|  |  |  |  |
| --- | --- | --- | --- |
| **Documento de Solução Arquitetural** | | | |
| Nome do Projeto: | | | |
| Área Gestora |  | Consultor CGPP |  |
| Gerente de Projeto |  | Data da Solicitação |  |

|  |
| --- |
| Arquitetura da Aplicação |
| Este documento apresenta uma visão geral e estruturada a respeito da arquitetura dos aplicativos a serem desenvolvidos para o Ministério da Educação. Além dessa visão geral, apresenta também particularidades e soluções que foram adotadas para o projeto SISU.  Os aplicativos serão desenvolvidos tendo como base a arquitetura multicamadas aderente à arquitetura orientada a serviço utilizando REST (*Representational State Transfer*), *Cordova* e *Sencha Touch* na camada de apresentação.  Macintosh HD:Users:marioeugenio:Downloads:5.png   1. **Objetivos e Restrições da Arquitetura**   A arquitetura proposta tem como objetivo disponibilizar um aplicativo mobile acessível e escalável, utilizando camadas separadas fisicamente e aderentes à arquitetura REST, expondo funcionalidades de negócios ou aplicação.  Os critérios utilizados para a seleção da solução arquitetural foram:   * Escalabilidade * Manutenibilidade * Segurança * Acessibilidade e Usabilidade * Integração  1. **Pré-requisitos e Restrições de Ambiente**   Softwares Utilizados  Para execução do aplicativo será necessária os seguintes softwares:   * Android 2.3 ou superior; * IOS 6 ou superior; * Windows Phone 8;   Disponibilidade  Para garantir uma alta disponibilidade os sistemas devem segmentar suas funcionalidades de forma a aplicar uma distribuição para execução das suas aplicações separadas, que será implementada pela AKAMAI.   * Funcionalidade que envolve grande quantidade de acesso em uma aplicação distinta, isso evitará a sobrecarga de acesso. (Este item será atendido quando o aplicativo também suportar tal intensidade de acesso) * AKAMAI realiza a disponibilização da nota de corte: Após a geração do lote (MEC) de arquivos (.json) com as nota de corte separado por oferta, os mesmos serão disponibilizados no servidor AKAMAI para consumo do aplicativo mobile. Regras gerais do arquivo:   1 - Disponibilizado na pasta /sisu/mobile/notas/  2 - Extensão e nomenclatura do arquivo <identicador da oferta>.json  3 - Estrutura de dados do arquivo  {  “notacorte\_ampla”: “400,00”,  “datahora”: “24/06/2015 08:00”  }   * AKAMAI realiza a disponibilização das futuras atualizações de banco em uma pasta. O aplicativo realizará consultas e aplicar as atualizações necessárias no banco. Regras gerais do arquivo.   1 - Disponibilizado na pasta /sisu/mobile/dados/  2 - O Arquivo não possui extensão, a nomenclatura deve ser incremental com valor inicial 1.  3 - Estutura de dados são comandos sql INSERT, UPDATE e DELETE.   * Também há validação prévia das mudanças de banco e valores das notas de corte em uma pasta para homologação onde aplicativo aponta para está para checagem pré-produção.   Pastas  /hmg/sisu/mobile/notas/ e /hmg/sisu/mobile/dados/   * Segurança: O acesso para consumo das notas de corte se dará por meio de um token dinâmico com tempo de expiração de 1 dia.   Escalabilidade   * Os sistemas devem ser construídos de forma a manter uma escalabilidade vertical, principalmente para aplicativos que possua um grande volume de acesso.   Manutenibilidade   * Os components e plugins devem ser construídos de forma que possam ser reutilizados, e descentralizar os componentes. * Manter o padrão nomenclatura e de classes por este documento mencionado   Segurança  Para garantia de segurança do sistema que transacione informação confidenciais, foi utilizado um token dinâmico, segundo especificações técnicas fornecidas pela equipe tecnologia do MEC.   1. **Critérios da Avaliação Arquitetural**   Foi utilizado como critério as seguintes primícias para o desenvolvimento de aplicativos mobile:   * Aplicativos híbrido * Framework *Sencha Touch* versão 2.4.1 * Cordova versão 4.3.0 * HTML 5 |

|  |
| --- |
| Visão lógica |
| A fim de atender às camadas estabelecidas, a organização do código divide-se principalmente em 2 conjuntos principais:   * Aplicação - contém os códigos referentes às implementações específicas das regras de negócio da aplicação em questão, e encontram-se na pasta app localizada na raiz do código da aplicação. * Bibliotecas - são as bibliotecas de apoio que compõem o framework e permitem a implementação das diversas regras da aplicação. Encontram-se na pasta touch localizada na raiz do código da aplicação.  1. **Estrutura de trabalho (distribuição física dos arquivos)**   O framework em uso estabelece uma estrutura fixa e padronizada para sua utilização, seguindo o padrão do Sencha Touch. Praticamente todo o trabalho é realizado na pasta app localizada na raiz do projeto. Dentro desta pasta encontramos a seguinte estrutura hierárquica para os arquivos, também representada de forma simplificada (sem arquivos) no gráfico abaixo:   * **/**   + **resources** - Camada responsável por armazenar arquivos "extras" que será utilizado eventualmente no aplicativo. Por exemplo, um CSS personalizado para os componentes visuais de seu aplicativo, um arquivo JavaScript que contém a lógica de negócios para a aplicação.     - css - camada de armazenamento de arquivos css personalizados para o aplicativo ou determinado componente.     - **data - camada responsável por armazenar arquivos de carga para mock de determinados registros que serão utilizados na aplicação.**     - icons - camada de armazenamento dos ícones que serão utilizados pelo aplicativo.     - img - camada de armazenamento de imagens eventualmente utilizados no aplicativo.     - **js - camada responsável pelo armazenamento de bibliotecas externas que serão utilizados no aplicativos, no contexto SISU MOBILE foi utilizado o D3 para geração de gráfico de Radial Progress, e Marker Cluster para utilização de geolocalização.**     - **loading - camada responsável pelo armazenamento das imagens de loading do aplicativo.**     - locale - camada responsável pelo armazenamento dos arquivos de locale de tradução dos componentes utilizados no aplicativo.     - sass - camada responsável pelo armazenamento dos arquivos de SASS.     - siesta - camada de armazenamento do framework de teste SIESTA.     - startup - camada responsável pelo armazenamento dos arquivos de splash screen.   + **touch** - Armazena toda estrutura de plug-ins default do Framework Sencha Touch.   + **cordova** - Estrutura responsável pelo empacotamento do aplicativo, CORDOVA/PLATFORMS armazena os projetos por plataforma Ex.: Android, IOS, Windows Phone ...), CORDOVA/PLUGINS tem a responsabilidade de armazenar todos os plug-ins cordova para integração com o dispositivo.   + **tests** - Camada responsável pelo armazenamento de todos os scripts de teste, desde testes unitários, integração e testes de UX/UI. * **App/**   + **controller** - A Camada de Controle, também chamada de Controller, é responsável por receber os parâmetros enviados na requisição do usuário;   + **model** - Esta camada, também chamada de Model, é um superconjunto de camadas as quais visam lidar com diversos aspectos estruturais (Identity Field, Concrete Table Inheritance), comportamentais (Unit of Work, Identity Map), de mapeamento (Repositoty, Metadata Mapping) e fonte de dados (Data Mapper, Active Record).   + **util** - A função dessa camada é incluir na mesma todos os componentes customizados para a aplicação em questão, desde componentes de paginação para listas até animações customizadas.   + **store** - A função dessa camada é manter todos os Stores da aplicação móvel. Entende-se como Store, as classes que encapsulam o armazenamento e acesso a dados no cliente (aplicação móvel). Assemelha-se a um DAO (Data Access Object), porém podendo acessar dados de diversos tipos de repositórios, como: remotos (web service) ou locais (no dispositivo).   + **view** - A função da camada é manter os arquivos responsáveis pela camada de visualização; |

|  |
| --- |
| Visão de Implantação |
| A implantação de aplicativos só será finalizada a partir da publicação do mesmo em suas respectivas lojas, conforme a plataforma acordada (Google Play, Itunes Connect, Windows Phone Store).  A entrega e implantação de aplicativos mobile deveram seguir as seguintes especificações, segundo a plataforma.   1. **ANDROID - Google Play**  * Acessar Google Play Console - https://play.google.com/apps/publish/signup/ * Definir Usuário e Senha credenciados a publicação APP * Clicar em Adicionar novo APP * Fazer upload de novo APP, assinado com certificado (O certificado poderá ser criado em servidor local ou certificado disponibilizado por unidade certificadora do MEC) * Preencher os dados obrigatórios conforme solicitado na publicação do aplicativo  1. **IOS - Itunes Connect**  * Acessar Itunes Connect - https://itunesconnect.apple.com/ * Definir Usuário e Senha credenciados a publicação APP * Clicar em Adicionar novo APP * Preencher os dados obrigatórios conforme solicitado na publicação do aplicativo * Utilizando a IDE Xcode, o aplicativo é publicado na loja. * Após revisão o mesmo fica disponível para ser baixado.  1. **Windows Phone**  * Acessar - http://www.windowsphone.com/pt-br/store/ * Definir Usuário e Senha credenciados a publicação APP * Definir Usuário e Senha credenciados a publicação APP * Fazer upload de novo APP * Após revisão o mesmo fica disponível para ser baixado.   Macintosh HD:Users:marioeugenio:Downloads:1.png |

|  |
| --- |
| Visão de Implementação |
| Os aplicativos são distribuídos da seguinte forma:   * Servidores Web * Servidores de Aplicação;     **Estrutura Lógica do funcionamento dos hardwares**  **Segurança**  A aplicação deverá estar organizada de forma a conseguir separar fisicamente a interface das camadas de Negócio, Serviço, e Persistência. Buscando assim, dar uma maior segurança a aplicação.  Quanto há uso de certificado digital, as aplicações devem ser alocadas em uma área restrita, diversa a das aplicações comuns.  **Escalabilidade**  A aplicação deverá estar preparada de forma a utilizar múltiplos processamentos, possibilitando a distribuição em máquinas distintas para realizar o processamento.  **Manutenibilidade**  A aplicação deverá estar nas melhores práticas de implementação para que as manutenções sejam feitas de forma breve, e também se adequando e suportando novos ambientes. Visão de Implementação.   1. **Visão Geral**    1. **View**   A Camada de Visualização, ou simplesmente View, é comumente a camada responsável por 2 itens principais: a apresentação de um modelo fornecido pela Controller, e fornecer meios de interação com a modelo para sua criação, alteração, exclusão ou apenas visualização. A View é também responsável por ser a interface de apresentação de dados à outras aplicações em formatos pré-definidos de intercâmbio, mais comumente o XML e JSON.  Os papéis atribuídos à camada View são:   * Exibir informações para usuário, tanto em telas interativas, quanto de forma estática através de relatórios para impressão; * Filtros de saída e Escape em variáveis a fim garantir a correta apresentação de elementos para o HTML5; * Disponibilizar os dados de um modelo fornecido pela Controller em distintos formatos de saída, permitindo o provimentos de informações entre sistemas, ou seja, realizando a conversão dos dados através de helpers ou filtros de saída, para o formato desejado (definição de cabeçalhos de mime-type, conversão de objeto em objeto json, e outros ainda a serem definidos).   Esta camada será implementada utilizando a biblioteca Sencha Touch (em conformidade com as determinações da Arquitetura da Informação).  A inclusão de plugins para esta camada será realizada sempre mediante estudo de usabilidade do mesmo, e será homologado pela arquitetura da MBA Mobi através de critérios técnicos relativos à qualidade do plugin e conformidade com os requisitos definidos.   * 1. **Controller**   A Camada de Controle, também chamada de Controller, é responsável por receber os parâmetros enviados na requisição do usuário, direcionar ela aos serviços correspondentes e designar a View necessária para apresentar a resposta.  Os papéis atribuídos a camada Controller são:   * Receber e tratar a requisição para repassar os dados à camada serviço; * Delegar o processamento da requisição para um ou mais serviços; * Delegar a exibição da resposta para uma View.   1. **Model**   Esta camada, também chamada de Model, é um superconjunto de camadas as quais visam lidar com diversos aspectos estruturais (Identity Field, Concrete Table Inheritance), comportamentais (Unit of Work, Identity Map), de mapeamento (Repositoty, Metadata Mapping) e fonte de dados (Data Mapper, Active Record).  Os papeis atribuídos a camada Model são:   * Organização da lógica de domínio da aplicação; * Disponibilização de dados de diversas fontes encapsulados em objetos para processamento pelas regras do negócio, conforme a necessidade do negócio e a fonte dos dados;   Toda a utilização da Model será realizado através de gerenciadores de entidades, responsáveis por administrar a existência das mesmas e criar a ponte entre as mesmas e suas fontes de persistência. O proxy ou persistência utilizado no aplicativo foi Sqlite, a estrutura do banco está no padrão ORM com as seguintes estruturas:  Macintosh HD:Users:marioeugenio:Downloads:8888OS_03_01.jpg   1. **Padrão de Codificação**   Implementação da codificação deverá seguir os seguintes parâmetros:   * Boas práticas checkstyle, conforme definido em convenção. * Padrão de nomenclatura para métodos e classes Camelcase. * Comentários sintéticos e descritivos conforme a regra implementada em cada método. * Nomenclatura de pacotes e arquivos conforme o domínio. * Plugins e componentes descentralizados para reutilização. |