**Desafio Técnico**

O Objetivo desta avaliação é testar seus conhecimentos para resolver os principais desafios encontrados no dia-a-dia de um desenvolvedor Android. O tema proposto é Pagamento Móvel e você pode se sentir a vontade para explorar a sua criatividade em cima do tema proposto, sendo assim você pode fazer um App de qualquer tipo de pagamento, por exemplo, relógios, revistas, livros, etc. A idéia é criar uma loja virtual onde (1) o usuário deverá se cadastrar e (2) logar e usar a App deve usar os serviços REST definidos abaixo para fazer o cadastro e login e logout do usuário. Sua App deverá também, consumir um serviço REST e possuir os seguintes requisitos:

* A aplicação deve fazer busca, inserção, alteração e exclusão dos pagamentos.
* Utilizar Android Background Service
* Content Provider
* Você tem que usar o Android Studio
* Salvar dados usando SharedPreferences e também salvar dados no SQLite utilizando algum ORM de sua preferência
* Usar Fragments para criar telas dinâmicas
* Suportar diferentes tamanhos e resoluções de tela
* Orientação de tela Portrait e Landscape em múltiplos dispositivos e tamanhos de tela.

Somos uma equipe muito focada em sempre entregar o melhor para os nossos clientes, portanto os **TESTES SÃO ESSENCIAIS.** Quanto mais, melhor! Testes unitários, TDD, Testes de interface e qualquer outro teste que você achar adequado, são bem vindos.

A experiência do usuário ao usar nossa aplicação tem que ser a melhor possível, portanto explore os patterns de layout do Android, tente seguir o Guideline da Google e o novo Material Design e faça a aplicação mais bonita possível.

**Ganhe pontos extras**

* Usar Orientação à Objetos e Design Patterns
* Uso de Mock da API de pagamentos
* Gestão de dependências no projeto.
* Mapeamento Json -> Objeto
* Framework para Comunicação com API.
* Testes no projeto.
* Testes funcionais.
* Usar Git e Git Flow (por favor quando nós enviar o repositório queremos ver você usando branches de features e como você estrutura a implementação em pequenos commits).

**Submissão do código-fonte**

Por favor é muito importante que você não submeta o código-fonte em nenhum repositório público. Precisamos que você compacte o repositório (incluindo do diretório .git) e envie para o rh@m4u.com.br.

Boa sorte!

**API Restful**

Para o desafio, você vai necessitar desenvolver para uma API Restful. Para isso queremos que você crie os mocks do servidor simulando que a equipe de Backend ainda está desenvolvendo o backend, mas você precisa iniciar o desenvolvimento da App. Para ter mais detalhes da Api acesse: <http://editor.swagger.io/#/> e cole o seguinte documento Swagger Yaml e visualize como funciona a Api.

**Swagger Yaml**

swagger: "2.0"

info:

 description: |

   This a sample API to be used in M4U Android Recruitment App

 version: "1.0.0"

 title: M4U Android Recruitment App

 termsOfService: http://www.m4u.com.br/terms/

 contact:

   name: rh@m4u.com.br

 license:

   name: Apache 2.0

   url: http://www.apache.org/licenses/LICENSE-2.0.html

host: recruitment.m4u.com.br

basePath: /v1

schemes:

 - http

paths:

 /payment:

   post:

     tags:

       - payment

     summary: Submits a new Payment Request

     description: ""

     operationId: submitPaymentRequest

     consumes:

       - application/json

       - application/xml

     produces:

       - application/json

       - application/xml

     parameters:

       - in: body

         name: body

         description: Payment Request

         required: false

         schema:

           $ref: "#/definitions/PaymentRequest"

     responses:

       "405":

         description: Invalid input

   put:

     tags:

       - payment

     summary: Update an existing Payment Request

     description: ""

     operationId: updatePaymentRequest

     consumes:

       - application/json

       - application/xml

     produces:

       - application/json

       - application/xml

     parameters:

       - in: body

         name: body

         description: Payment Request object that needs to be added to the store

         required: true

         schema:

           $ref: "#/definitions/PaymentRequest"

     responses:

       "405":

         description: Validation exception

       "404":

         description: Payment Request not found

       "400":

         description: Invalid ID supplied

 /payment/{paymentId}:

   get:

     tags:

       - payment

     summary: Find Payment Request by ID

     description: Returns a Payment Request when ID < 10.  ID > 10 or nonintegers will simulate API error conditions

     operationId: getPaymentRequestById

     produces:

       - application/json

       - application/xml

     parameters:

       - in: path

         name: paymentId

         description: ID of PaymentRequest that needs to be fetched

         required: true

         type: integer

         format: int64

     responses:

       "404":

         description: Payment Request not found

       "200":

         description: successful operation

         schema:

           $ref: "#/definitions/PaymentRequest"

       "400":

         description: Invalid ID supplied

   delete:

     tags:

       - payment

     summary: Deletes a Payment Request

     description: ""

     operationId: deletePaymentRequest

     produces:

       - application/json

       - application/xml

     parameters:

       - in: header

         name: api\_key

         description: ""

         required: true

         type: string

       - in: path

         name: paymentId

         description: Payment Request id to delete

         required: true

         type: integer

         format: int64

     responses:

       "400":

         description: Invalid payment request value

 /users:

   post:

     tags:

       - user

     summary: Create user

     description: This can only be done by the logged in user.

     operationId: createUser

     produces:

       - application/json

       - application/xml

     parameters:

       - in: body

         name: body

         description: Created user object

         required: false

         schema:

           $ref: "#/definitions/User"

     responses:

       default:

         description: successful operation

 /users/login:

   get:

     tags:

       - user

     summary: Logs user into the system

     description: ""

     operationId: loginUser

     produces:

       - application/json

       - application/xml

     parameters:

       - in: query

         name: username

         description: The user name for login

         required: false

         type: string

       - in: query

         name: password

         description: The password for login in clear text

         required: false

         type: string

     responses:

       "200":

         description: successful operation

         schema:

           type: string

       "400":

         description: Invalid username/password supplied

 /users/logout:

   get:

     tags:

       - user

     summary: Logs out current logged in user session

     description: ""

     operationId: logoutUser

     produces:

       - application/json

       - application/xml

     responses:

       default:

         description: successful operation

 /users/{username}:

   get:

     tags:

       - user

     summary: Get user by user name

     description: ""

     operationId: getUserByName

     produces:

       - application/json

       - application/xml

     parameters:

       - in: path

         name: username

         description: The name that needs to be fetched. Use user1 for testing.

         required: true

         type: string

     responses:

       "404":

         description: User not found

       "200":

         description: successful operation

         schema:

           $ref: "#/definitions/User"

       "400":

         description: Invalid username supplied

   put:

     tags:

       - user

     summary: Updated user

     description: This can only be done by the logged in user.

     operationId: updateUser

     produces:

       - application/json

       - application/xml

     parameters:

       - in: path

         name: username

         description: name that need to be deleted

         required: true

         type: string

       - in: body

         name: body

         description: Updated user object

         required: false

         schema:

           $ref: "#/definitions/User"

     responses:

       "404":

         description: User not found

       "400":

         description: Invalid user supplied

   delete:

     tags:

       - user

     summary: Delete user

     description: This can only be done by the logged in user.

     operationId: deleteUser

     produces:

       - application/json

       - application/xml

     parameters:

       - in: path

         name: username

         description: The name that needs to be deleted

         required: true

         type: string

     responses:

       "404":

         description: User not found

       "400":

         description: Invalid username supplied

definitions:

 User:

   properties:

     id:

       type: integer

       format: int64

     username:

       type: string

     firstName:

       type: string

     lastName:

       type: string

     email:

       type: string

     password:

       type: string

     phone:

       type: string

     userStatus:

       type: integer

       format: int32

       description: User Status

 Sender:

   properties:

     name:

       type: string

     email:

       type: string

 PaymentRequest:

   required:

     - sender

     - name

     - description

     - reference

     - date

   properties:

     id:

       type: integer

       format: int64

     sender:

       $ref: "#/definitions/Sender"

     name:

       type: string

     description:

       type: string

     reference:

       type: string

     date:

       type: string

       format: date-time

|  |
| --- |
|  |