo *Service Worker* sendo carregados do cache



Fonte: Tela *Network* das ferramentas de desenvolvedor presentes no navegador Google Chrome.

A Figura 32 mostra o usuário **sem conexão** à rede, tendo os recursos sendo carregados diretamente do cache previamente armazenado pelo SW. A ferramenta também mostra o nome dos recursos carregados, a resposta do status do pedido (200 para encontrado), o tipo do arquivo ou solicitação de origem, de onde foi carregado (disco ou SW) e o tempo de resposta.

Os ativos que não puderem ser encontrados no cache, através do evento “*fetch*” do SW, serão buscados na rede. A premissa do objetivo desse trabalho sendo atendida. O código para atingir tal propósito encontra-se exibido pela Figura 33.

Figura 33: Código do evento *fetch* do *Service Worker*

```

55 self.addEventListener("fetch", function(event){
56
57     let pedido = event.request
58
59     let promiseResposta = caches.match(pedido).then(respostaCache => {
60         let resposta = respostaCache ? respostaCache : fetch (pedido)
61         return resposta
62     })
63
64     event.respondWith(promiseResposta)
65 })

```

Fonte: Desenvolvido pelo autor.

O código da Figura 33 exemplifica como o SW funciona. Na prática é adicionado um escutador de eventos na página do site, para que durante o evento “*fetch*” do SW uma função que realize um “pedido” (requisição) seja executada e responda com uma *Promise*. Se o pedido encontrar dados salvos em cache (*caches.match()*), então é carregado e exibido do cache, senão um pedido é feito para a rede. Isso comprova a flexibilidade durante a transição dos estados de conectado e sem rede, comprovando que independente do estado que a rede estiver durante o pedido, o usuário continuará acessando a aplicação sem interrupções.

Por fim, foi desenvolvido o banco de dados indexado do protótipo, que ficou encarregado de armazenar e gerenciar o conteúdo e recursos dinâmicos. A Figura 34 mostra como o IDB é apresentado nas ferramentas para desenvolvedores do navegador Mozilla Firefox.

Figura 34: “*tcc_db*” criado para o protótipo

Database Name	Storage	Origin	Version
tcc_db	default	http://localhos...	1

Fonte: Tela *Storage* das ferramentas de desenvolvedor presentes no navegador Mozilla Firefox.

O navegador Mozilla Firefox consegue mostrar para o armazenamento informações sobre nome do banco de dados indexado (*tcc_db*), tipo de armazenamento (*default*), local (*http://localhost:8080*) de origem, além da versão atual da base (1). O código escrito para implementar o IDB pode ser visto na Figura 35.

Figura 35: Codificação para o *IndexedDB* do protótipo

```

1 (function() {
2   'use strict';
3
4   //check for support
5   if (!('indexedDB' in window)) {
6     console.log('This browser doesn\'t support IndexedDB');
7     return;
8   }
9
10  var dbPromise = idb.open('tcc_db', 1);
11
12  idb.open('tcc_db', 1, function(upgradeDB) {
13    var store = upgradeDB.createObjectStore('sessaoTreino', {
14      keyPath: 'id_sessaoTreino'
15    });
16    store.put({id_sessaoTreino: 2, Treinador: 'Treinador', numeroSessaoTreino: 2,
17      dataSessaoTreino: '2017-11-23', localSessaoTreino: 'Outdoor', statusSessaoTreino: 'A'});
18    store.put({id_sessaoTreino: 1, Treinador: 'Treinador', numeroSessaoTreino: 1,
19      dataSessaoTreino: '2017-11-23', localSessaoTreino: 'Academia FATEC", statusSessaoTreino: 'R'});
20    store.put({id_sessaoTreino: 3, Treinador: 'Treinador', numeroSessaoTreino: 1,
21      dataSessaoTreino: '2017-11-22', localSessaoTreino: 'Academia FATEC", statusSessaoTreino: 'R'});
22    store.put({id_sessaoTreino: 5, Treinador: 'Treinador', numeroSessaoTreino: 1,
23      dataSessaoTreino: '2017-11-23', localSessaoTreino: 'outdoor - teste', statusSessaoTreino: 'A'});
24    store.put({id_sessaoTreino: 6, Treinador: 'Treinador', numeroSessaoTreino: 4,
25      dataSessaoTreino: '2017-11-26', localSessaoTreino: 'Parque', statusSessaoTreino: 'A'});
26  });
27
28 })();

```

Fonte: Desenvolvido pelo autor.

O código mostrado pela Figura 35 realiza a verificação se o navegador utilizado suporta a tecnologia do *IndexedDB* e, em seguida, abre o banco passando seu nome e número de versão. Após isso, cria uma “loja de objetos” onde ficam armazenados os recursos dinâmicos que serão consumidos pelo protótipo durante o estado off-line.

Os navegadores que atualmente possuem suporte ao IDB são: para o desktop: Google Chrome versão 24 em diante, Mozilla Firefox 16.0 em diante, Internet Explorer 10, Opera 17 e para o mobile: Android versão 4.4, Firefox Mobile 16.0, IE Phone 10 e Opera Mobile 17 (MOZILLA DEVELOPER NETWORK, 2017b).

5 CONSIDERAÇÕES FINAIS

O presente trabalho trouxe como proposta o desenvolvimento de um protótipo de uma aplicação Web que controle as sessões de treino e os exercícios físicos de um treinamento esportivo e que, principalmente funcione on-line enquanto possuir acesso à rede e em uma eventual indisponibilidade funcione também off-line. Antes do desenvolvimento do protótipo propriamente dito, foi necessária uma breve diferenciação acerca das diferentes categorias de aplicações mais comuns, de modo a encontrar o cenário com a maior abrangência para implementação do protótipo. Seguido do estudo sobre quais tecnologias permitem o armazenamento de recursos off-line e quais são recomendadas para construção de *Progressive Web Apps*.

Seguindo esse raciocínio, foi averiguado que aplicações Web tendem a ter mais vantagens de alcance, acesso e portabilidade para os usuários do que aplicações nativas. A aplicação Web, em princípio funciona em qualquer dispositivo que tenha um navegador Web instalado. Na sequência, foi constatado que **o conjunto de APIs** *Service Workers*, *Cache Storage* e *IndexedDB* permitem a manipulação, a persistência e o armazenamento local de dados no navegador do usuário.

O desenvolvimento do protótipo permitiu comprovar a **aplicabilidade prática** dessas tecnologias (**funcionamento parcial off-line – para a sessão de treino**) e sugerir uma **inovação** para o funcionamento tradicional da Web, ou seja, a classe principal da aplicação – sessão de treino – pôde ser manipulada, através do **evento *fetch()*** do *Service Work*, para interoperar entre os estados de conectividade da rede e armazenar dados no *Cache Storage* durante o primeiro acesso. Utilizando o *IndexedDB* foi possível **recuperar e atualizar** o status da sessão de treino quando o usuário estava momentaneamente off-line, através do uso de transações na base de dados local do navegador (IDB).

Diversas tecnologias auxiliares incrementaram o conhecimento teórico-prático do autor durante a elaboração desse trabalho. Dentre as que foram utilizadas para apoiar o desenvolvimento do protótipo e que não foram mencionadas estão: a biblioteca Bootstrap versão 3.7.7, DataTables versão 1.10.16 para visual do protótipo, EasyPHP versão Devserver 17.0 para gerenciamento da API implementada, MySQL Workbench 6.3.9 para o DER e criação do banco de dados e o editor de código Atom em sua versão 1.22.1.

Como propostas futuras para esse trabalho intenta-se a implementação das tecnologias estudadas para permitir que a aplicação funcione **totalmente** off-line e implementar criptografia para assegurar e preservar os dados sensíveis dos usuários que ficam armazenados localmente no cache criado e no IDB.

REFERÊNCIAS

ARCHIBALD, Jake. **Web fundamentals**: funções assíncronas - simplificando promessas. on-line, 2017. Disponível em: <<https://developers.google.com/web/fundamentals/primers/async-functions>>. Acesso em: 11 nov. 2017.

_____. **Web fundamentals**: the offline cookbook. on-line, 2017b. Disponível em: <<https://developers.google.com/web/fundamentals/instant-and-offline/offline-cookbook/>>. Acesso em: 18 nov. 2017.

BARBANTI, Valdir J. **Dicionário de Educação Física e do Esporte**. São Paulo: Manole Ltda., 2003. p. 249.

BASQUES, Kayce. **Tools for Web Developers**: inspecionar e gerenciar armazenamento, bancos de dados e caches. on-line, 2017. Disponível em: <<https://developers.google.com/web/tools/chrome-devtools/manage-data/local-storage#local-storage>>. Acesso em: 18 nov. 2017.

BOMPA, Tudor O.; CORNACCHIA, Lorenzo J. **Treinamento de força consciente**. São Paulo: Phorte, 2000. p. 45.

COHEN, Marc. **Web fundamentals**: Visão geral do armazenamento Web. on-line, 2017. Disponível em: <<https://developers.google.com/web/fundamentals/instant-and-offline/web-storage/>>. Acesso em: 08 nov. 2017.

DEMOED, Alex Russel. **Progressive Web Apps**: Escaping Tabs Without Losing Our Soul – Infrequently Noted. on-line. Disponível em: <<https://infrequently.org/2015/06/progressive-apps-escaping-tabs-without-losing-our-soul/>>. Acesso em: 05 nov. 2017.

DEMOED, Alex Russel. *et al.* **Service Workers 1**. on-line. 2017. Disponível em: <<https://www.w3.org/TR/service-workers-1/>>. Acesso em: 12 nov. 2017.

DEVERIA, Alexis. **Can I use... Support tables for HTML5, CSS3, etc**: web storage. on-line. Disponível em: <<https://caniuse.com/#search=web%20storage>>. Acesso em: 02 nov. 2017.

DRIFTY CO. **Ionic Resources | Service Worker**. on-line. Disponível em: <<https://developers.google.com/web/fundamentals/primers/service-workers/https://ionicframework.com/docs/developer-resources/service-worker/>>. Acesso em: 14 nov. 2017.

ELMASRI, Ramez; NAVATHE, Shamkant. B. **Sistemas de banco de dados**. Tradução de Daniel Vieira. 6. ed. São Paulo: Pearson Addison Wesley, 2011. p. 65.

GAUNT, Matt. **Web fundamentals**: Service Workers: uma introdução. on-line, 2017. Disponível em: <<https://developers.google.com/web/fundamentals/primers/service-workers/>>. Acesso em: 13 nov. 2017.

GAUNT, Matt; KINLAN, Paul. **Web fundamentals: O manifesto do aplicativo Web.** on-line, 2017. Disponível em: <<https://developers.google.com/web/fundamentals/web-app-manifest/?hl=pt-br>>. Acesso em: 10 nov. 2017.

GOMES, Antonio Carlos. **Treinamento desportivo: estrutura e periodização.** 2. ed. Porto Alegre: Artmed, 2009. p. 179-193.

GOMES, Jaydson. APIs geniais da web moderna. *In: FILHO, Almir et al. (Org.). Coletânea front-end: uma antologia da comunidade front-end brasileira.* 1. ed. São Paulo: Casa do Código, 2016. cap. 8, p. 116.

GOOGLE DEVELOPERS. **Progressive Web Apps training: core technologies.** on-line, 2017. Disponível em: <<https://developers.google.com/web/ilt/pwa/core-technologies>>. Acesso em: 08 nov. 2017.

_____. **Progressive Web Apps Training: working with IndexedDB.** on-line, 2017b. Disponível em: <<https://developers.google.com/web/ilt/pwa/working-with-indexeddb#introduction>>. Acesso em: 19 nov. 2017.

GUEDES, Gilleanes Thorwald Araujo. **UML 2: uma abordagem prática.** 2. ed. São Paulo - SP: Novatec, 2011. p. 21, 24, 30, 36, 101 e 277.

JUAREZ FILHO. **Web.br 2016 Conferência Web W3C Brasil Progressive Web Apps: melhorando a experiência do usuário utilizando tecnologias web.** on-line. 2016. Disponível em: <<http://conferenciaweb.w3c.br/2016/progressive-web-apps-melhorando-a-experiencia-do-usuario-utilizando-tecnologias-web/>>. Acesso em: 04 nov. 2017.

KITAMURA, Eiji. **Working with quota on mobile browsers: a research report on browser storage - HTML5 Rocks.** on-line. 2014. Disponível em: <<https://www.html5rocks.com/en/tutorials/offline/quota-research/>>. Acesso em: 14 nov. 2017.

LAUDON, Kenneth C.; LAUDON, Jane P. **Sistemas de informação gerencial.** Tradução de Luciana do Amaral Teixeira. 9. ed. São Paulo - SP: Pearson Prentice Hall, 2011. p. 110.

LEPAGE, Pete. **Web fundamentals: seu primeiro Progressive Web App.** on-line, 2017. Disponível em: <<https://developers.google.com/web/fundamentals/codelabs/your-first-pwapp/>>. Acesso em: 01 nov. 2017.

LOPES, Sérgio. **A web mobile: programe para um mundo de muitos dispositivos.** 1. ed. São Paulo: Casa do Código, 2013. p. 13- 16, 19, 20, 22, 25 e 26.

_____. **Aplicações mobile híbridas com Cordova e PhoneGap.** 1. ed. São Paulo: Casa do Código, 2016. p. 7.

MACORATTI, José Carlos. **HTML5 - Armazenamento de dados no Cliente - IndexedDB.** on-line. 2015. Disponível em: <<https://developers.google.com/web/ilt/pwa/working-with-indexeddb#introduction>>. Acesso em: 19 nov. 2017.

MAHEMOFF, Michael. "**Offline**": what does it mean and why should I care? - HTML5 Rocks. on-line. 2013. Disponível em: <<https://www.html5rocks.com/en/tutorials/offline/whats-offline/>>. Acesso em: 04 nov. 2017.

MANIERO, Antonio. **Quais são as diferenças entre aplicação web e aplicação desktop?**. on-line. Disponível em: <<https://pt.stackoverflow.com/questions/187344/quais-s%C3%A3o-as-diferen%C3%A7as-entre-aplica%C3%A7%C3%A3o-web-e-aplica%C3%A7%C3%A3o-desktop>>. Acesso em: 07 out. 2017.

MILETTO, Evandro Manara.; BERTAGNOLLI, Silvia de Castro. **Desenvolvimento de software II**: introdução ao desenvolvimento web com HTML, CSS, JavaScript e PHP. Porto Alegre: Bookman, 2014. p. 4 - 5, 62.

MOZILLA DEVELOPER NETWORK. Mozilla Foudantion. **CacheStorage - Web APIs | MDN**. on-line, 2017. Disponível em: <<https://developer.mozilla.org/en-US/docs/Web/API/CacheStorage>>. Acesso em: 16 nov. 2017.

_____. _____. **IndexedDB | MDN**. on-line, 2017b. Disponível em: <<https://developer.mozilla.org/pt-BR/docs/IndexedDB>>. Acesso em: 19 nov. 2017.

_____. _____. **LocalStorage - Web APIs | MDN**: web storage. on-line, 2017c. Disponível em: <<https://developer.mozilla.org/en-US/docs/Web/API/Storage/LocalStorage>>. Acesso em: 03 nov. 2017.

_____. _____. **Service Worker API - Web APIs | MDN**. on-line, 2017d. Disponível em: <https://developer.mozilla.org/en-US/docs/Web/API/Service_Worker_API>. Acesso em: 12 nov. 2017.

_____. _____. **Using Service Workers | MDN**. on-line, 2017e. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/API/Service_Worker_API/Using_Service_Workers>. Acesso em: 12 nov. 2017.

_____. _____. **HTTP cookies - HTTP | MDN**. on-line, 2017f. Disponível em: <<https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies>>. Acesso em: 14 out. 2017.

_____. _____. **Progressive Enhancement | MDN**. on-line, 2017g. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Glossary/Progressive_Enhancement>. Acesso em: 10 nov. 2017.

_____. _____. **Web App Manifest | MDN**. on-line, 2017h. Disponível em: <<https://developer.mozilla.org/pt-BR/docs/Web/Manifest>>. Acesso em: 10 nov. 2017.

PEREIRA, Caio Ribeiro. **Node.js**: aplicações web real-time com Node.js. 1. ed. São Paulo: Casa do Código, 2014. p. 25.

PILGRIM, Mark. **Local Storage**: dive into HTML5. on-line. 2012. Disponível em: <<https://diveintohtml5.com.br/storage.html>>. Acesso em: 04 nov. 2017.

PINHO, Diego Martins de. **ECMAScript 6: entre de cabeça no futuro do JavaScript**. 1. ed. São Paulo: Casa do Código, 2017. p. 161, 165 e 166.

PRESSMAN, Roger. S. **Engenharia de software: uma abordagem profissional**. Tradução de Ariovaldo Griesi e Mario Moro Fecchio. 7. ed. Porto Alegre: Mc Graw Hill, bookman, AMGH Editora Ltda., 2011. p. 34, 37, 126.

REMOALDO, Pedro. **O guia prático do dreamweaver CS3 com PHP, JavaScript e Ajax**. Lisboa - Portugal: Centro Atlantico, 2008. p. 35.

ROUSE, Margaret. **Web application (Web app) - Definition from WhatIs.com**. on-line, 2011. Disponível em: <<http://searchsoftwarequality.techtarget.com/definition/native-application-native-app>>. Acesso em: 07 out. 2017.

_____. _____. on-line. 2013. Disponível em: <<http://searchsoftwarequality.techtarget.com/definition/Web-application-Web-app>>. Acesso em: 07 out. 2017.

SANTOS, Ana Lúcia Padrão dos; SIMÕES, Antonio Carlos. Educação física e qualidade de vida: reflexões e perspectivas. **Saúde Soc.**, São Paulo, v. 21, n. 1, p. 181-192, jan. 2012.

SILBERSCHATZ, Abraham; KORTH, Henry F.; SUDARSHAN, S. **Sistema de banco de dados**. Tradução de Daniel Vieira. 5. ed. Rio de Janeiro: Elsevier, 2006. p. 4.

SORIANO, L. **Periodização e planificação do treinamento desportivo: uma revisão bibliográfica**, Campinas -SP, 1996. p. 22.

VAN, Anne; HICKSON, Ian. **Offline Web Applications**. on-line. Disponível em: <<https://www.w3.org/TR/offline-webapps/>>. Acesso em: 01 nov. 2017.

W3C. ALABBAS, Ali; BELL, Joshua. **Indexed Database API 2.0**. on-line. Disponível em: <<https://www.w3.org/TR/IndexedDB-2/>>. Acesso em: 20 nov. 2017.

W3SCHOOLS. **HTML5 Web Storage**. on-line. Disponível em: <https://www.w3schools.com/html/html5_webstorage.asp>. Acesso em: 03 nov. 2017.

WHATWG. **HTML Living Standard: web storage**. on-line, 2017. Disponível em: <<https://html.spec.whatwg.org/multipage/webstorage.html>>. Acesso em: 01 nov. 2017.

_____. **Living Standard: offline web applications, 2017b**. on-line. Disponível em: <<https://html.spec.whatwg.org/multipage/offline.html#offline>>. Acesso em: 01 nov. 2017.


APÊNDICE A - Tela de *login* do protótipo




The image shows a login screen for a system named SIGMIT. The screen is enclosed in a light gray border. At the top center, the word "SIGMIT" is displayed in a large, dark purple font. Below the title, there are two input fields. The first is labeled "Nome de Usuário" in a dark purple font, and the text "Aluno" is entered into the field. The second is labeled "Senha" in a dark purple font, and the password is represented by a series of 15 black dots. Below the password field, there is a dark purple button with the word "Acessar" written in white, bold, sans-serif font.

Fonte: Desenvolvido pelo autor.


APÊNDICE B - Tela visualizar sessão de treino do protótipo




SIGMIT | Aluno 

Sessão Treino: 2



Aluno	Aluno
Treinador	Treinador
Número da Sessão de Treino	2
Status da Sessão de Treinamento	A fazer
Microciclo	Recuperativo
Magnitude de Carga	0.15
Características do Microciclo	As cargas entre 10-20%
Data da Sessão de Treino	2017-11-23
Hora de Inicio	17:00:00
Hora de Saida	18:00:00
Local do Treinamento	Outdoor



 Home  Sessão de Treino
 Contato

Fonte: Desenvolvido pelo autor.

APÊNDICE C - Tela enviar sessão de treino do protótipo

SIGMIT | Aluno
☰

Enviar sessão treino: 2

☰ Listar Treinamentos

Aluno	Aluno
Treinador	Treinador
Número da Sessão de Treino	2
Status da Sessão de Treino	<input type="text" value="A"/>
Microciclo	Recuperativo
Magnitude de Carga	0.15
Características do Microciclo	As cargas entre 10-20%
Data da Sessão de Treino	<input type="text" value="2017-11-23"/>
Hora de Início	<input type="text" value="17:00:00"/>
Hora de Saída	<input type="text" value="18:00:00"/>
Local do Treinamento	<input type="text" value="Outdoor"/>
	☑ Enviar Sessão de Treino

🏠 Home
⚙️ Sessão de Treino
☎️ Contato

Fonte: Desenvolvido pelo autor.

APÊNDICE D - Tela de pesquisa das sessões de treino do protótipo

SIGMIT | Aluno
☰

Sessões de Treinamento

☰ Listar Treinamentos

10

▼

resultados por página

Pesquisar

Treinador ↓↑	Sessão ↓↑	Data ↓↑	Local ↓↑	Status ↓↑	
+	Treinador	1	2017-11-23	Academia FATEC	Realizado
+	Treinador	1	2017-11-22	Academia FATEC	Realizado

Mostrando de 1 até 2 de 2 registros (Filtrados de 5 registros)

< 1 >

🏠 Home

📍 Sessão de Treino

☎ Contato

Fonte: Desenvolvido pelo autor.