MySNS Carteira

Especificação Técnica

Versão 1.0

16-Jan-2018

MySNS Carteira – Documentação

Indice

Introdução

Sobre esta documentação

Este documento pretende ser um guia completo para a MySNS Carteira. Contém toda a documentação para o funcionamento da perpectiva do utilizador bem como todos os recursos para o *setup* e desenvolvimento.

Partilhamos esta mesma documentação com objetivo da comunidade possa colaborar connosco no desenvolvimento e evolução da MySNS Carteira.

A arquitectura da aplicação, em especial do backend, está em contante evolução, pelo que faremos esforços para atualizar estas informações sempre que possível. Não podemos garantir a atualização contante deste documento. No entanto, estamos sempre disponíveis para fornecer mais informações.

O que é a MySNS Carteira?

A MySNS Carteira, reúne a informação de saúde do cidadão numa aplicação residente no seu *smartphone*, construída de acordo com o interesse do cidadão, que escolhe concretamente a informação pretende guardar na MySNS Carteira.

A MySNS Carteira é uma aplicação "vazia de conteúdo", cabendo ao cidadão, de forma voluntária e ativa, escolher a informação que pretende nela incluir. É uma plataforma de disponibilização de informação de saúde, associada ao cidadão através do seu número de Utente SNS e validada com a informação presente no Registo Nacional de Utente (RNU), onde o cidadão associa "cartões" específicos por componentes informativas do seu interesse.

A MySNS Carteira é construída de forma a garantir os dois pilares essenciais da disponibilização de informação em formato eletrónico, de acordo com as boas práticas internacionais de segurança:

1. Segurança (*Security* e *Safety*) – toda a informação é guardada de forma segura, usando standards internacionais, nomeadamente: *AES, RSA*;
2. Identidade Digital do Cidadão (*eID*) – na ativação da aplicação é garantido que quem acede é quem diz que é.

Open Source

Este FAQ pretende responder a questões comuns relacionadas com Open Source, abordando o uso de Open Source Software (OSS) bem como a disponibilização de código.

1. O que é Open Source?

Por *software Open Source* entende-se todo o programa informático cujo código fonte seja de acesso livre e universal e cuja licença ofereça a todos sem exceção, as seguintes quatro liberdades:

- A liberdade de estudar o funcionamento do programa e de o adaptar a novos problemas;
- A liberdade de distribuir o programa a terceiros;
- A liberdade de utilizar o programa para qualquer fim;
- A liberdade de melhorar o programa e de tornar as modificações públicas, em benefício de toda a comunidade.

2. Porquê usar tecnologias Open Source?

Tendo em vista a necessidade de proteger a informação dos cidadãos, a implementação de tecnologias e sistemas *Open Source* assegura a transparência e o conhecimento profundo do funcionamento das mesmas. A não dependência de fabricantes externos, par além de promover a integração de empresas e comunidades tecnológicas, fomenta a sinergia organizacional e a partilha de conhecimento. Por poder ser adaptado às necessidades de cada projeto, pode proporcionar uma maior fiabilidade e melhor experiência para os programadores - e em consequência, para os utilizadores.

Da não dependência de fabricantes, resulta uma redução bastante significativa de custos, que podem ser alocados noutras funcionalidades que melhorem o serviço prestado aos mesmos utilizadores.

Paralelamente, verifica-se também uma maior interoperabilidade - uma maior liberdade para integrar com outros sistemas de Gestão Integrada (ERP). No *Close Source*, por vezes, por se tratarem de produtos de empresas diferentes (muitas vezes, concorrentes), os *softwares* proprietários costumam ser pouco flexíveis na sua interação e compatibilidade com outras aplicações.

3. Como é possível ter suporte sem pagar pelo produto?

O suporte é fornecido pela comunidade que se cria à volta da tecnologia - há um debate aberto, sugestões de diversas pessoas e um melhor *debugging* - a probabilidade de encontrar um erro é maior, bem como a probabilidade de encontrar uma solução em menos tempo. Ao existir uma comunidade *Open Source*, a velocidade com que correções são escritas é bastante superior. Foi o caso de um problema descoberto na implementação TCP/IP de qualquer sistema operacional, em 1996. A correção para *Linux* foi publicada em 20 minutos, enquanto que para outros sistemas demorou pelo menos 2 dias úteis. Há dezenas de casos semelhantes.

Estas comunidades de programadores geralmente são heterógenas - são programadores que não pertencem necessariamente à mesma organização.

Um bom exemplo é a Linux Foundation, uma organização sem fins lucrativos com reconhecimento mundial com milhares de colaboradores, que trabalham para o desenvolvimento de tecnologias e resolução de problemas de segurança críticos. Esta fundação open-source tem centenas de projetos colaborativos open-souce e é financiada por empresas como Microsoft, Bloomberg, Intel, Huawei, IBM, Cisco, Fuitsu, Oracle, Facebook, Accenture, Tshiba, Vmware, Adobe, Amazon Web Services, Bosch, ebay, Google entre outras.

No entanto, existem muitas outras fundações OpenSource de sucesso (Apache Software Foundation, Eclipse Foundation, Mozilla Foundation) e outros projetos independentes (Joomla, Wordpress)

4. Qual o interesse das empresas em ter serviços com código aberto?

Em prol da transparência, não basta apenas utilizar tecnologias Open Source - mas sim criar código aberto. Numa geração onde se procura a transparência, ter produtos OSS é, cada vez mais, sinónimo de boa publicidade, funcionando como um impulsionador de uma imagem positiva.

Quando um projeto Open Source é atrativo, atrai contribuidores externos - permitindo avançar no projeto de forma mais rápida e com menos custos. Ao ter mais utilizadores, haverão cada vez mais casos de uso a serem explorados, resultando em um código mais robusto. Para além dos contribuidores individuais, poderá também estimular o espirito colaborativo entre empresas - ao facultar código, facilita a aprendizagem entre pares. A titulo de exemplo, podemos referir

a Box e o Facebook, duas empresas que trabalharam em conjunto para resolver problemas em uma ferramenta que tinham em comum - HHVM (open-source PHP runtime).

5. Porque é que alguém quereria desenvolver para um projeto open source sem ter retorno monetário?

Nos projetos Open Source, o retorno monetário direto nunca é o objetivo. Quando uma equipa de programadores de uma empresa ou entidade está a desenvolver um projeto recorrendo a diversas tecnologias, por vezes depara-se com problemas que impedem o progresso do projeto. Ao usar tecnologias open source, a equipa não tem que esperar pela reposta da empresa que gere o software - há mais facilidade em reportar o erro e, se possível, contribuir para a resolução do problema.

O desenvolvimento do software/tecnologia Open Source depende de programadores que contribuem por diferentes motivações:

- Porque utilizam o software/tecnologia e sentem necessidade de colaborar com a resolução dos problemas que surgem;
- Liberdade de personalização - permitindo desenvolver novas aplicações que se adequem melhor às necessidades da sua empresa;
- Pelo desafio técnico;
- Pelo reconhecimento;
- Porque trabalham em organizações parceiras do projeto.

6. Mas é código aberto! Como é que pode ser seguro?

Ao adquirir um software / licença fechada, tudo o que o cliente tem são promessas dos fornecedores sobre o nível de segurança que envolve a aplicação. Na verdade, raramente se sabe o que é feito com a informação recebida. Ao ter um software open source, é possível auditar o código e assim, ter uma certeza maior do nível de confiabilidade do produto.

Apesar de existirem diferenças entre software Open Source e Closed Source que afetam a segurança - havendo desvantagens tanto no Open Source como no Closed Source, estes fatores tendem a equilibrar-se, não representando diferenças significativas neste ponto.

Segundo o Communications-Electronics Security Group do Governo do Reino Unido, ***Não há diferença estatística significativa nas taxas de vulnerabilidade entre software livre e software proprietário***, embora a motivação para investir em segurança seja diferente em cada modelo, assim como as vulnerabilidades. Este afirmação é replicada na *OSS Toolkit Security Note* publicada pelo Governo

Britânico - ***Open Source, como uma categoria, não é mais ou menos seguro do que o Closed Source***.

7. Qual é a estrutura de uma comunidade Open Source? Não há o risco de haverem pessoas mal-intencionadas?

Nos projetos open source não deixa de haver uma hierarquia - existe um sistema de governação. Todas as submissões (pull request) têm que ser revistas e aprovadas. O trabalho de um programador é revisto e auditado por outros.

Um estudo publicado por Kevin Crowston and James Howison (*Hierarchy and centralization in Free and Open Source Software team communications* - School of Information Studies Syracuse University) que procurou analisar as interações de correcção de bugs no sistema de *bug tracking* em mais de 150 projetos Open Source com mais de trinta mil participantes, conclui que

Os projetos são na sua maioria bastante hierárquicos em quatro medidas de hierarquia, consistentes com pesquisas anteriores, mas contrárias à imagem popular desses projetos. Além disso, verificamos que o nível de centralização está negativamente correlacionado com o tamanho do projeto, sugerindo que projetos maiores se tornam mais modulares.

A imagem abaixo, publicada no mesmo artigo, apresenta uma estrutura simplificada tipica de uma comunidade Open Source (*onion-like structure*).

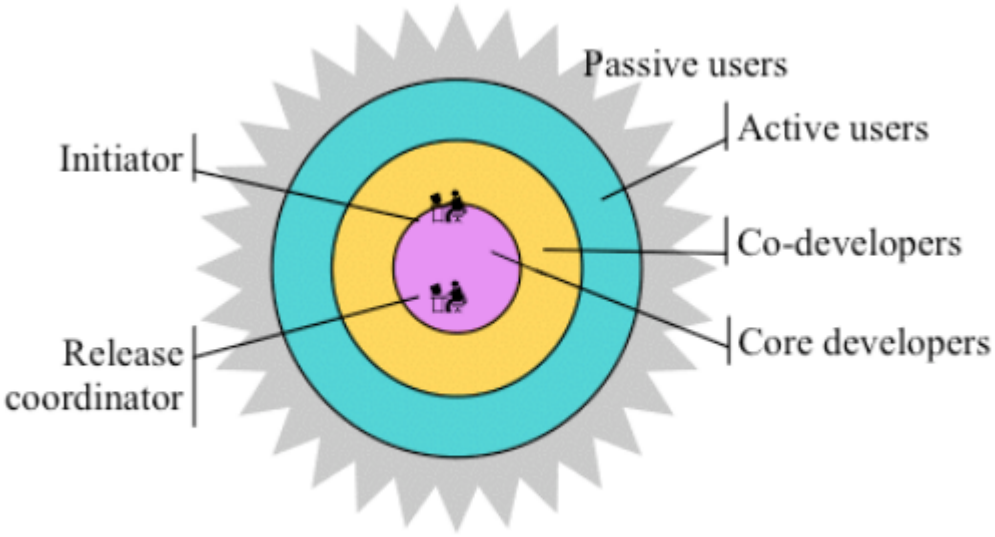


Figure 1. A synthesized FLOSS development team structure.

8. O que dizem as instituições governamentais?

Em 2014, a Comissão Europeia publicou o "Open source software strategy 2014-2017", que visa reforçar o papel do software de código aberto para muitos dos

seus principais serviços e soluções de software TIC. A estratégia renovada coloca uma ênfase especial nos contratos públicos, na contribuição para os projetos de software de código aberto e no fornecimento de software de código aberto na Comissão. Entre as 10 recomendações, destacam-se as seguintes:

- *A Comissão deverá continuar a adotar formalmente,(...) o uso de tecnologias e produtos OSS*
- *OSS tem um importante papel em projetos e-Government e deverá ser considerado no âmbito da framework dessas atividades.*
- *Para o desenvolvimento interno de novos sistemas de informação, em especial nos casos em que a distribuição está prevista por terceiros fora da infraestrutura da CE, o OSS deve ser a escolha preferida e ser utilizado sempre que possível*

Em 2016, a Comissão Europeia tornou público o inventário das soluções open source utilizadas pela Comissão e pelo Parlamento Europeu, sendo esta uma das metas do projeto "EU Free and Open Source Software Auditing" (EU-FOSSA).

A EU-FOSSA pretende oferecer uma abordagem sistemática para que as instituições da UE garantam que o software crítico amplamente utilizado possa ser confiável. O projecto contribuirá para reforçar o contributo das instituições da UE para assegurar e manter a integridade e a segurança dos principais programas informáticos de código aberto. Neste âmbito, anunciou que iria providenciar auditorias de segurança gratuitas para projetos que utilizavam Apache HTTP Server e KeePass (Password manager), dois dos softwares mais utilizados nos projetos da Comissão Europeia e UE.

As páginas comunitárias da EU-FOSSA fornecem informações sobre o estado e os resultados do projeto de auditoria de software livre e Open Source da UE. Entre os documentos publicados no âmbito do EU-FOSSA, procurou-se analisar diversas ferramentas, boas práticas e metodologias de desenvolvimento de software, pretendendo desenvolver *guidelines* para o uso de OSS e incentivar a sua utilização.

Em **Portugal**, na Resolução do Conselho de Ministros n.º 12/2012, for aprovado o plano global estratégico de racionalização e redução de custos com as TIC (PGETIC) na Administração Pública, elaborado pelo Grupo de Projeto para as Tecnologias de Informação e Comunicação (GPTIC).

Entre as medidas de estímulo ao crescimento económico encontram -se a adoção de software aberto nos sistemas do Estado, a melhoria dos processos e soluções de compras públicas, a disseminação internacional de metodologias, de soluções TIC e de conhecimento através de clusters de competitividade nacionais e, no âmbito da estratégia de Administração Aberta, a ampla **disponibilização de informação do sector público em formatos reutilizáveis**, através de projetos como o dados.gov.pt, favorecendo a coprodução de serviços com a sociedade civil, com valor acrescentado para o Estado e para a economia.

Em particular, destaca-se a **Medida 21: Adoção de software aberto nos sistemas de informação do Estado**, que visa promover a utilização de software aberto nos sistemas de informação da AP sempre que a análise à maturidade e o custo total de posse revelem ser mais favoráveis.

No âmbito desta medida e em linha com a estratégia da Direção -Geral da Informática da Comissão Europeia para a adoção progressiva de soluções open source, devem ser identificadas as ferramentas que devem ser, desde já, utilizadas pela AP com carácter de recomendação ou obrigatoriedade. Este estudo, que não pode ser dissociado do Catálogo de Software do Estado (ver Medida 17), promoverá a utilização de software aberto, produzido pelo Estado, privados ou sociedade civil, de acordo com a licença europeia para software aberto, a EUPL, aprovada pela Comissão Europeia

O PGETIC possui uma plataforma de controlo de indicadores, calculados de acordo com os dados de execução reportados pelos Ministérios e pelas entidades responsáveis pelas Medidas de implementação transversal do PGETIC. Nesta plataforma web, e possível consultar a taxa de execução global e por ministério, a poupança quantitativa efetuada. Até à data, a medida apresenta uma taxa de execução global superior a 50%, .

9. Software livre pode ser licenciado?

Sim, pode ser. Uma licença de software livre, à semelhança do software proprietário, é um documento que determina quais ações o utilizador pode ou não executar em relação a um determinado software.

A diferença é que, em uma licença de software proprietário toda a cópia, redistribuição, ou modificação são estritamente proibidas, podendo ter como sequência processos judiciais. Para contornar as restrições anteriormente referidas, deve-se contactar a entidade que detém os direitos do software, para que este dê permissão para o fazer, ou então adquirir uma licença para cada um dos casos anteriormente descritos.

No caso do software livre, existe uma grande variedade de licenças, que se adaptam a casa projeto, seja gratuito ou comercial. A primeira licença de software livre foi criada por Richard Stallman na Free Software Foundation (FSF), no âmbito do projeto GNU.

A FSF publica diferentes licenças escritas com o propósito de promover e preservar a liberdade do software.

- GNU General Public License (GPL)
- GNU Lesser General Public License (LGPL)
- GNU Affero General Public License (AGPL)
- GNU Free Documentation License (FDL)

No entanto, existem outras licenças Open Source bastante populares.

- MIT License A licença MIT permite que o software seja tratado sem restrições para o uso, modificação e distribuição. Desta forma, pode ser utilizada tanto em projetos de software livre, quanto em projeto de software proprietário. No texto desta licença não existe copyright, desta forma outros grupos podem modificar a licença, com o objetivo de atender as suas necessidades.
- Mozilla Public License A Mozilla Public License é uma licença para software livre de código aberto. Esta licença define que o código-fonte copiado ou alterado sob ela deve continuar sob a mesma licença. Entretanto, é permitido que este código seja combinado em um software com arquivos proprietários. Além disso, é possível criar uma versão proprietária de um código sob a licença Mozilla. Esta licença também permite a redistribuição do código produzido, mas obriga a inclusão de citação do autor.
- Apache License
- BSD 2 Clause
- BSD 3 Clause

10. Exemplos de Projetos Open Source / baseados em Open Source

Tal como supramencionado, a Linux Foundation é um dos maiores exemplos de comunidades open source. Graças a esta comunidade, desenvolveram-se tecnologias e serviços que agora são utilizados em todo o mundo. Deixaremos dois exemplos:

Node.JS - tecnologia utilizada na SPMS, é um interpretador de código JavaScript server-side e tem uma das maiores comunidades open-source do mundo, com mais de 4 milhões de utilizadores ativos. Em parte, graças à fundação Linux.

Let's Encrypt - um serviço gratuito de certificados de segurança, que procura contribuir para uma internet mais segura e acessível. É aberto e não está sobre o controlo de nenhuma empresa. Cresceu graças ao apoio de dezenas de entidades como a Mozilla, Cisco, Chrome e Facebook

Paralelamente, a Apache Software Foundation também teve bastantes projectos populares. Entre eles:**Apache Tomcat** - Um dos mais populares open source Java Application Servers.

Apache HTTP Server - servidor HTTP mais utilizado no mundo

OpenOffice.org - conjunto de aplicativos e ferramentas de produtividade

Outros projetos Open Source independentes:

OpenSSL - projeto de Open Source que fornece um conjunto de ferramentas completo para os protocolos TLS (Transport Layer Security) e Secure Sockets Layer (SSL). É também uma biblioteca de criptografia.

WordPress Foundation - plataforma Open Source de blogging e gerenciamento de conteúdo para a web, com milhares de plugins e temas publicados todos os dias

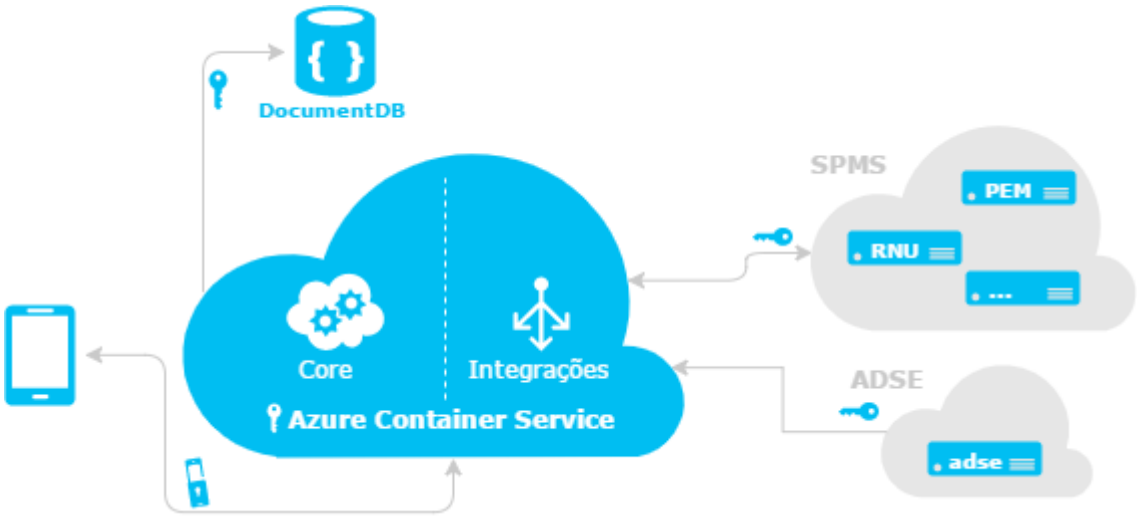
Em Portugal, existem diversos projetos nacionais baseados em open source (entenda-se - que usam tecnologias open source). Alguns destes são divulgados pela ESOP - Associação de Empresas de Software Open Source Portuguesas

- CRM - Gestão de Dados na Saúde - CHMNP Egas Moniz
- Sistema Integrado de Gestão - Conselho Superior da Magistratura
- Fly Tap - Tap Portugal
- Web Portal - Fundação Calouste Gulbenkian
- Lisboa Participa - Camara Municipal de Lisboa
- Plataforma de Gestão de Processos - - A3ES - Agência de Avaliação e Acreditação do Ensino Superior

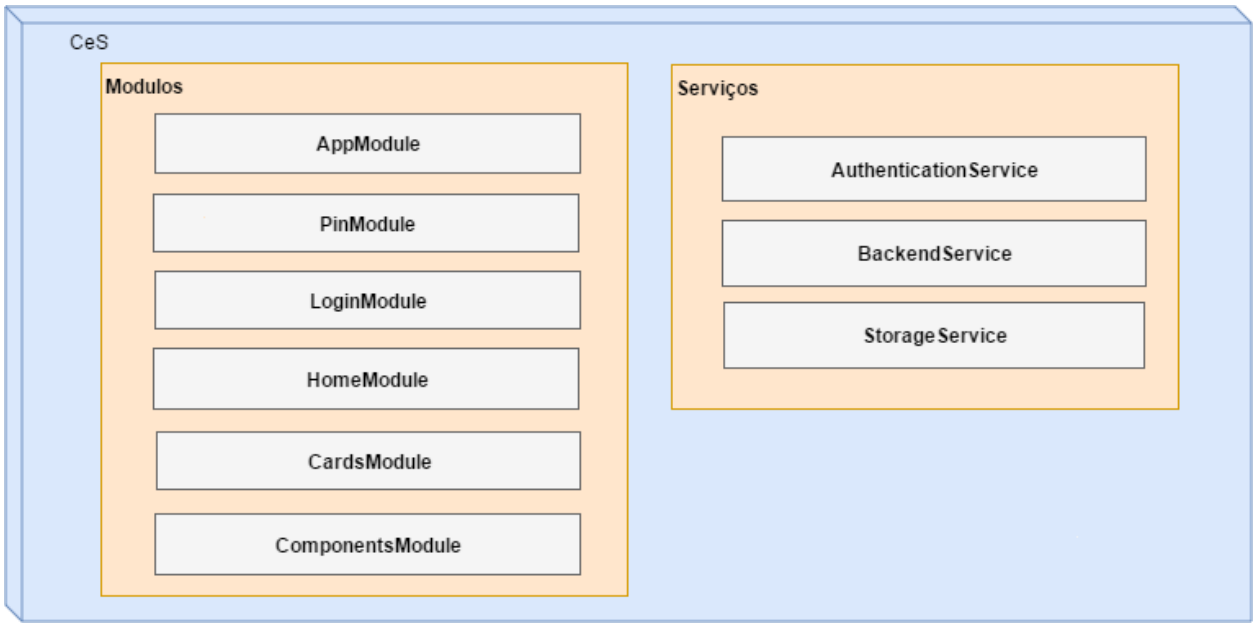
11. Documentação

- European Commission - Case studies on open source software
- OSS Watch - Is Open Source Software Insecure? An Introduction To The Issues
- Hierarchy and centralization in free and open source software team communications. . Howison J. , Crowston, K.
- ESOP - Associação de Empresas de Software Open Source Portuguesas
- UK Government - OSS Toolkit Security Note
- Open Source Initiative
- Free Software Foudation
- Producing OSS - How to Run a Successful Free Software Project

Arquitetura da Aplicação



A aplicação CeS é desenvolvida em Angular 2, pelo que apresenta a arquitectura demonstrada no diagrama anterior.
De o modo geral, é constituída por 2 componentes gerais: **Módulos** e **Serviços**.



Módulos

Entre os módulos, cada um destes tem a sua responsabilidade, facilitando assim uma manutenção mais simples.

AppModule - Cada aplicativo tem pelo menos um módulo Angular, o módulo raiz (*root*) que inicia o aplicativo. Cartões de Introdução e Error (páginas de erro),

PinModule - Módulo para registo do PIN da Carteira, autenticação e mudança de PIN, com ligação ao serviço encriptado de Autenticação. O PIN fica guardado apenas no armazenamento do dispositivo móvel (Storage Service).

LoginModule - componente de autenticação na aplicação. Mediante o serviço de autenticação escolhido (Chave Móvel Digital ou Registo Nacional de Utente)

CardsModule - Conteúdo dos cartões (view de Cartões da CeS em detalhe)

HomeModule - *view* de Cartões da CeS (vista fechada), Menus (Settings e About), CardsList, adicionar, visualizar e remover cartões.

ComponentsModule - componentes de interface (navigation drawer, scroll, etc)

Serviços

AuthenticationService - Autenticação e ligação aos servidores (Backend) de forma segura e encriptada.

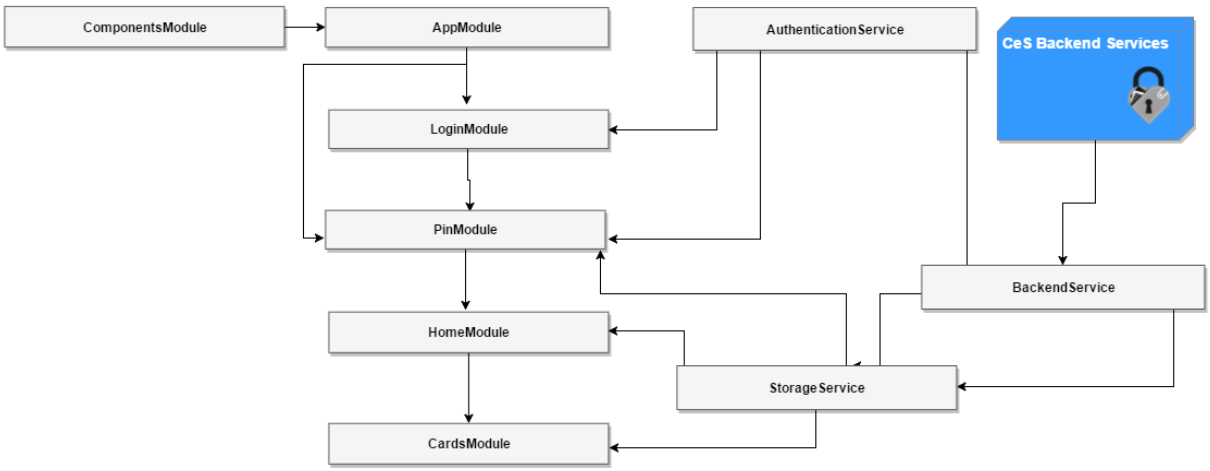
BackendService - ligação às base de dados dos vários sistemas de informação (centrais e locais) e o Servidor CeS (Azure Container Service -Docker Swarm) é feita usando o protocolo HTTPS (ver diagrama 1)

Toda a informação trocada é ainda encriptada e assinada digitalmente usando os algoritmos de encriptação RSA (2048 bit) e AES (256 bit). O formato de dados utilizado é JOSE - JSON Object Signing and Encryption. O par de chaves RSA do servidor está alojado no Azure Key Vault, sendo as operações de assinatura e desencriptação efetuadas lá.

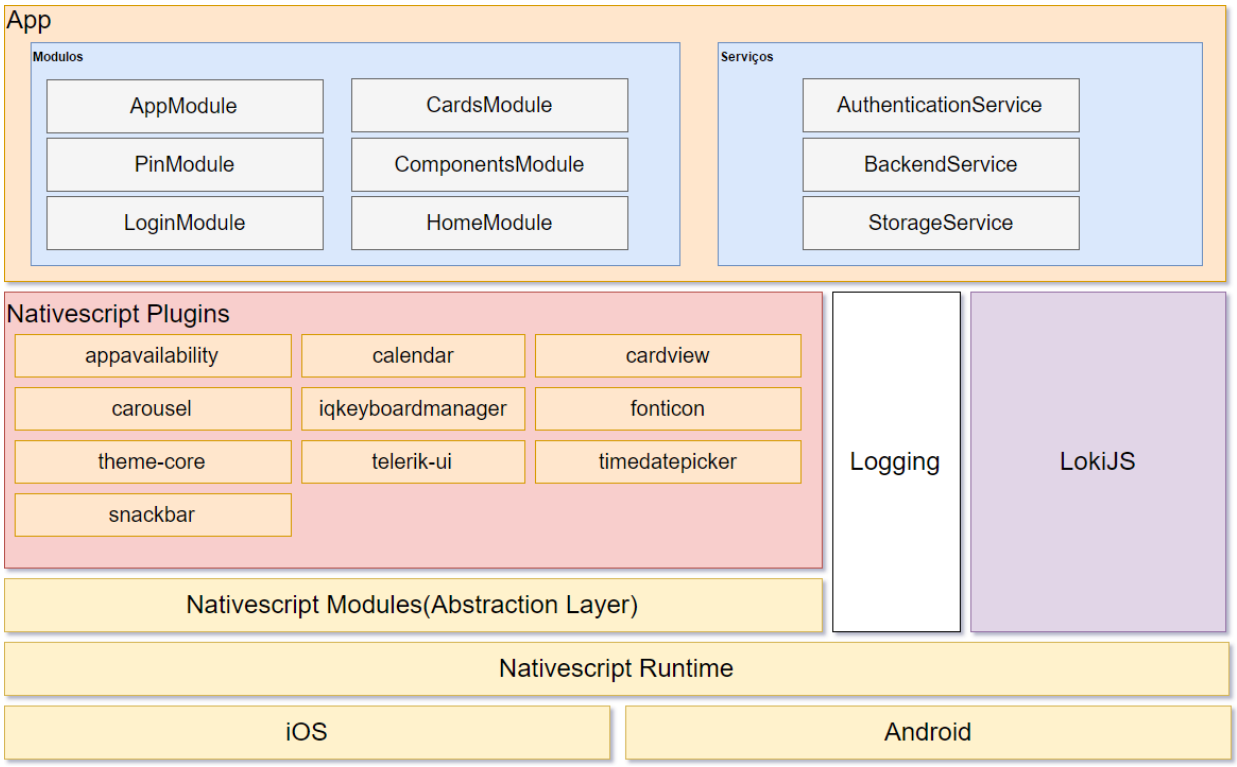
Storage Service - Serviço de armazenamento de dados no dispositivo móvel, devidamente encriptado. Através deste serviço, é armazenado o PIN, Home View e os cartões.

CeS Backend Services - conjunto de serviços de apoio ao funcionamento da CeS (autenticação, comunicação, telemetria.)

Comunicação



Arquitetura NativeScript (simplificada)



De momento já são utilizados mais plugins do que os mencionados, pelo que esta se trata de uma arquitectura simplificada.

Plugins

appavailability - para verificar a disponibilidade de outros aplicativos no dispositivo

calendar - permite manipular eventos no calendário nativo do dispositivo, permitindo apagar, criar, eliminar e encontrar eventos. Na CeS, é utilizado para adicionar eventos de levantamento de prescrições da eGuia de Tratamento.

cardview - fornece um widget XML para implementar o componenteMaterial Design CardView

carousel - efeito carousel, utilizado nos ecrãs informativos iniciais

iqkeyboardmanager - wrapper para iOS que impede a a sopreposição do teclado nos controlos UITextView
fonticon - font icons com NativeScript
theme-core - core NativeScript theme
telerik-ui - Telerik UI
timedatepicker - selectores de data e hora para Android e iOS
snackbar - Material Design Snackbar para Android e SSSnackbar Cocoapod para iOS

health data for nativescript – plugin desenvolvido internamente para aceder ao Apple HealthKit e Google Fit API.

Segurança Fisica e Lógica

Medidas de segurança física

A informação do cidadão ficará residente nas bases de dados locais e centrais notificadas junto da CNPD em processos próprios.

Toda a informação adicional, ficará guardada localmente de forma encriptada (por AES 256), no dispositivo móvel do cidadão. A meta-informação do Servidor CeS será guardada usando algoritmos de criptografia RSA 2048 (armazenados em HSM – hardware security module - FIPS 140-2 Level 2) cujos requisitos de segurança e necessidades de escalabilidade de produto são assegurados pela opção de cloud Azure Key Vault, que contempla níveis de serviço (SLA) não só a nível de disponibilidade como a nível de cibersegurança.

A solução Azure tem inúmeros requisitos de segurança que, para serem implementados num datacenter SPMS implicavam um custo de dimensões não suportáveis nem justificáveis, considerando a criticidade da informação aí alojada (recordamos que a informação de saúde do Cidadão, não fica residente nesta base de dados, apenas nos dispositivos móveis encriptada e nas bases de dados centrais e locais já existentes e notificadas).

Medidas de segurança lógica

A comunicação realizada entre o dispositivo móvel e o servidor CeS usa o protocolo HTTPS. A comunicação realizada entre a base de dados dos vários sistemas de informação (centrais e locais) e o Servidor CeS usa o protocolo HTTPS e toda a informação trocada é encriptada e assinada digitalmente usando os algoritmos de encriptação RSA (2048 bit) e AES (256 bit). Para a encriptação usando o algoritmo RSA, o par de chaves de encriptação é do tipo HSM (hardware security module - FIPS 140-2 Level 2), cujos requisitos de segurança e necessidades de escalabilidade de produto, levaram a uma opção de cloud Azure Key Vault. O Servidor CeS não guarda qualquer tipo de informação relativa aos cartões do Cidadão, mas guarda informação de registo (logs encriptados) para diagnóstico de problemas de funcionamento da aplicação, com tempo de conservação máximo de 1 mês.

Funcionamento

Recolha de Dados

Para ativação da MySNS Carteira o cidadão regista na aplicação o Número de SNS + Data de Nascimento + Número de Telemóvel.

A informação do cidadão, quer relativa a dados pessoais, quer a dados de saúde, disponibilizada através de cartões, está residente nas bases de dados locais e centrais notificadas junto da CNPD em processos próprios.

Toda a restante informação fica guardada **localmente** de forma encriptada, no dispositivo móvel do Cidadão.

Comunicação e Interconexões de Dados

A MySNS Carteira poderá receber informação das várias bases de dados de informação de saúde, devidamente notificadas junto da CNPD - Comissão Nacional de Protecção de Dados em processos próprios.

A comunicação feita entre o dispositivo móvel e o servidor MySNS Carteira é feita usando o protocolo **HTTPS**.

A comunicação feita entre a base de dados dos vários sistemas de informação (centrais e locais) e o Servidor MySNS Carteira é feita usando o protocolo HTTPS e toda a informação trocada é ainda encriptada e assinada digitalmente usando os algoritmos de encriptação **RSA (2048 bit)** e **AES (256 bit)**.

Para a encriptação usando o algoritmo RSA, o par de chaves de encriptação é do tipo HSM (hardware security module - FIPS 140-2 Level 2), cujos requisitos de segurança e necessidades de escalabilidade de produto, levaram a uma opção de cloud Azure Key Vault.

O Servidor MySNS Carteira não guarda qualquer tipo de informação relativa aos cartões do cidadão, guarda apenas informação de registo (logs encriptados) para diagnóstico de problemas de funcionamento da aplicação, com tempo de conservação máximo de 1 mês e está alojado na *cloud*.

A informação será guardada localmente no dispositivo móvel do Cidadão (encriptada) cabendo a este, de forma discricionária, apagá-la quando achar conveniente. Relativamente ao servidor MySNS Carteira, conforme já referido no ponto de comunicação e interconexões de dados, é apenas guardada meta informação (enquanto o cidadão tiver a MySNS Carteira ativa) e logs encriptados para deteção de problemas, no **prazo máximo de um mês**.

O direito de acesso aos dados recolhidos no âmbito da aplicação MySNS Carteira é efetuado na SPMS – Serviços Partilhados do Ministério do Saúde, E.P.E. O acesso aos dados dos cartões disponibilizados através da MySNS Carteira

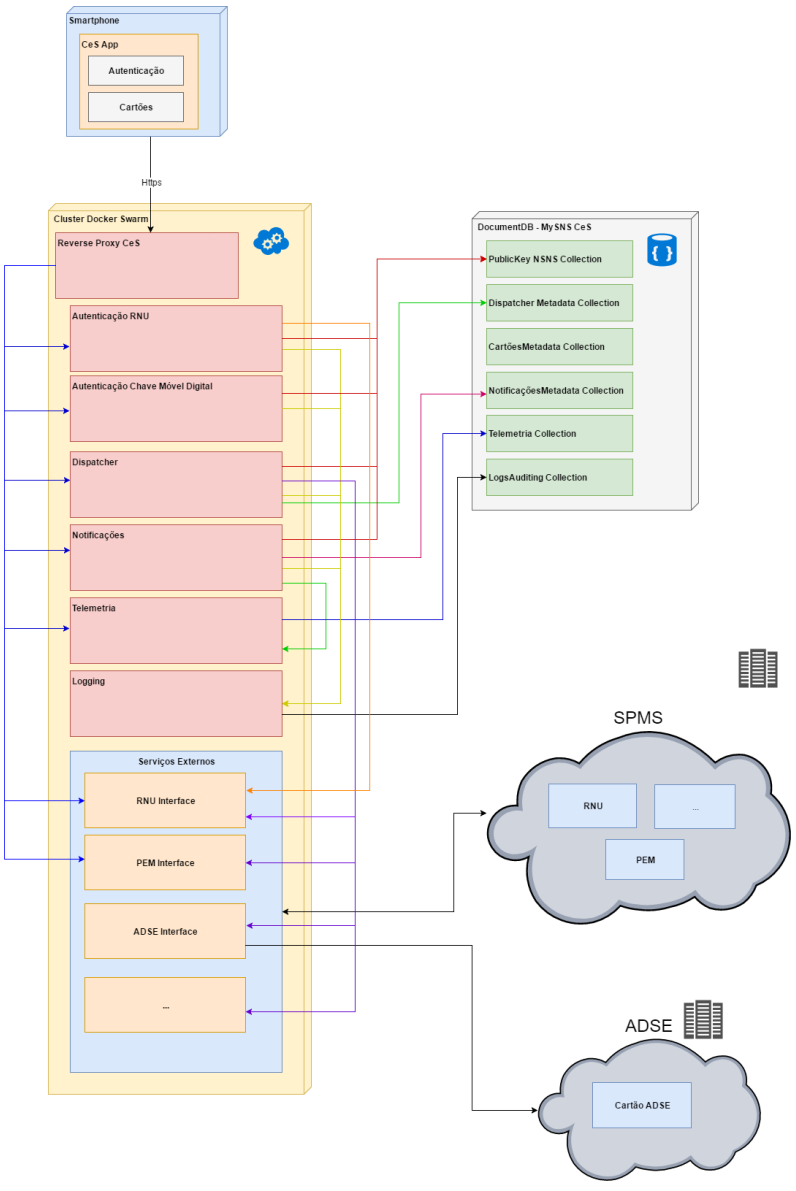
continua a ser exercido junto da instituição de saúde onde o titular dos dados está inscrito, ou no caso de dados da saúde, por intermédio de médico escolhido pelo titular dos dados.

Backend - Serviços

Arquitectura

A arquitectura da aplicação está em contante evolução, pelo que esta documentação será atualizada sempre que possível, podendo haver algum atraso em relação à versão atual da aplicação.

Nesta aplicação utilizou-se o **Docker Swarm** (ver a secção Serviços para mais informações) no **Azure Container Service**. O Docker Swarm, utilizando a API do Docker nativa, fornece um ambiente para a implementação de cargas de trabalho de conteúdo através de um conjunto agrupado de anfitriões de Docker.



No diagrama acima, é possivel identificar duas secções no Container Service: **Core** e **Integrações**
A aplicação comunica diretamente com o Reverse Proxy CeS. Cada serviço é executado num contentor Docker. Todos os contentores Docker são independentes (possuindo cada um o seu Dockerfile), pelo que a inoperabilidade de um não afeta os restantes.
A orquestração é feita segundo um ficheiro de configuração – `docker-compose.yaml`.
Contentores Docker:

- **Autenticação RNU** - responsável pela validação dos dados. Se forem validos, responde OK e envia uma sms com codigo TOTP pelo serviço da SPMS e faz um novo reply com o codigo TOTP (2-step authentication).
- **Autenticação CMD** - redirecciona o cidadão para a pagina Autenticação.GOV. Depois de o cidadão se autenticar, é gerada uma chave.
- **Dispatcher** - toda a comunicação da CeS, excepto Telemetria e Autenticação. No futuro, irá fazer a avaliação de Requests e Replies.
- **Notificações** -
- **Telemetria** - dados estatísticos (quantas pessoas iniciaram sessão, descarregaram cartões, etc)
- **Logging** - auditorias e logs

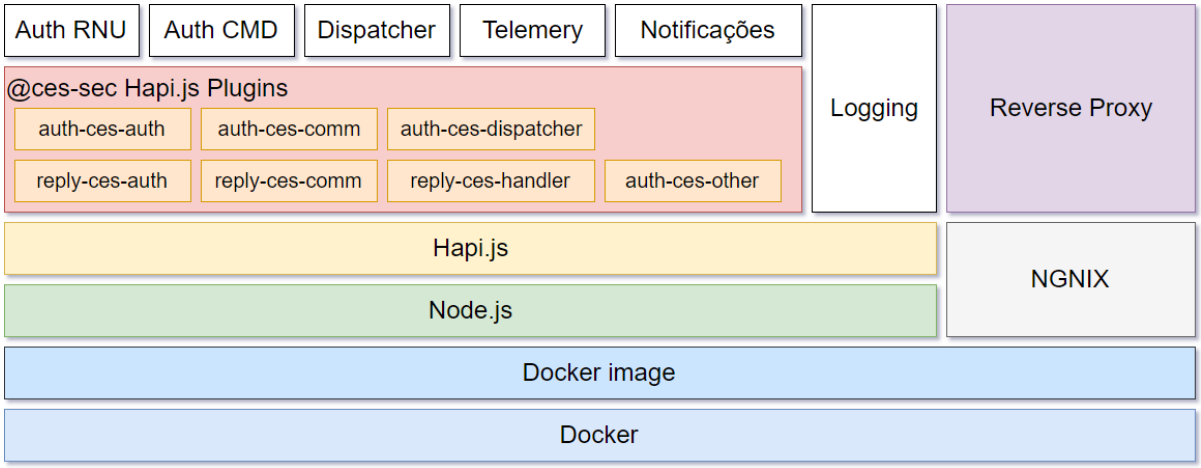
As integrações são os **serviços externos**, que facilitam o acesso aos servidores SPMS e ADSE. Para tal, criou-se uma interface para cada serviço acedido (RNU, PEM, ADSE). Cada serviço é executado num contentor Docker próprio e isolado. É criada uma rede virtual no Docker para isolar os contentores de integração dos de serviços Core.

No **Core** do Azure Container Service, encontram-se os serviços que comunicam com a base de dados **DocumentDB**, responsável pelo armazenamento de *logs* encriptados para diagnóstico de problemas de funcionamento da aplicação, telemetria, associação (encriptada) do dispositivo ao cidadão e ainda metadatas de cartões, notificações e dispatcher. Estes dados não são armazenados no dispositivo móvel, mas sim da base de dados de backend (DocumentDB).

Dados armazenados na DocumentDB

- **PublicKeys NSNS Collection** -
- **Dispatcher Metadate Collection** -
- **CartõesMetadata Collection** - Apenas guarda dados de *Look and Feel* - cor, logo etc, não armazenando quaisquer dados pessoais
- **NotificaçõesMetadata Collection**
- **Telemetria Collection** - dados estatísticos
- **LogAudition Collection** - auditorias e logs

Serviços e Tecnologias (simplificado)



Operations support systems (OSS)

- Node.js
- Hapi.js
- PM2
- Docker
- DocumentDB / MongoDB
- NPM modules

Node.js

É um runtime de Javascript Server-side. Permite trabalhar na mesma linguagem e ambiente no frontend e backend (equipas mais compactas).
É amplamente suportado por muitas empresas comerciais e Node.js Foundation (Linux Foundation), tendo a maior comunidade no GitHub
É utilizado por: PayPal, Netflix, LinkedIn, Uber, IBM

- mais informações
- Github

Hapi.Js

Web and services application framework - framework de Node.js para desenvolver serviços baseado na configuração. Criado no Wallmart Labs para responder às necessidades do Wallmart.
É utilizado por: PayPal, Disney, GOV.UK, mozzila, Yahoo, etc

- mais informações
- github - Hapi.Js + TypeScript

- Biblioteca Joi - Object schema description language and validator for JavaScript objects.

  Plugins Hapi.js criados:

  Autenticação

- auth-ces-auth
- auth-ces-comm
- auth-ces-dispatcher
- auth-ces-other

  Resposta

- reply-ces-auth
- reply-ces-comm
- reply-ces-handler

  ## *PM2*

  PM2 é um Process Manager de Node.js para aplicações em produção, com balanceador de carga incorporado.
  PM2 é um gerenciador de processos de produção para aplicativos Node.js com load balancer incorporado. Facilita as tarefas comuns do administrador do sistema, tais como: criar Cluster aplicacional, Workflows de deployment, Gestão de Logs e Monitorização
  É utilizado por: PayPal, Microsoft, IBM, etc

- mais informações
- Documentação
- Github

  ## *Docker*

  Docker é um projeto open-source que permite criar um contentor (*container*) com a infraestrutura que se pretender por forma a poder partilha-la em qualquer máquina e manter sempre o mesmo comportamento. Este contentor contem um sistema operativo Linux lightweight, com o tradicional sistema de ficheiros e politicas de segurança que caracterizam o sistema operativo Linux, utilizando assim menos recursos de memória e espaço em disco. Em suma, permite compartimentalizar serviços e aplicações em qualquer tecnologia e executar em qualquer ambiente. É suportado pelos maiores fornecedores cloud (Amazon, Microsoft, Google).

  É utilizado por: PayPal, BBC, eBay, GE, Spotify, ING, etc

- mais informações
- Documentação
- Github

DocumentDB / MongoDB

MongoDB
Base de Dados NoSQL open-source e cross-platform, sem esquemas (*schemaless*), orientado a documentos. Tem um bom suporte de JavaScript e uma grandde comunidade.
A informação é guardada no formato JSON.
É utilizado por: Forbes, BOSCH, Facebook, GOV.UK, McAfee, etc.

- Documentação

Azure DocumentDB
Base de Dados semelhante ao MongoDB, mas gerida na Cloud. Pertence à Microsoft Azure platform.

- Documentação

NPM modules

Node Package Manager. Todos os módulos criados para as aplicações MySNS podem ser consultados aqui:

- MySNS - Available Packages

Segurança - Bibliotecas e Tools

De modo a facilitar o entendimento da CeS, optou-se por descrever algumas bibliotecas e ferramentas.

Azure Key Vault:

O cofre de chave do Azure ajuda a salvaguardar as chaves criptográficas e os segredos utilizados pelas aplicações em nuvem e pelos serviços.

- mais informações
- npm
- API docs

(RSA) jsrsasign

'jsrsasign' (RSA-Sign JavaScript Library) é uma biblioteca criptográfica opensouce que suporta RSA/RSAPSS/ECDSA/DSA signing/validation, ASN.1, PKCS#1/5/8 private/public key, X.509 certificate, CRL, CMS SignedData, TimeStamp & CAdES e JSON Web Signature(JWS)/Token(JWT)/Key(JWK)

- site
- npm
- API docs

- certificados CA:

  - JS Certification Authority

Crypt AES - Crypto-js

Crypto-js é uma biblioteca JavaScript de padrões de criptografia.

- github
- npm

Validação de JSON

- site
- npm
- conversão de tipos typescript e JSON Schema - github

TOTP

Para a gestão de passwords, utilizou-se o One-Time Password manager. Compativel com HOTP (counter based one time passwords) e TOTP (time based one time passwords). Suporta autenticação de dois fatores.

- npm

JSON Web Key (JWK) - para Storage

- Docs

JSON Web Signature (JWS) - Payload JWE

- Docs

JSON Web Encryption (JWE) - Payload MessageAction(CeS)

- Docs

NativeScript + Angular

Esta aplicação utiliza **NativeScript com TypeScript + Angular**

NativeScript é uma estrutura livre e de código aberto para a construção de aplicações em iOS e Android usando JavaScript e CSS. O NativeScript processa interfaces de utilizador com o mecanismo de renderização da plataforma nativa - sem WebViews - resultando em desempenho nativo e UX.

Juntamente com Angular 2, o resultado é uma arquitetura de software que permite criar aplicativos para dispositivos móveis usando a mesma estrutura - e em alguns casos o mesmo código.

Documentação auxiliar:

Tutorial + Documentação - TypeScript

Tutorial - NativeScript com TypeScript + Angular.

Angular 2 - Style Guide

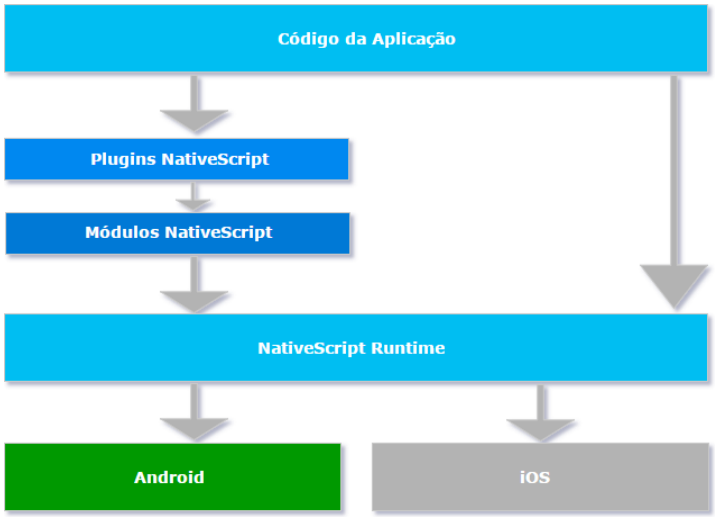
Arquitectura de NativeScript

Para permitir o acesso a recursos de plataforma e dispositivos nativos da plataforma de destino (iOS/Android), NativeScript utiliza um design modular - todas as funcionalidades de dispositivos, plataformas ou interfaces de usuário residem em módulos separados. Os Módulos expõem os recursos da plataforma nativa do dispositivo de forma consistente.

Um aplicativo NativeScript típico usará o conjunto de módulos que adicionam camada de abstração (*abstraction layer*) entre plataformas sobre as APIs específicas do SO.

Durante a compilação, o runtime do NativeScript traduz código não específico da plataforma para o idioma nativo da plataforma de destino. As ferramentas NativeScript usam os SDKs e ferramentas da plataforma nativa para criar um pacote de aplicativo nativo

O seguinte diagrama de blocos descreve o funcionamento da estrutura de uma app em NativeScript.



Colabore connosco

Para esclarecer dúvidas e/ou caso pretenda colaborar connosco é favor enviar-nos email para servicedesk@spms.min-saude.pt