Nomes: Gustavo Henrique e Ícaro Adão Data: 30/08/18

Turma: 303 Professor: João Eduardo Montandon

Disciplina: Tecnologias de Programação – TP

Relatório de Trabalho

API: fixer.io (API de cotação de moedas)

INTRODUÇÃO:

Nosso software gira em torno da cotação de moedas com base no euro e a API utilizada para tal tarefa é a fixer.io. A API por si só oferecia determinados planos com diferentes tipos de recursos de acesso dentro da própria API; diante disso viamos a possibilidade de usar apenas o plano grátis disponibilizado. Nesse plano tínhamos acesso básico à alguns recursos oferecidos pela API, e o principal recurso que decidimos trabalhar, dentro das possibilidades, foi o de listagem de cotações recentes no espaço regular de uma hora.

PROBLEMAS:

LIMITAÇÕES NO PLANO DA API

O principal problema enfrentado pelo grupo foi a limitação do plano grátis da API em comparação com os planos pagos. Exemplificando, a falta de acesso à conversões diretas entre moedas, e também a posssibilidade de escolha de base da cotação entre moedas.

IMPLEMENTAÇÃO DO MECANISMO DE BUSCA

Outro grave problema encontrado foi durante a implementação do recurso de busca à moedas. A ideia é que a busca se adapte dinamicamente conforme a entrada que o usuário digita. A dificuldade encontrada foi perder o dado ‘cotação’ da última requisição, isso nos levou a ter várias sugestões sobre como resolver o problema, no entanto muitas deram errado e outras não eram viáveis, como por exemplo a tentativa de fazer várias requisições à API enquanto o usuário digitava no campo de busca. Essa escolha acarretaría um sobrecarregamento de processamento do aplicativo e por consequência uma mal funcionamento do mesmo.

LISTAGEM DE MOEDAS

Em relação à listagem de moedas e suas respectivas cotações o grupo também teve que considerar a estrutura da resposta dada à requisição à API. Essa estrutura tinha a característica de retornar o nome da moeda e sua cotação concatenados, e não o retorno do nome e cotação em variáveis separadas

MOTIVAÇÕES:

A principal motivação do grupo era não desistir das funcionalidades e não abrir mão de uma possível utilidade que esse aplicativo possa representar à um possível usuário, a começar por nós mesmos. Tinhamos muitos recursos ,como fonte de pesquisa, disponíveis, e sabíamos que erámos competentes para o desenvolvimento das funcionalidades requeridas.

SOLUÇÕES e DECISÕES TÉCNICAS:

DESENVOLVIMENTO DE FUNÇÕES PRÓPIAS

A API possui limitações em relação à suas própias funções que são definidas pelo plano do usuário. Como o grupo possuía apenas o plano grátis que dava acesso às cotações de moedas e restringia a cotação com base no Euro tivemos que desenvolver nossas próprias funções com base nos valores das cotações. Uma das funções desenvolvidas foi a conversão de moedas com base no Euro e vice e versa.

SUGESTÕES DE BUSCA FILTRADAS EM UM ARRAYLIST

Tínhamos o problema de perder os dados da útima requisição à cada vez que o usuário digitava algo na barra de busca, então a partir disso resolvemos guardar os útimos dados retornados pela requisição em um arraylist contido na função **OnQueryTextChange** que se encontra na **Main**. Com isso o dado retornado pela busca era guardado e os dados de cotação expostos na listView mantinham sua integridade original.

COMPONENTE BÁSICO DE LISTAGEM

Por característica própria da API, o json retornado através da requisição das útimas cotações “**latest**”, não retornava o resultado em modúlos que continham nome e cotação, as cotações possuiam os dados concatenados. A solução foi definir um componente básico em forma de uma classe chamada **Taxa,** que continha nome e cotação, juntamente com uma outra classe chamada “**Taxas**”**,** que continham um arraylist de **Taxa**. A API retorna aproximadamente 140 moedas, diante dessa situação escolhemos os principais escudos de moeda da atualidade “incluindo o Real – BRL”, e os representamos arbitrariamente na classe **Taxas**. Depois modelamos nosso ListView em função da classe **Taxa**.

MODELAGEM DA RESPOSTA Json

Para representarmos as cotações de maneira coerente usamos ao todo 3 classes que modelam a resposta do Json retornado pela requisição das útimas cotações.

- **UltimasTaxas**: é a classe de resposta padrão do requesição. Além de retornar um objeto(**rates**) com as cotações retorna também outros dados, como o fuso-horário, a data da requisição, etc.
- **Taxas**: desenvolvida para coletar os dados do objeto **“rates”**, possui os getters apropriados. Nela também há as moedas arbitrárias com suas respectivas cotações.
- **Taxa:** é a classe que representa uma moeda apenas. Possui como atributos o nome da moeda e sua cotação. A ListView gira em torno dessa classe.