UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

Aplicações PWA em desenvolvimento móvel

Diogo Valadas Reis Viegas Maia

Mestrado em Engenharia Informática

Versão Pública

Trabalho de projeto orientado por:
Profª. Doutora Maria Beatriz Duarte Pereira do Carmo

2024

Agradecimentos

Gostaria de agradecer à Professora Beatriz Carmo pelo acompanhamento, orientação, auxilio e total disponibilidade, contributos sem os quais cumprir este objetivo não teria sido possível. Agradecer também à EMVENCI na pessoa de Alexandre Aniceto pela possibilidade de desenvolver este trabalho na empresa.

Agradecer aos meus colegas que, como eu, viveram este grande desafio e me apoiaram nos meses deste trabalho e em todos os desafios que enfrentámos. Um obrigado ao Daniel, ao João, à Núria e ao Vasco, sem vocês teria sido certamente mais difícil. Um obrigado ao Óscar pela companhia nas horas de almoço.

Um agradecimento à JUST e a todos os que dela fizeram parte e especialmente aos que comigo viveram mais de perto esta aventura do melhor da vida universitária. Fizeram-me cair, crescer, rir e aprender. São das melhores recordações que daqui levo. A todos os meus amigos, aos da JUST e aos de antes, obrigado.

Um agradecimento muito especial ao meu pai, à minha mãe e ao meu irmão, à minha namorada e a toda a minha família pelo apoio, compreensão e suporte incondicionais neste último ano tão desafiante.

Obrigado a todos pelo apoio.

Ao futuro e ao futuro eu

Resumo

Este trabalho de projeto do Mestrado em Engenharia Informática foi desenvolvido na EMVENCI. O seu objetivo foi construir uma aplicação móvel para *Android* que fosse parte integrante do produto *SaaS* (*Software as a Service*, em português *Software* como serviço) denominada *Cybersecurity Cloud*.

A plataforma *Cybersecurity Cloud* representa o produto principal da empresa e oferece uma solução integrada para a gestão de temas de cibersegurança, para tal, divide-se em diversos módulos que podem ser subscritos (*eLearning*, *phishing*, *policy*, entre outros).

No desenvolvimento da aplicação *Android*, de seu nome *Cybercloud* foram estudadas duas opções de desenvolvimento, utilizando código nativo ou fazendo uso de uma *PWA* (progressive web app), as duas opções foram comparadas dando preferência por aquela que se adequava mais à realidade da empresa e ao caminho futuro a traçar. Foi escolhido o desenvolvimento *PWA* e o trabalho terminou com a sua publicação na *Google Play* para substituir a versão atualmente disponível.

Palavras-chave: *PWA, Angular, Cibersegurança, Android, Bubblewrap*

Abstract

This project in Informatics Engineering was developed at EMVENCI. Its objective was to build a mobile application for Android that would be part of the SaaS (Software as a Service) product called Cybersecurity Cloud.

The Cybersecurity Cloud platform represents the company's main product and offers an integrated solution for managing cybersecurity issues and incidents. It is divided into several modules that can be subscribed by the customer (eLearning, phishing, policy, among others).

In the development of the Android application, named Cybercloud, two development options were studied: using native code or making use of a PWA (progressive web app). Both options were compared, giving preference to the one that best suited the company's reality and future direction. The PWA development was chosen, and the work was concluded with its publication on Google Play to replace the currently available version.

Keywords: *PWA, Angular, Cybersecurity, Android, Bubblewrap*

Conteúdo

Lista de Figuras

Siglas e Acrónimos

API *Application Programming Interface*.

CSS *Cascading Style Sheets*.

HLS *HTTP Live Streaming*.

HTML *Hypertext Markup Language*.

HTTPS *Hypertext Transfer Protocol Secure*.

PDF *Portable Document Format*.

PWA *Progressive Web App*.

SDK *Software Development Kit*.

SMS *Short Message Service*.

SPA *Single Page Applications*.

SaaS *Software as a Service*.

TS *Typescript*.

TWA *Trusted Web Activity*.

UI/UX *User Interface/User Experience*.

aab *Android App Bundle*.

apk *Android Package Kit*.

iOS *iPhone Operating System*.

mvc *Model-View-Controller*.

Capítulo 1

Introdução

Neste capítulo será explicado o contexto de desenvolvimento deste trabalho, a motivação, os objetivos a atingir, as contribuições e uma breve apresentação da estrutura deste documento.

1.1 Contexto

Este trabalho de projeto em Engenharia Informática foi desenvolvido na EMVENCI e orientado na empresa por Alexandre Aniceto e no Departamento de Informática da Faculdade de Ciências da Universidade de Lisboa pela Professora Beatriz Carmo. Serviu de base a plataforma *SaaS* "Cybersecurity Cloud", produto da EMVENCI.

A empresa EMVENCI desenvolve uma plataforma SaaS (Software as a Service) com módulos e funcionalidades voltadas para a cibersegurança como Phish Simulator (Simulacros de Phishing para testar e avaliar os colaboradores), eLearning Cybersecurity & Privacy (Formação em Cibersegurança e Privacidade), Threat & Vulnerability Manager (Gestão de Ameaças e Vulnerabilidades de Segurança), Log Manager (Gestão de Eventos de Segurança), Privacy Manager (Facilitar a gestão de requisitos do RGPD), Policy Manager (Gestão de Politicas de Segurança). A plataforma denominada *Cybersecurity Cloud* possui uma aplicação móvel *Android* e *iOS* de seu nome *CyberCloud*.

1.2 Motivação

Foi decidido pela empresa que a aplicação atualmente disponível para o sistema *Android*, desenvolvida em *NativeScript* (a *framework* de desenvolvimento escolhida pela empresa na altura) teria de ser substituída desde logo pela necessidade de melhoria, otimização e atualização das funcionalidades disponíveis, de forma a acompanhar o evoluir da plataforma. Para essa nova versão a desenvolver surgiu o tema base deste trabalho, a escolha entre um desenvolvimento nativo ou com recurso a uma *PWA*.

Importa referir que a versão para *iOS* da aplicação *CyberCloud*, que estava também desenvolvida em *NativeScript*, foi já desenvolvida nativamente para *iOS*, *Apple TV* e demais sistemas da

Apple tendo servido como tema de um trabalho de projeto do Mestrado em Engenharia Informática de outro colega [27] precisamente por, também no sistema *iOS*, terem sido identificadas possibilidades de melhoria, otimização e atualização da solução até então em vigor. Perante a decisão de substituição e sendo a aplicação *iOS* a mais recente, as funcionalidades e estilo da aplicação a desenvolver (para *Android*) seguirão sempre que possível as da versão *iOS* como se verá com mais detalhes nas próximas secções.

Segundo um estudo da *Statcounter* de 2019, citado pela *research.com* [8] 50.48% de todo o tráfego *web* provém de dispositivos móveis. Os dispositivos fixos, os tradicionais computadores, representam apenas 46.51% do tráfego e, por último, o tráfego proveniente de *tablets* fecha o pódio com apenas 3% de origem de todo o tráfego na internet. Numa publicação sobre o mesmo assunto a *Kinsta*[19], apesar de apresentar um ”pódio”diferente para a situação na Europa, vai mais longe afirmando que o tráfego proveniente de dispositivos móveis na Europa cresceu 568% desde 2016.

O imediatismo da informação é algo de extrema importância, especialmente quando falamos de cibersegurança. É importante que através dos dispositivos móveis consigamos ter acesso a informação, de forma rápida e simples. Isto, aliado a todas as estatísticas de utilização de dispositivos móveis já referidas, assim como, o dia-a-dia ”ligado”a um *smartphone* (Figura 1.1) tornam evidente que uma empresa que se queira competitiva, atual e presente tem de ter soluções otimizadas para os dispositivos móveis mais utilizados.

Perante a crescente popularidade na utilização dos dispositivos móveis e visto que na solução atual em *NativeScript* foram identificadas necessidades de melhoria, tornou-se fundamental o desenvolvimento de uma alternativa.

1.3 Objetivo

O objetivo deste trabalho foi desenvolver uma nova aplicação que permitisse substituir a versão atual (em *NativeScript*), estudando para isso as diferenças entre o desenvolvimento nativo e *PWA (progressive web app)* de forma a identificar a melhor arquitetura tendo em conta os requisitos técnicos, funcionais e a realidade e preferência da empresa.

Quando se fala em desenvolvimento nativo estamos a falar em desenvolvimento na linguagem de programação *Kotilin* [34], uma linguagem de programação orientada a objetos desenvolvida pela *Jetbrains*. Em 2017 foi anunciada pela *Google* como linguagem de programação oficial do *Android*.

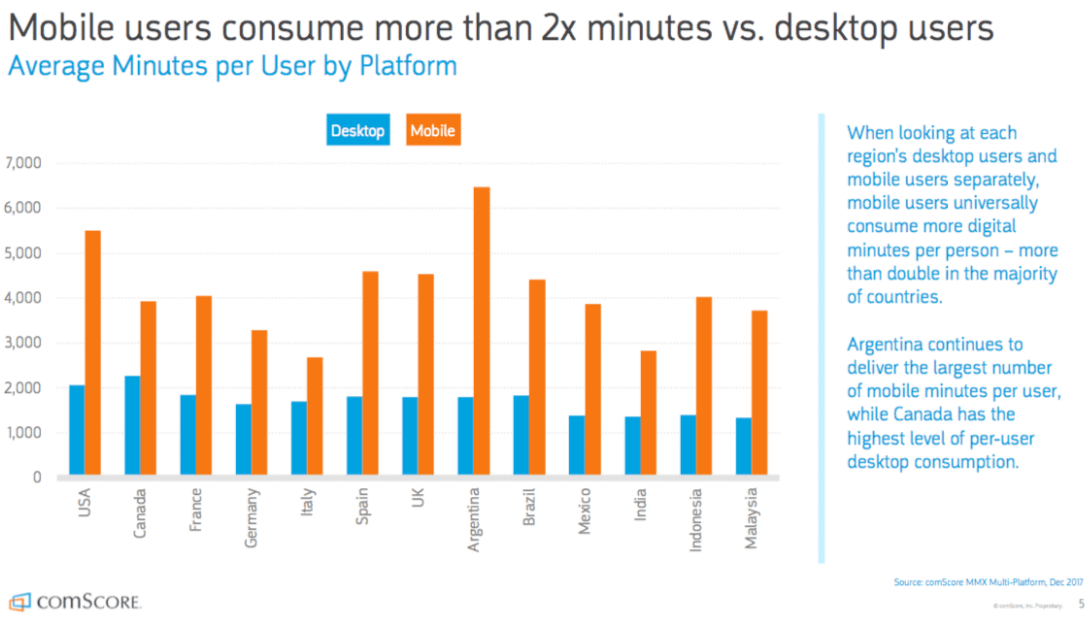

Figura 1.1: Tempo de utilização de dispositivos móveis quando comparado com computadores [4]

Uma *PWA* (Progressive Web Application) é uma aplicação *web* que tem como objetivo entregar uma experiência nativa, utilizando o browser. Neste caso específico falamos do desenvolvimento *Angular* [13], uma plataforma para construção de aplicações *web* baseada em *TypeScript*, também responsabilidade da *Google*. Esta é a forma como a plataforma da EMVENCI está construida, como perceberemos em mais detalhe nos próximos capítulos.

1.4 Contribuições

Neste trabalho foi comparado o desenvolvimento nativo com o desenvolvimento de uma *PWA* com o objetivo de encontrar a melhor solução para a construção da aplicação pretendida. O resultado final deste trabalho foi uma aplicação que permite aos utilizadores da plataforma o acesso a uma versão mais simples e rápida da plataforma, onde podem aceder aos conteúdos e gerir algumas definições. Como consequência foi construida uma nova versão da *CyberCloud*, que irá substituir a versão *NativeScript*. Foi entregue como produto final uma aplicação publicada na loja de aplicações do *Android* que substituiu a implementação atual da *CyberCloud*.

1.5 Estrutura do documento

Após o capítulo da introdução, este documento está organizado da seguinte forma:

- Capítulo 2 – Conceitos e Contextualização - Será apresentada uma breve descrição da plataforma, passando por exemplos dos seus módulos, a sua arquitetura atual e a metodologia de trabalho adotada na empresa;
- Capítulo 3 – Nativo ou *PWA* - Serão descritas as funcionalidades e exploradas as vantagens

e desvantagens de ambas as formas de desenvolvimento e serão apresentados os resultados da decisão nativo ou *PWA*, assim como os problemas associados à organização do código;

- Capítulo 4 – Implementação da solução *PWA* - Neste capítulo serão abordadas especificidades da solução *PWA* e apresentada a metodologia de organização de código adotada no repositório;
- Capítulo 5 – Portal de administração - Será apresentado o trabalho desenvolvido na parte do portal de administração da aplicação móvel;
- Capítulo 6 – Portal do utilizador - Será apresentado o trabalho desenvolvido na parte do portal do utilizador da aplicação móvel;
- Capítulo 7 – Finalização e publicação na *Google Play* - Neste capitulo será descrito todo o processo associado à finalização do projeto e publicação da aplicação na loja oficial;
- Capítulo 8 – Conclusão - Aqui serão apresentados os detalhes finais, resumido o trabalho e indicado o trabalho futuro a realizar.

Capítulo 2

Conceitos e Contextualização

Entendida a motivação e os objetivos deste trabalho, neste capítulo serão abordados aspetos importantes quanto à sua contextualização, os conceitos importantes para compreender o trabalho desenvolvido. Neste capítulo será apresentada uma breve descrição da plataforma usando para isso um dos seus módulos como exemplo e também a forma como esta está construida, isto é, a sua arquitetura. Para contextualizar o ambiente de execução deste trabalho falaremos da metodologia de trabalho usada na empresa.

A aplicação móvel foi baseada na plataforma *SaaS Cybersecurity Cloud* que possui módulos e funcionalidades voltadas para a cibersegurança. A plataforma está construida utilizando *Angular* para a parte de *frontend* e é já uma *PWA*, no entanto não otimizada para o acesso em dispositivos móveis e por isso acesso através destes não é possível. Existe já uma aplicação desenvolvida nativamente para *iOS* e por isso, as funcionalidades da aplicação a desenvolvida neste trabalho, e por uma questão de coerência, seguiram as funcionalidades da aplicação iOS.

Para *Android* há uma aplicação desenvolvida em *NativeScript* mas por existirem falhas e a aplicação em *iOS* ser posterior e mais atualizada, apesar deste trabalho se centrar no desenvolvimento para *Android* a aplicação *iOS* é que será considerada para completar o levantamento de requisitos, sendo a aplicação para o ecossistema da *Apple* aquela que servirá de exemplo sobre qual o produto final a atingir. Um das principais razões que levou a empresa a abandonar o desenvolvimento *NativeScript* foi o facto da versão para *iOS* não apresentar a performance esperada e por isso, para este sistema a empresa decidiu desenvolver nativamente uma aplicação para que fosse possível tirar partido das vantagens dos equipamentos móveis da *Apple*. Seguindo esta decisão, para o ecossistema *Android*, pretende-se então estudar uma alternativa.

2.1 Conceitos

Ao longo deste documento serão abordados vários termos, conceitos e ferramentas que serão explicados de seguida.

- **Desenvolvimento/código nativo (Kotlin)** - Entende-se como desenvolvimento/código nativo, aquele código que é escrito utilizando uma linguagem de programação voltada para determinadas características e arquitetura de um conjunto restrito de dispositivos. Este código é, em teoria, otimizado para determinado sistema operativo/*hardware*, o que resulta numa solução mais rápida e adaptada as especificidades do equipamento final. Neste caso, falando de *Android*, na EMVENCI, uma aplicação em código nativo seria escrita em *Kotlin* (por escolha da própria empresa), uma linguagem que em 2017 foi considerada pela *Google* como oficial do *Android* [26].

- **Repositório** - Termo utilizado para referir o código e a sua maneira de gerir. É uma solução centralizada que permite o controlo de versões, a colaboração, o acesso ao histórico. É utilizado pela EMVENCI e facilita o trabalho em equipa, assim como o acompanhamento de versões e o *tracking* de eventuais problemas decorrentes das alterações efetuadas. É exemplo disso o *GitHub*. No caso específico da empresa é utilizado o *Bitbucket*[7] uma solução mais comercial.

- **NativeScript** - É um framework de código aberto para desenvolver aplicações móveis para as plataformas *iOS* e *Android*. A solução atual da *CyberCloud* para *Android* está construida com esta *framework*.

- **PWA (progressive web app)** [14] [28] - São baseadas no *browser* e foram criadas para preencher o espaço entre os *websites* tradicionais e a experiência oferecida por uma aplicação desenvolvida nativamente. São *websites* desenhados para que se comportem de forma similar a uma aplicação nativa. Uma *PWA* corre no browser o que significa que ao contrário de uma aplicação nativa, uma PWA não se limita a um sistema operativo. O seu objetivo é que seja possível tirar proveito das vantagens de desenvolvimento *web* e nativo, oferecendo uma experiência rápida, eficiente e integrada. Uma aplicação PWA pode ser ”instalada”diretamente a partir dos navegadores, ainda que neste caso na EMVENCI, caso seja esta a opção escolhida, deve também estar disponível na loja de aplicações do *Android*.

- **Angular** [13] - É uma *framework* de *frontend* que tem por base o *TypeScript* e é mantida pela Google. O seu propósito principal é ser utilizada para desenvolver *single page applications (SPA)*. Como a maioria das *frameworks*, as vantagens do *Angular* também contribuem bastante para a facilidade de desenvolvimento, oferecendo um *standard* para que os *developers* mantenham e organizem o código independentemente do tamanho da aplicação ou solução desenvolvida. Segue o padrão *Model-View-Controller (MVC)*. A *framework* é conhecida pela abordagem que permite a modularidade e reutilização de código. Por ser a *framework* em que está feito o *frontend* do produto *SaaS* da EMVENCI, será o conceito mais abordado ao longo deste trabalho.

- **Angular Atomic/Atomic Design** [6] [1] - É uma metodologia de design que ajuda na modularidade e reutilização do código, ”partindo”o código em pequenos componentes reu-

tilizáveis, chamados átomos. A combinação destas unidades mais simples dá origem às moléculas, organismos, *templates* e páginas. As vantagens deste modelo são a sua consistência e eficiência, assim como o facto de facilitar a escalabilidade e reutilização, útil para aplicações mais complexas. É a arquitetura utilizada na solução *Angular* atual (o *website* atual do *software*) e o seu princípio fundamental é a reutilização de pequenas porções de código. A sua organização é detalhada mais à frente.

- **Multitenancy** [12] - É uma arquitetura em que uma aplicação é desenvolvida para servir múltiplos utilizadores, os chamados *tenants* (os clientes da *EMVENCI*). Estes utilizadores, apesar de compartilharem a mesma instância da aplicação, os seus dados e configurações são mantidos separadamente e de forma independente, podendo até corresponder a instâncias diferentes da base de dados. No contexto de cada utilizador, estes não conhecem os outros com quem partilham os recursos. Num contexto de *cloud computing*, *multitenancy* significa que múltiplos clientes de uma *cloud* utilizam os mesmos recursos mas não se conhecem uns aos outros. Vejamos um exemplo de um banco, vários clientes guardam lá o seu dinheiro, mas apesar disso, não interagem uns com os outros e nem sequer se conhecem. A arquitetura *multitenant* é bastante utilizada em produtos *SaaS*, como o da EMVENCI.

- **HLS (HTTP Live Streaming)** [11] - é um protocolo de comunicação HTTP desenvolvido pela *Apple*, utilizado em *media players, browsers,* dispositivos móveis *e streaming*. É considerado o formato de *streaming* mais popular. Este protocolo divide os arquivos de vídeo em arquivos HTTP menores distribuindo-os utilizando o protocolo HTTP. Uma das principais vantagens é o facto deste *streaming* ser capaz de adaptar a qualidade do vídeo de forma a se moldar às condições da rede. Isto permite que a qualidade do vídeo seja alterada dinamicamente, resultando numa experiência suave e estável mesmo em condições de rede variáveis.

- **SaaS (Software as a Service)** [9]- É uma forma de distribuição e comercialização de software. Segundo um estudo da *McKinsey & Company* citado pela *TechTarget*, o crescimento deste mercado é expectável que atinja 200 mil milhões de dólares em 2024. O fornecedor do software garante a disponibilização do sistema, normalmente mediante o pagamento de uma subscrição. Os clientes destas soluções não estão preocupados com o *setup* ou manutenção do serviço, sendo que em troca desta subscrição, obtém acesso a uma solução ”chave na mão”. Nestes produtos é normalmente utilizada a abordagem do *multi-tenant*, como já referido. Exemplos de produtos *SaaS* populares são a *Netflix*, o *Shopify*, o *Zoom*. A EMVENCI comercializa o seu produto também desta forma.

- **Cybersecurity Cloud ou plataforma** - Estes termos serão utilizados para fazer referência ao produto da EMVENCI que servirá de base a este trabalho: uma solução *SaaS* integrada de cibersegurança, baseada em *multitenancy*.

- **Cybercloud ou aplicação** - Estes termos serão utilizados para fazer referência à aplicação

móvel associada à *Cybersecurity Cloud*, quer em *Android*, quer em *iOS*.

2.2 Breve descrição da plataforma

A plataforma que serviu de base a este trabalho e que representa o *core business* da EMVENCI é a *Cybersecurity Cloud*, uma plataforma *SaaS* e *multi-tenant*. Como já anteriormente explicado é uma plataforma com funcionalidades voltadas para a cibersegurança onde estão disponíveis para os clientes vários módulos consoante a subscrição adquirida. Estes módulos permitem aos clientes e respetivos colaboradores tirar proveito das funcionalidades da plataforma, nomeadamente dos conteúdos de *eLearning*, das campanhas de *phishing* fictícias e dos vários outros módulos que permitem gerir variados temas relacionados com cibersegurança. Esta plataforma representa uma solução integrada para diversas áreas de cibersegurança de uma empresa. O modelo de negócio da EMVENCI baseia-se no licenciamento desta plataforma aos clientes empresariais. A subscrição é efetuada por módulos escolhidos pelo cliente. A utilização da plataforma pode ser separada em dois grandes *use cases*. Existe o acesso ao *user portal*, feito por um colaborador da empresa cliente e o acesso ao *admin portal*, feito por um administrador da empresa cliente.

No *user portal* um utilizador consegue aceder aos seus conteúdos (ex: *eLearning*), certificados, métricas, campanhas *policy* e respetivas informações assim como gerir os detalhes básicos da sua conta como alterar a linguagem, a *password*, habilitar a autenticação 2 fatores (autenticação que só fica concluída com um segundo código, normalmente comunicado por SMS ou por uma aplicação autenticadora) ou ativar/desativar a subscrição de *email*.

Na parte do *admin portal*, o administrador da organização faz toda a gestão dos módulos ativos na sua subscrição, cria campanhas, cria grupos, atribui campanhas e gere todas as definições e funcionalidades. Dentro do nível de acesso à parte administrativa é também possível personalizar as permissões para que sejam concedidos vários níveis de acesso aos administradores.

Do ponto de vista mais técnico a plataforma está desenvolvida no seu *frontend* em *Angular*. A comunicação com o *backend* é feita através de uma API e neste trabalho não entraremos em detalhe sobre a forma como o *backend* está construido. A arquitetura utilizada é arquitetura *Angular Atomic* que será explicada em detalhe de seguida. Na implementação é utilizado *multitenancy* onde cada *tenant* representa um cliente com uma licença subscrita, associado a um subdomínio.

São clientes da EMVENCI as empresas que queiram adquirir uma licença para a utilização do *Cybersecurity Cloud*, sendo que essas empresa terão utilizadores padrão (todos os colaboradores) e utilizadores administradores.

2.3 Exemplo de módulo: *Phishing Simulator*

Como exemplo, apresenta-se em seguida, de forma breve, um módulo da plataforma: o *Phishing Simulator*. Este módulo permite que sejam geradas campanhas de *phishing* fictícias que terão com alvo um conjunto definido de utilizadores. A seleção deste alvo pode ser feita utilizador a utilizador ou de forma dinâmica, isto é, escolhendo uma lista de critérios que a pessoa deve cumprir para ser alvo desta campanha (Ex: enviar o *e-mail* de *phishing* apenas aos utilizadores que representam maior risco e maior impacto na organização). A campanha tem diferentes opções de personalização, podendo a sua ”cara”ser o *e-mail* de um banco, da *Amazon*, etc. Estas campanhas depois de agendadas/lançadas farão chegar à caixa de correio eletrónico dos utilizadores definidos como alvos um *e-mail* fictício que poderá por exemplo, requerer a introdução dos dados do *homebanking* para avaliar uma transação suspeita. O administrador da plataforma pode a qualquer momento aceder aos detalhes e estatísticas da campanha lançada, conseguindo ver que utilizadores receberam o e-mail, abriram o e-mail, acederam ao seu link e que utilizadores submeteram informações no formulário falso. Toda esta interação possível dos utilizadores está associada também a um sistema de pontos, onde são perdidos pontos caso seja desencadeada uma ação a evitar (ex: submeter informação no formulário) e ganhos pontos caso, por exemplo, o e-mail seja reportado à equipa responsável. Este módulo pode ser integrado com o de *eLearning* para que os utilizadores com menor pontuação e maior risco, tenham de completar determinadas horas de conteúdo didático previamente escolhido.

Para percebermos o contexto e a importância da integração do módulo de *phishing* na plataforma, pode ser importante olharmos para alguns dados, como os da Figura 2.1. Segundo estudos citados pela *Techopedia* [24] 36% de todas as falhas e fugas de dados registadas nos Estados Unidos foram responsabilidade de ataques de *phishing*. Estima-se que todos os anos 86% das empresas sofram com este tipo de ataques. Entre 2020 e 2021 foi observado um aumento de 345% de *websites* de *phishing*. Só em 2022 foram reportados ao FBI mais de 300 mil ataques. Estima-se que exista em média um gasto de 4,91 milhões de dólares, por parte das empresas, em cada ataque. O *phishing* por *e-mail* é relatado como a forma mais comum deste ataque sendo que a principal ”cara”destes, são falsos *e-mails* que se fazem passar por empresas de entregas postais e de encomendas (mais de 50%) e, logo a seguir, *e-mails* que se fazem passar por bancos e outras instituições financeiras (com pouco mais de 30%). Os ataques de *phishing* foram considerados pelo *IC3* como o crime mais frequente em 2022. Em 2022 a *Kaspersky* analisou mais de 150 milhões de *e-mails* sendo de cerca de 25% eram originários da Rússia, com o segundo lugar a pertencer à Alemanha, com cerca de 14%.

Este é, como vimos, um risco real, recorrente e crescente e quando falámos de empresas, dependendo do tamanho destas, todos os colaboradores apresentam um risco, sendo cada um deles um alvo para estas campanhas. Como vimos, é importante conseguir conter este risco, pois as consequências de não o prever, podem ser imprevisíveis.

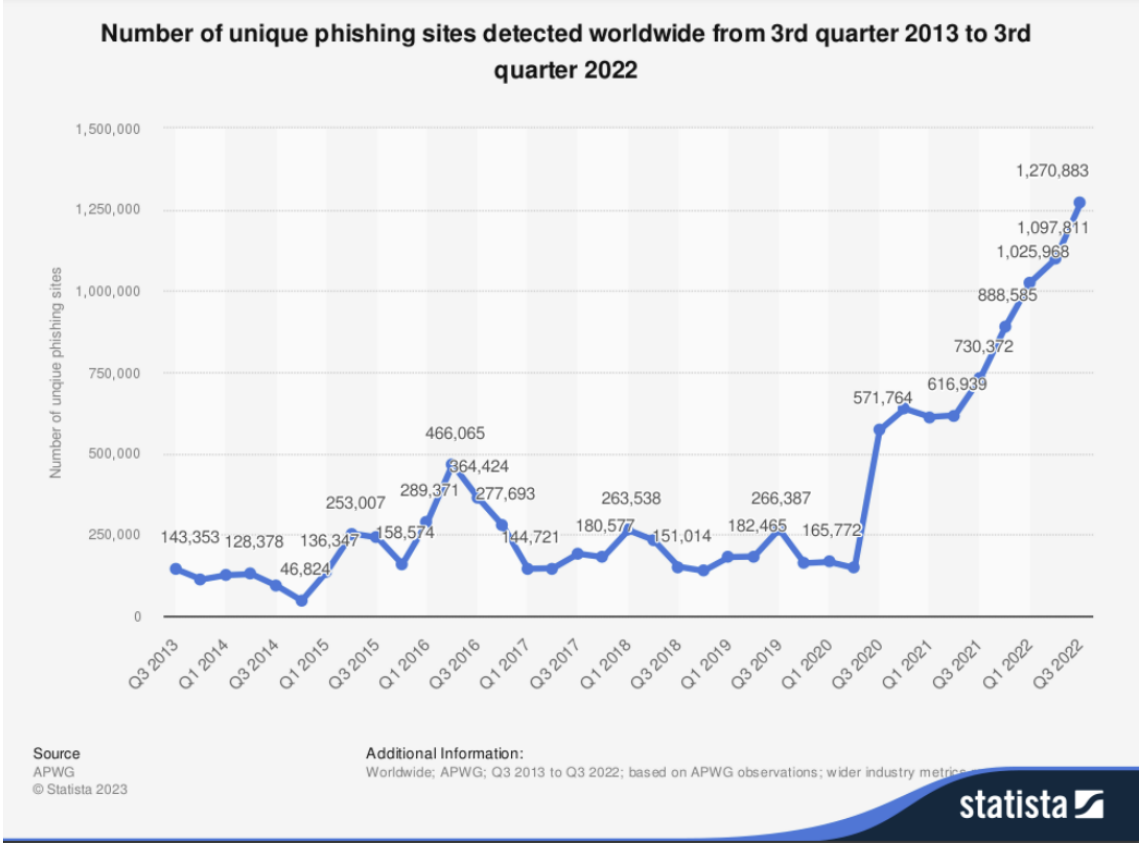

Figura 2.1: Evolução do número de *sites phishing* em todo o mundo [24]

Este é apenas um exemplo de um módulo da plataforma, onde pode ser avaliado o risco, digamos, humano e perceber se toda a estrutura da organização está preparada e conhece todos os procedimentos a adotar perante um possível caso de tentativa de *phishing*, nomeadamente naqueles com níveis de acesso mais elevado e por isso, com maior impacto em caso de comprometimento ou falha na deteção de um ataque deste tipo. À semelhança de outros módulos na plataforma, este permite aos administradores retirar conclusões e informações muito importantes no que toca à segurança de uma organização e na gestão do seu risco.

2.4 Arquitetura

A arquitetura utilizada na solução Angular atual é a denominada "*Angular Atomic*"[6] e o seu princípio fundamental é a reutilização de pequenas porções de código, os denominados átomos. Na sua organização divide-se em átomos, moléculas, organismos, *templates* e páginas, que vão ser detalhados de seguida (Figura 2.2).

Átomos - A unidade mais básica, é reutilizada em todos os locais e normalmente representa um elemento HTML simples e com um estilo também ele simplificado. Por exemplo uma *string* de texto.

Moléculas - Um conjunto de átomos. Por exemplo uma *string* de texto e um botão.

Os átomos e as moléculas são os elementos mais simples, reutilizáveis e que recebem diretivas *Input* (por exemplo, no caso de um botão, o texto desse mesmo botão) e emitem diretivas *Output* (no mesmo caso do botão, seria por exemplo, o evento que ocorre no clique).

Organismos - São grupos de moléculas já com algum nível de complexidade. Por exemplo, o *header* de um site pode ser considerado um organismo, uma vez que é composto por várias moléculas (botões, caixa de pesquisa, menu de páginas, entre outros)

Templates - Podem ser considerados capas de um componente e são normalmente apenas compostos por HTML e CSS.

Páginas - Tratam do *input/output* das aplicações e fazem uso de serviços.

Grande parte da plataforma *Cybersecurity Cloud* já está implementada seguindo esta arquitetura, sendo que a empresa caminha para a sua total utilização.

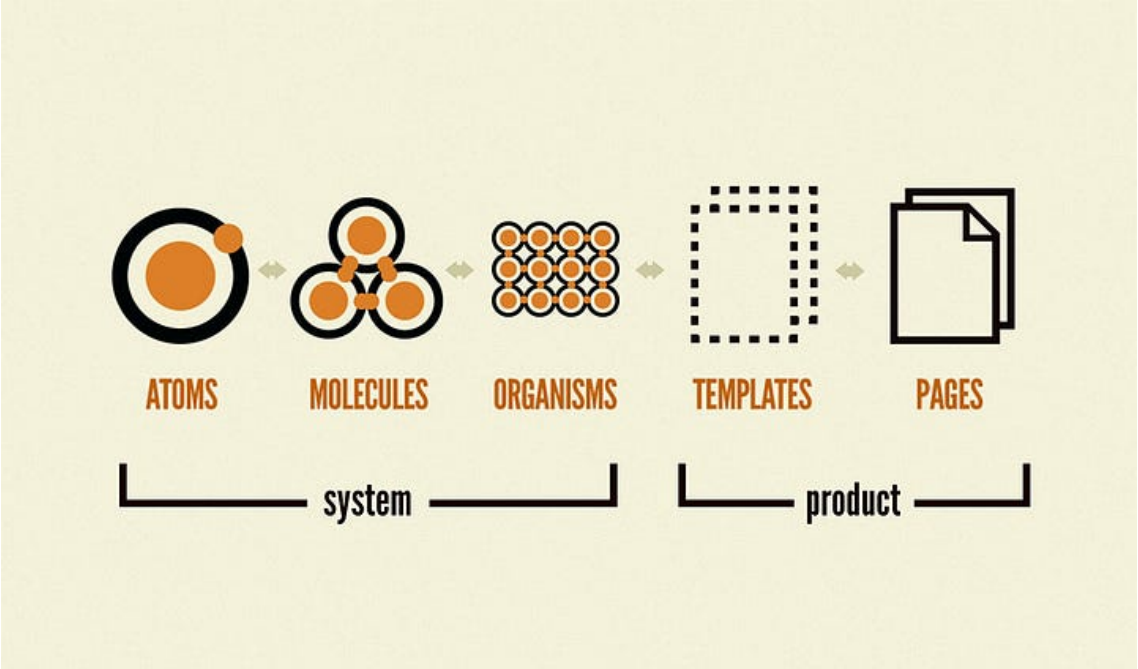

Figura 2.2: Angular Clean Architecture [6]

Trabalhar com arquitetura atómica é trabalhar sobre uma boa base de componentes e avançar na sua combinação e complexidade criando moléculas, *templates* e páginas - mais complexas. As grandes vantagens são [1]:

- **Eficiência** - Ao reutilizar componentes, evita-se tarefas repetitivas e que consomem mais tempo;
- **Escalabilidade** - O princípio desta arquitetura é a construção de átomos. Como já foi referido estes componentes são altamente reutilizáveis, permitindo que estes sejam utilizados onde e quando forem necessários, em qualquer lugar da aplicação;
- **Consistência** - Ao promover a reutilização de código estamos a contribuir para a criação de uma interface de utilizador mais consistente e apelativa;
- **Modularidade e Crescimento** - Esta arquitetura é muito útil em grandes projetos pois permite que estes componentes reutilizáveis sejam atualizados e modificados no futuro de forma fácil e eficaz.

Em suma, esta metodologia modular e de reutilização facilita bastante o trabalho e a manutenção de grandes projetos, contribuindo para a construção de uma solução consistente e eficiente, com uma facilidade de adaptação a custos reduzidos (quer monetários, quer de tempo despendido).

2.5 Metodologia de trabalho

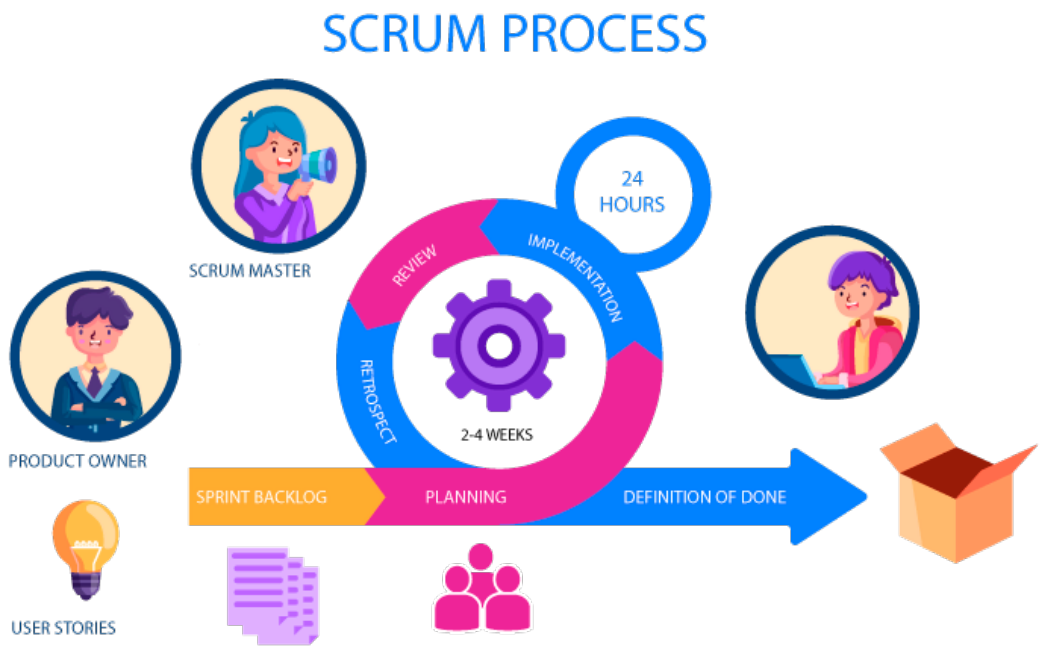

Figura 2.3: O processo da metodologia *Scrum* [25]

A metodologia de trabalho na EMVENCI é a metodologia *Scrum* [25]. O Scrum é um framework ágil que organiza o desenvolvimento de software em iterações chamadas de *sprints*. O

processo, demonstrado na Figura 2.3 inicia-se com um *backlog*, uma lista priorizada de funcionalidades. Depois de efetuado um planeamento, inicia-se a fase de implementação. Esta fase corresponde à execução por parte dos *developers* de pequenos *tickets* criados durante a fase de planeamento onde cada um corresponde a um problema a resolver ou uma nova funcionalidade a implementar. Estes *tickets* não devem demorar mais que o tempo da *sprint* a ser resolvidos caso contrário devem ser repartidos por outros *tickets* de menor complexidade. Cada *sprint*, no contexto da empresa tem a duração de 2 semanas, resulta num incremento do produto. Esta metodologia promove a flexibilidade, a colaboração e entrega contínua, o chamado *continuous integration continuous delivery*. Esta metodologia tem vantagens na flexibilidade, controlo e gestão e é adequada para projetos de todos os tamanhos. Na EMVENCI diariamente existe uma reunião de acompanhamento do trabalho que está a ser e será efetuado. Alinhado com o início de cada *sprint* há uma reunião de planeamento da *sprint* a iniciar. Todo o trabalho associado é gerido por um *scrum master*, a pessoa responsável pela gestão da implementação desta metodologia.

Para a organização do código foi utilizado um repositório no serviço *Bitbucket* [7], uma ferramenta de gestão de código, que equivale à solução empresarial do *Github*, desenvolvida pela *Atlassian*. A utilização do produto da *Atlassian* permite uma integração total entre a gestão do código e a gestão de *tickets* com o *software Jira* [5] também da mesma empresa.

Capítulo 3

Nativo ou *PWA*

Na primeira parte deste capítulo será explicada a fase de análise deste trabalho começando por um levantamento das funcionalidades da aplicação, a elaboração da documentação interna do trabalho a desenvolver e a comparação entre as opções de código nativo ou *PWA*. Na segunda parte, será abordado o início da implementação, passando desde logo pela escolha entre o desenvolvimento nativo ou de uma *PWA*. Tomada essa decisão avaliaremos a forma como foi organizado o código para minimizar ao máximo o impacto de uma nova estrutura de código no repositório.

3.1 Análise

3.1.1 A solução para *Android*

Descritas brevemente as principais funcionalidades da plataforma e sendo o imediatismo da informação e o seu fácil acesso algo que já se assumiu de grande importância, ainda mais perante uma crescente tendência na utilização de dispositivos móveis, o desenvolvimento de uma aplicação que seja segura, estável, confiável é importante. A aplicação deverá permitir a um utilizador aceder a funções básicas como visualizar os conteúdos de *eLearning*, responder a questionários, visualizar políticas e respetivos conteúdos. Uma lista de funcionalidade detalhada será apresentada mais à frente e, como referido, seguirá, sempre que possível, as funcionalidades presentes na solução iOS atual. Na parte do administrador, deve permitir o acesso a estatísticas e estados das campanhas criadas. O objetivo deste trabalho foi precisamente este, sendo que a primeira decisão a tomar foi se esta aplicação seria desenvolvida de forma nativa ou com recurso a uma *PWA*.

3.1.2 Funcionalidades da aplicação *Android*

O início do trabalho deste projeto propriamente dito deu-se com o levantamento das funcionalidades a implementar. Como já referido anteriormente, existe uma aplicação *iOS* desenvolvida nativamente e as suas funcionalidades servirão de base para definir aquilo que serão as funcionalidades da aplicação. É também importante que as funcionalidades sejam baseadas naquelas já disponíveis na versão *desktop* da plataforma, como perceberemos nos capítulos mais à frente. Na sua maioria estão divididas em funcionalidades para o *Admin Portal*, para o *User Portal* e comuns aos dois. A maioria das funcionalidades estão disponíveis na utilização da plataforma através de

um computador, sendo esta aplicação móvel, na sua grande maioria, resultado de uma redução das funcionalidades disponíveis num computador. Sendo assim são, de forma resumida, as seguintes:

- *Admin Portal* - Acesso apenas para utilizadores com as permissões administrativas
  - Consultar as campanhas de *eLearning*, *Policy*, *Phishing*, respetivos detalhes e estatísticas quer gerais quer associadas aos utilizadores alvo.
- *User Portal* - Acesso para todos os utilizadores
  - Aceder aos conteúdos de eLearning, sejam estes vídeos ou conteúdo PDF;
    - No acesso aos conteúdos de vídeo deve ser possível selecionar o idioma do áudio e das legendas;
  - Aceder aos conteúdos *Policy*;
  - Aceder e responder aos questionários;
- Comuns - funcionalidades gerais da aplicação
  - Autenticar-se, para isso, tem de indicar o *link* do *tenant*.
  - Aceder à área *"Platform News"*;
  - Aceder às suas notificações;
  - *Light e Dark mode*
  - Aceder às definições;

Um administrador acumula obviamente as funcionalidades disponíveis para o utilizador, dito normal.

3.1.3 Documentação

Um dos primeiros passos, ainda durante a etapa de levantamento de funcionalidades, foi a escrita da documentação para a aplicação a desenvolver. Esta parte é fundamental para que finda este trabalho, a aplicação possa ser mantida por qualquer *developer* que venha a ter essa responsabilidade. A documentação escrita continha, entre outros, os seguintes tópicos:

- *Business Requirements*
  - *Objective* - Um resumo daquilo que é o objetivo;
  - *Background and Strategic Fit* - O porquê da aplicação ser importante;
  - *Assumptions*;
  - *Requirements* - Lista de *user stories* e funcionalidades relacionadas;
  - *Success Metrics* - A lista de objetivos a cumprir;
  - *Implementation Details* - Detalhes técnicos do processo de implementação.;
  - Entre outros.

No decorrer deste trabalho, todos os avanços, decisões e instruções foram devidamente documentados para que fosse fácil dar continuidade à manutenção do projeto por outro *developer* que não conhecesse as suas particularidades. Através da leitura da documentação escrita qualquer *developer* consegue adaptar-se ao ambiente de desenvolvimento, desenvolver e dar *deploy* das mudanças implementadas.

3.1.4 Nativo vs *PWA*

Dos primeiros passos deste trabalho foi perceber qual seria a melhor solução para o desenvolvimento da solução esperada, se *Kotlin* [34] (como linguagem nativa), se *Angular (PWA)* (atualmente utilizado pela empresa). De forma a testar ambas as implementações foram desenvolvidas duas aplicações simples, uma em *Kotlin* e outra em *Angular PWA* contendo apenas um ecrã de *login* e após autenticado, um *player* de vídeo *HLS* [11], *HTTP Live Streaming*, sendo este o protocolo de comunicação de *streaming* utilizado no conteúdo de vídeo da plataforma.

Logo à partida, o desenvolvimento em código nativo terá obviamente vantagens quando ao seu desempenho visto ser um desenvolvimento voltado especificamente para um conjunto certo de características. O desenvolvimento nativo para *iOS* foi escolhido precisamente para tirar proveito desta vantagem. No que toca ao desenvolvimento Angular PWA e no caso particular da EMVENCI, a grande vantagem é a possibilidade de tirar partido da organização atómica do repositório da *Cybersecurity Cloud*, reutilizando código e evitando desenvolvimento duplicado. Será interessante, de forma teórica, comparar as vantagens e desvantagens de uma e outra abordagem [14].

Vantagens PWA

- **Custo de desenvolvimento** - Desenvolver uma *PWA* é mais económico, visto que não existe a necessidade, neste contexto, de manter duas versões completamente distintas da mesma aplicação (*iOS e Android*), levando a uma poupança financeira e de tempo.
- **Manutenção** - Como numa *PWA* lidamos apenas com uma base de código, manter este tipo de aplicações é mais fácil e económico. Um desenvolvimento nativo obriga a uma manutenção de dois códigos totalmente distintos e por isso é desde logo mais dispendioso.
- **Segurança** - Apesar de as aplicações nativas apresentarem mais opções de segurança, as aplicações *PWA* serão desde logo mais seguras porque correm sobre o protocolo *HTTPS*.
- **Atualizações imediatas** - Como uma *PWA* na sua essência é um *website* a disponibilização de atualizações é imediata em todas as plataformas.
- **Vantagem principal no contexto da empresa** - Nesta abordagem é possível tirar proveito da arquitetura *Angular Atomic* e reutilizar os seus componentes ao mesmo tempo que se tira proveito de uma equipa de *developers* já familiarizada com esta tecnologia.

Desvantagens PWA

- **Acesso limitado a funcionalidades nativas** - Aplicações *PWA* podem ter mais dificuldade em interagir com funcionalidades nativas de um equipamento, por exemplo, a leitura de um sensor como o acelerómetro.

- **Interação limitada** - As interfaces e funcionalidades de uma aplicação *PWA* podem não ser tão otimizadas e fluídas como numa aplicação nativa.

- **Desvantagem principal no contexto da empresa** - Se mal implementada, pode adicionar uma grande complexidade à organização de código atual tornando a versão da plataforma para o computador mais pesada.

Vantagens Nativo

- **Velocidade** - Aplicações nativas são mais rápidas e eficientes, visto que estão preparadas especificamente para as características onde estão a ser executadas.

- **Melhor experiência** - Uma aplicação nativa é capaz de oferecer uma melhor experiência uma vez que é capaz de interagir com o *hardware* do equipamento e retirar todo o seu potencial.

- **Acessibilidade** - Aplicações nativas podem ser descarregadas diretamente das lojas de aplicações, tornado o seu acesso mais fácil do que numa abordagem *PWA* convencional - ainda que neste caso particular, como veremos, tal não se aplica.

Desvantagens Nativo

- **Desenvolvimento mais lento** - Como são necessárias versões para cada um dos sistemas operativos móveis mais utilizados (*Android e iOS*), o desenvolvimento de uma aplicação nativa é desde logo mais demorado.

- **Atualizações constantes e difíceis** - Por ser necessário manter duas bases de código, aplicar novas atualizações para novas funcionalidades ou correção de erros é bastante mais complexo numa aplicação nativa.

- **Custo mais elevado** - Por exigir a manutenção de pelo menos duas linguagens diferentes com os respetivos recursos humanos necessários, a manutenção de uma aplicação nativa é mais dispendiosa financeiramente e até em termos de tempo requerido.

- **Desvantagem principal no contexto da empresa** - A empresa não possui recursos humanos com experiência no desenvolvimento *Kotlin* nem esta linguagem faz parte do portefólio, não sendo usada em nenhum dos seus produtos.

Em suma, nos diversos artigos consultados [2] as vantagens e desvantagens apontadas a cada uma das soluções de desenvolvimento são bastante semelhantes, sendo possível chegar a uma conclusão teórica na escolha de uma destas opções. Se o desempenho for um fator importante o modelo de desenvolvimento a seguir deve ser o nativo, pois como já referido este é o único que permite tirar proveito total das características dos equipamentos. Caso não se considere prioritário o desempenho e se pretenda desenvolver uma solução mais equilibrada, em custo e facilidade de construção/manutenção a escolha deve ser a *PWA*. Como na solução pretendida pela empresa não existiu uma exigência de tirar partido das funcionalidades nativas (ex: sensores) do dispositivo, todas as vantagens já enumeradas juntando ao facto de já existir uma *PWA* para o computador desenvolvida, colocaram a solução *PWA* como a preferencial.

3.1.5 O desenvolvimento nativo

Um dos principais desafios no desenvolvimento nativo é lidar com o formato de vídeo HLS (HTTP Live Streaming) e o facto de todos os conteúdos de vídeo da plataforma estarem encriptados. O desenvolvimento nativo representa também um desafio no sentido em que, desenvolver código nativamente exige a manutenção de mais um repositório, com a necessidade de contratar novos recursos humanos especializados. A nível pessoal representou um desafio adicional por não dominar nem nunca ter tido contacto com *Kotlin* o que atrasou também o processo de desenvolvimento da aplicação de teste. Ao desenvolver esta aplicação ficaram evidentes as dificuldades quer a nível pessoal quer para a empresa. *Kotlin* não faz parte da lista de linguagens utilizadas pela empresa nos seus produtos, sendo esta uma das grandes desvantagens se se decidisse avançar com esta abordagem.

3.1.6 O desenvolvimento PWA

Como a plataforma *Cybersecurity Cloud* é desenvolvida em *Angular*, na prática esta já é uma *PWA*, sendo uma enorme vantagem na adoção desta abordagem para o desenvolvimento da aplicação *Android*. Escolhendo esta abordagem é também possível tirar partido de todas as vantagens já previamente mencionadas do *atomic design*, poupando imenso tempo no desenho de uma nova interface, pois seria apenas necessário (em teoria) reutilizar os componentes já desenvolvidos para a versão *web*. Em termos teóricos aquilo que seria necessário, se se decidisse esta abordagem, seria adaptar a plataforma e o seu *design*, reduzindo-lhe as funcionalidades, para que fosse entregue uma solução mais otimizada para ser acedida em dispositivos móveis, algo até então inexistente.

Um dos principais desafios previamente identificados no desenvolvimento *PWA* é decidir qual deve ser a organização do código para que a aplicação *mobile PWA* e a aplicação para computador consigam coexistir sem comprometer a compreensão e organização do repositório atual. Isto acontece porque a aplicação terá de partilhar a mesma base de código que a solução já existente para o browser, sendo que a versão mobile será uma adaptação desta e com menos funcionalidades. Um dos maiores receios da empresa na adoção desta abordagem é precisamente o risco que se poderá

correr de comprometer o repositório atual, se a este formos adicionar a o código necessário para o funcionamento da versão mobile da plataforma.

3.2 Os desafios da implementação

3.2.1 A escolha entre Nativo ou *PWA*

O desenvolvimento em código nativo representou a nível pessoal um desafio acrescido, por ser uma tecnologia que não domino e sobre a qual tive de aprender. Da mesma forma, a empresa não possui experiência no desenvolvimento *Kotlin* pelo que a adoção desta opção de desenvolvimento era desde logo pouco provável. Sendo uma grande vantagem da arquitetura *Angular Atomic* a reutilização de código através de componentes, fazia todo o sentido que uma nova solução para o *Android* tirasse proveito da vantagem que é ter já uma plataforma desenvolvida em *Angular* e da qual se podem reutilizar componentes. Por todas as vantagens e pontos a favor do desenvolvimento *PWA* e não havendo uma necessidade expressa de tirar proveito das vantagens do desenvolvimento nativo (ex: acesso a sensores), a o desenvolvimento *PWA* foi a solução escolhida. Neste caso e seguindo esta abordagem e uma vez que a plataforma Cybersecurity Cloud é já uma *PWA* o trabalho necessário será a sua adaptação para o acesso em dispositivos móveis, atualmente não é possível desde logo porque não se encontra otimizada e porque a plataforma ao perceber que o acesso se dá de um dispositivo móvel, exige a utilização da respetiva aplicação.

3.2.2 A organização do código

Tendo optado pelo desenvolvimento da aplicação *Cybercloud* utilizando a abordagem *PWA*, o trabalho que se seguiu foi entender como integrar o seu desenvolvimento no repositório da plataforma atual de forma a que o impacto na organização do código e estrutura fossem mínimos e que esta nova base de código não interferisse com o desenvolvimento e o funcionamento da plataforma para computador.

Quando se abordou a necessidade de estudar a melhor forma de integrar o código com o repositório atual é porque visto que a plataforma já é uma PWA e o pretendido será que a aplicação móvel seja uma interface simplificada, com funcionalidades reduzidas dessa mesma *PWA*, é necessário que a versão *PWA mobile* e a versão para o computador atual coexistam no mesmo repositório. Existiu obviamente a possibilidade de criar toda uma nova base de código *Angular*, num novo repositório, onde seria desenvolvida a *PWA mobile*, mas esta abordagem seria desde logo desaconselhada pois deitaria por terra todas as vantagens já oportunamente abordadas da adoção do *design* atómico.

Desta forma,estudou-se a melhor opção para integrar ambas as soluções (preferencialmente) num mesmo repositório. Tendo sido tidos em conta os seguintes pontos:

- Criar um novo repositório?

  - A integração do código na solução atual não é viável?
  - Os ganhos de organização compensam a perda das vantagens da reutilização dos componentes já criados e de todas as vantagens do *design* atómico?

- Manter o repositório atual da solução *Angular*?
  - Que ficheiros manter e que ficheiros criar?
  - Criar ficheiros .html que corresponderão à versão mobile?
  - Como se devem chamar e como integram a estrutura atual do repositório?
  - É necessário criar os ficheiros .css correspondentes ou é possível reutilizar os da versão para computador?
  - É possível reutilizar a lógica dos ficheiros .ts (ficheiros TypeScript, responsáveis pela lógica da aplicação, nomeadamente as chamadas à API que suporta o *backend*)?
  - É necessário criar ficheiros .ts específicos para a versão *mobile*?

Estes foram os maiores desafios. Foi desde logo evidente a importância de uma boa resposta a estas questões para que a integração da PWA *mobile* no mesmo repositório, não afetasse o *workflow* e o caminho de otimização de desenvolvimento a ser seguido pela empresa. A resposta a estas perguntas ocupou grande parte do tempo da fase de análise e investigação deste trabalho.

Para estes testes e de forma a responder as questões já previamente levantadas foi tida em conta a abordagem que menos impactava o *workflow* atual da equipa de *frontend* e que apresentasse os maiores níveis de organização, estruturação e compreensão de código quando introduzida no repositório atual. A arquitetura atómica, já abordada e para qual a empresa está também a transitar apresenta importantes vantagens na organização e estruturação do código, úteis para grandes projetos. Uma futura solução para a aplicação *mobile PWA* não poderia comprometer essas vantagens nem interferir com a transição a decorrer, para um repositório mais organizado e fácil de compreender pela equipa. Todas as opções possíveis para organização do código foram discutidas com a equipa de desenvolvimento *frontend*, ao longo da fase de investigação, para que se conseguisse chegar a um consenso sobre a melhor abordagem e também para ter em conta a perspetiva de quem no futuro terá a responsabilidade de manter a aplicação *Android* desenvolvida.

Capítulo 4

Implementação da solução *PWA*

Capítulo oculto devido à confidencialidade do projeto.

Capítulo 5

Portal de administração

Capítulo oculto devido à confidencialidade do projeto.

Capítulo 6

Portal do utilizador

Capítulo oculto devido à confidencialidade do projeto.

Capítulo 7

Finalização e publicação na *Google Play*

Capítulo oculto devido à confidencialidade do projeto.

Capítulo 8

Conclusão

Capítulo oculto devido à confidencialidade do projeto.

Bibliografia

[1] 1shubham7. Atomic Design. https://www.geeksforgeeks.org/atomic-design/, 11 2023.

[2] Ricardo Santos Anacleto. Comparação do desempenho de diferentes abordagens para o desenvolvimento de aplicações móveis. Master's thesis, IPT - ESTT - Escola Superior de Tecnologia de Tomar, 2018.

[3] Angular. Angular Internationalization. https://angular.dev/guide/i18n.

[4] AnyforSoft. Native App or Progressive Web App: A Choice That Matters. https://www.techopedia.com/phishing-statistics, 05 2020.

[5] Atlassian. Jira — Issue Project Tracking Software - Atlassian. https://www.atlassian.com/software/jira.

[6] Fer Ayguavives. Angular Clean Architecture. https://medium.com/weekly-webtips/angular-clean-arquitecture-d40e9c50f51, 01 2021.

[7] Atlassian Bitbucket. Atlassian Bitbucket - Oficial page. https://bitbucket.org/.

[8] Imed Bouchrika. Mobile vs Desktop Usage Statistics for 2023. https://research.com/software/mobile-vs-desktop-usage, 04 2023.

[9] Wesley Chai. Software as a Service (SaaS). https://www.techtarget.com/searchcloudcomputing/definition/Software-as-a-Service, 10 2022.

[10] Anjali Chaudhary. Progressive web apps vs native apps: What should you pick? https://www.turing.com/blog/progressive-web-apps-vs-native-apps/, 03 2023.

[11] Cloudflare. What is HTTP Live Streaming? — HLS streaming. https://www.cloudflare.com/learning/video/what-is-http-live-streaming/.

[12] Cloudflare. What is multitenancy? https://www.cloudflare.com/learning/cloud/what-is-multitenancy/.

[13] Chinmayee Deshpande. What is Angular?: Architecture, Features, and Advantages. https://www.simplilearn.com/tutorials/angular-tutorial/what-is-angular, 07 2023.

[14] Roman Furman. Pwa vs. native app - which is better in 2023? https://www.sommo.io/blog/pwa-vs-native-app-which-is-better-in-2023, 07 2023.

[15] Google. Adding Your Progressive Web App to Google Play. https://developers.google.com/codelabs/pwa-in-play, 09 2022.

[16] Instinctools. Pwa vs native app: Pros and cons in 2023. our experts weigh in. https://www.instinctools.com/blog/pwa-vs-native-app/.

[17] Intercom. Intercom. https://www.intercom.com/.

[18] jsverse. Transloco Angular i18n. https://jsverse.github.io/transloco/.

[19] Kinsta. Mobile vs. Desktop Market Share and Usage Statistics. https://kinsta.com/mobile-vs-desktop-market-share/.

[20] Kevin Kurniawan. Bubblewrap: How To Publish Your Progressive Web App (PWA) In The Google Play Store. https://medium.com/@kevinkurniawan97/introduction-to-pwa-twa-dcc2368268a3, 01 2022.

[21] Google Chrome Labs. Bubblewrap. https://github.com/GoogleChromeLabs/bubblewrap.

[22] Christian Liebel. Bubblewrap: How To Publish Your Progressive Web App (PWA) In The Google Play Store. https://www.thinktecture.com/en/pwa/twa-bubblewrap/, 07 2020.

[23] Node.js. Node.js. https://nodejs.org/pt.

[24] Jo Rushton. 50+ Phishing Statistics You Need to Know – Where, Who What is Targeted. https://www.techopedia.com/phishing-statistics, 07 2023.

[25] Bhaskar S. Scrum Methodology Explained: An Introduction for Agile Teams. https://www.nimblework.com/agile/scrum-methodology/, 07 2024.

[26] Maxim Shafirov. Kotlin on android. now official. https://blog.jetbrains.com/kotlin/2017/05/kotlin-on-android-now-official/, 05 2017.

[27] Miguel Silva. Migração de aplicação móvel híbrida para desenvolvimento nativo. Master's thesis, Faculdade de Ciências da Universidade de Lisboa, 2022.

[28] Adobe Experience Cloud Team. Mobile vs. Desktop Market Share and Usage Statistics. https://business.adobe.com/blog/basics/pwa-definition, 12 2023.

[29] NativeScript Team. NativeScript Documentation - What is NativeScript? https://docs.nativescript.org/.

[30] Jackie Tran. Pwa vs native app and how to choose between them. https://www.magestore.com/blog/pwa-vs-native-app-and-how-to-choose-between-them/, 12 2023.

[31] Google web.dev. Enhancements PWA. https://web.dev/learn/pwa/enhancements.

[32] Wikipédia. Android SDK. https://en.wikipedia.org/wiki/Android_SDK, 07 2024.

[33] Wikipédia. Java Development Kit. https://pt.wikipedia.org/wiki/Java_Development_Kit, 01 2024.

[34] Wikipédia. Kotlin. https://pt.wikipedia.org/wiki/Kotlin, 05 2024.