

Integração com APIs Remotas (REST/GraphQL)

Grupo: 7

Programação de Dispositivos Móveis

Elaborado por: Inês Gomes nº51554 / Manuel Canhoto nº51670/ Tiago Monteiro nº51804/
Maria Pedro nº52080/ Sara Santos nº52245

Professor: Paulo André Pais Fazendeiro

O que é uma API?

Uma API, ou Interface de Programação de Aplicações, é um conjunto de regras, protocolos e ferramentas que permite que diferentes softwares se comuniquem entre si, funcionando como um intermediário para a troca de informações e funcionalidades de forma segura e eficiente.



REST vs GraphQL

REST

REST, ou Representational State Transfer, é um estilo arquitetural para sistemas distribuídos, como a web, que se baseia em um conjunto de princípios e restrições para a comunicação entre cliente e servidor.

GraphQL

GraphQL é uma linguagem de consulta para APIs e um ambiente de execução que permite que os clientes solicitem exatamente os dados de que precisam, sem sobrecarga ou necessidade de múltiplas requisições

Critérios de Escolha

1.

Usamos REST para aplicações mais simples, com operações de criar, ler, atualizar e apagar bem definidas e quando a cache simples é vantajosa

2.

Já o GraphQL é ideal para aplicações que precisam de dados flexíveis ou que envolvem múltiplas requisições de dados, como aplicativos móveis, pois permite ao cliente solicitar exatamente o que precisa em uma única requisição.

Boas Práticas Gerais em APIs

Resiliência

- Limites de tempo
 - Novas tentativas controladas
 - Pausas temporárias
-

Erros claros

- Quando dá erro, é fácil entender o quê e porquê
 - Códigos e mensagens consistentes
-

Segurança

- Autenticação por tokens
 - Regras de acesso no navegador
 - Validação/filtragem de entradas
-

Boas Práticas Gerais em APIs

Observabilidade

- Registos organizados
 - Identificador por pedido
 - Métricas essenciais
-

Paginação

- Listas por partes e marcadores:
 - Limites e regras claras
-

Versionamento

- REST: só criamos nova versão quando há quebras; indicamos a versão e mantemos coexistência com um guia de migração.
 - GraphQL: evitamos novas versões; avisamos e retiramos campos com prazo, mantendo compatibilidade.
-

Resumo comparativo

Critério	REST	GraphQL
Quantidade de dados	Pode acontecer receber dados a mais ou a menos	Cliente pede apenas o que precisa
Cache	Tira bom partido da cache web	Cache mais usada no lado do cliente
Descrição da API	Descrito através de ferramentas externas	Faz parte do próprio sistema
Erros	Códigos de resposta + mensagem de erro	Dados parciais + lista de erros
Paginação	Páginas simples ou por marcadores	Paginação incremental com indicação de continuação
Riscos	Muitos pontos de acesso	Consultas pesadas se não houver limites
Versionamento	Criação de novas versões quando há mudanças grandes	Evita-se criar versões; campos são descontinuados com aviso

Mini-App

MainActivity.java

```
btnRest = findViewById(R.id.btnRest);  
btnGraphQL = findViewById(R.id.btnGraphQL);  
  
btnRest.setOnClickListener(v ->  
    startActivity(new Intent(this, RestActivity.class))  
);  
  
btnGraphQL.setOnClickListener(v ->  
    startActivity(new Intent(this, GraphQLActivity.class))  
);
```


RestActivity.java

```
private void fetchQuotesAsPosts() {
    showLoading(true);
    ApiClient.rest().getQuotes(120, 10).enqueue(new Callback<QuoteResponse>() {
        @Override public void onResponse(Call<QuoteResponse> c, Response<QuoteResponse> resp) {
            showLoading(false);
            if (resp.isSuccessful() && resp.body() != null) {
                List<Post> posts = new ArrayList<>();
                for (Quote q : resp.body().getResults()) {
                    Post p = new Post();
                    p.setId(q.getLength());
                    p.setTitle(q.getAuthor());
                    p.setBody(q.getContent());
                    posts.add(p);
                }
                showPosts(posts);
            } else { showError("Erro " + resp.code()); }
        }
        @Override public void onFailure(Call<QuoteResponse> call, Throwable t) {
            showLoading(false); showError(t.getMessage());
        }
    });
}
```

GraphQLActivity.java

```
private void fetchCountries() {
    showLoading(true);

    JsonObject body = new JsonObject();
    body.addProperty("query",
        "query { countries { name capital emoji currency } }"
    );

    ApiClient.graphql().runQuery(body).enqueue(new Callback<JsonObject>() {
        @Override public void onResponse(Call<JsonObject> call, Response<JsonObject> resp)
        {
            showLoading(false);
            JSONArray arr = resp.body().getAsJsonObject("data").getAsJsonArray("countries");
            List<Country> list = new ArrayList<>();
            for (JsonElement el : arr) {
                JsonObject obj = el.getAsJsonObject();
                list.add(new Country(obj.get("name").getString(),
                    obj.get("capital").isJsonNull()? null : obj.get("capital").getString(),
                    obj.get("emoji").getString(),
                    obj.get("currency").isJsonNull()? null : obj.get("currency").getString()));
            }
            adapter.updateData(list);
        }
        @Override public void onFailure(Call<JsonObject> call, Throwable t) {
            showLoading(false); showError(t.getMessage());
        }
    });
}
```

CountryAdapter.java

```
@Override
public void onBindViewHolder(@NonNull
    CountryViewHolder h, int pos) {
    Country c = countries.get(pos);
    String line1 = c.getEmoji() + " " + c.getName();
    String line2 = "Capital: " + (c.getCapital() == null ?
        "—" : c.getCapital());
    h.text1.setText(line1);
    h.text2.setText(line2);
}
```

ApiClient.java / ApiService.java

```
public class ApiClient {
    private static Retrofit restRetrofit, graphqlRetrofit;

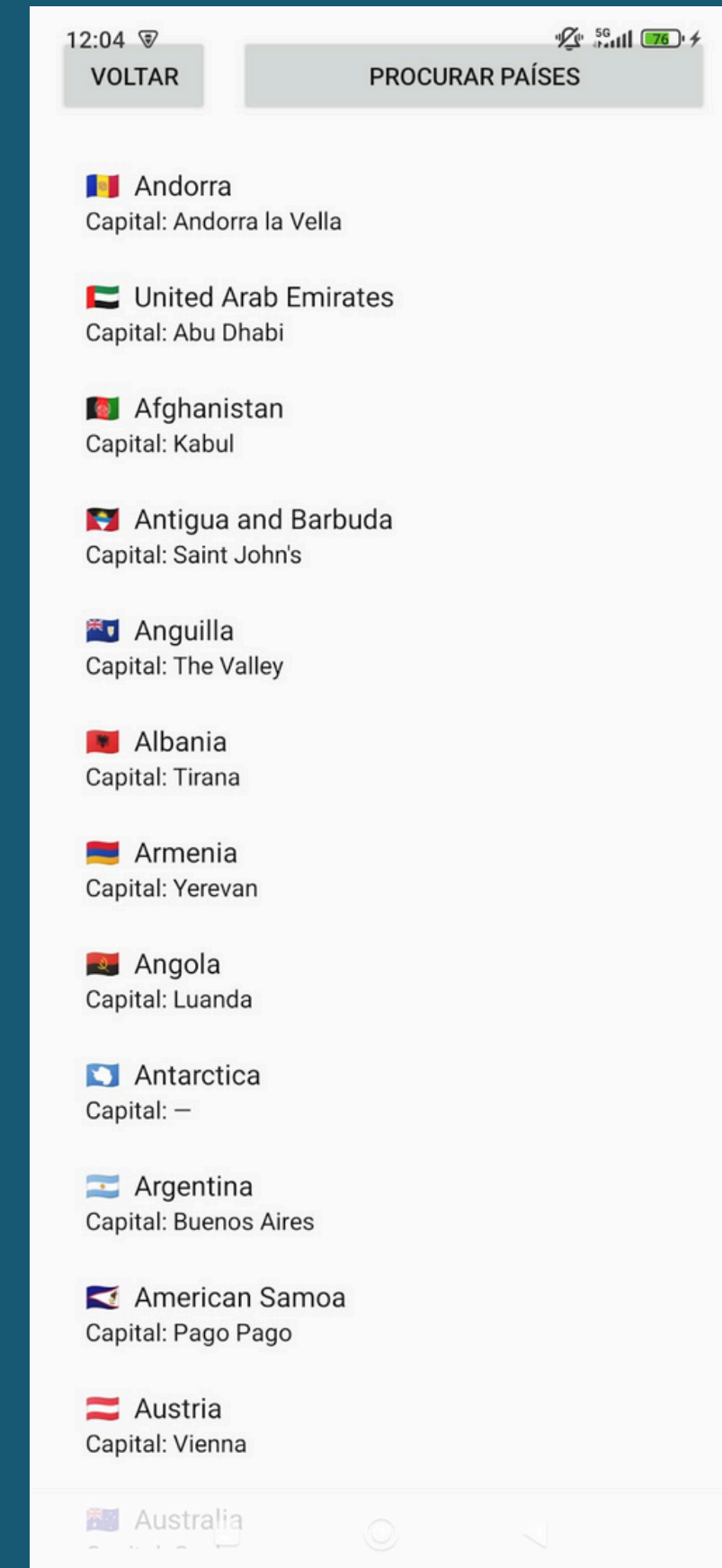
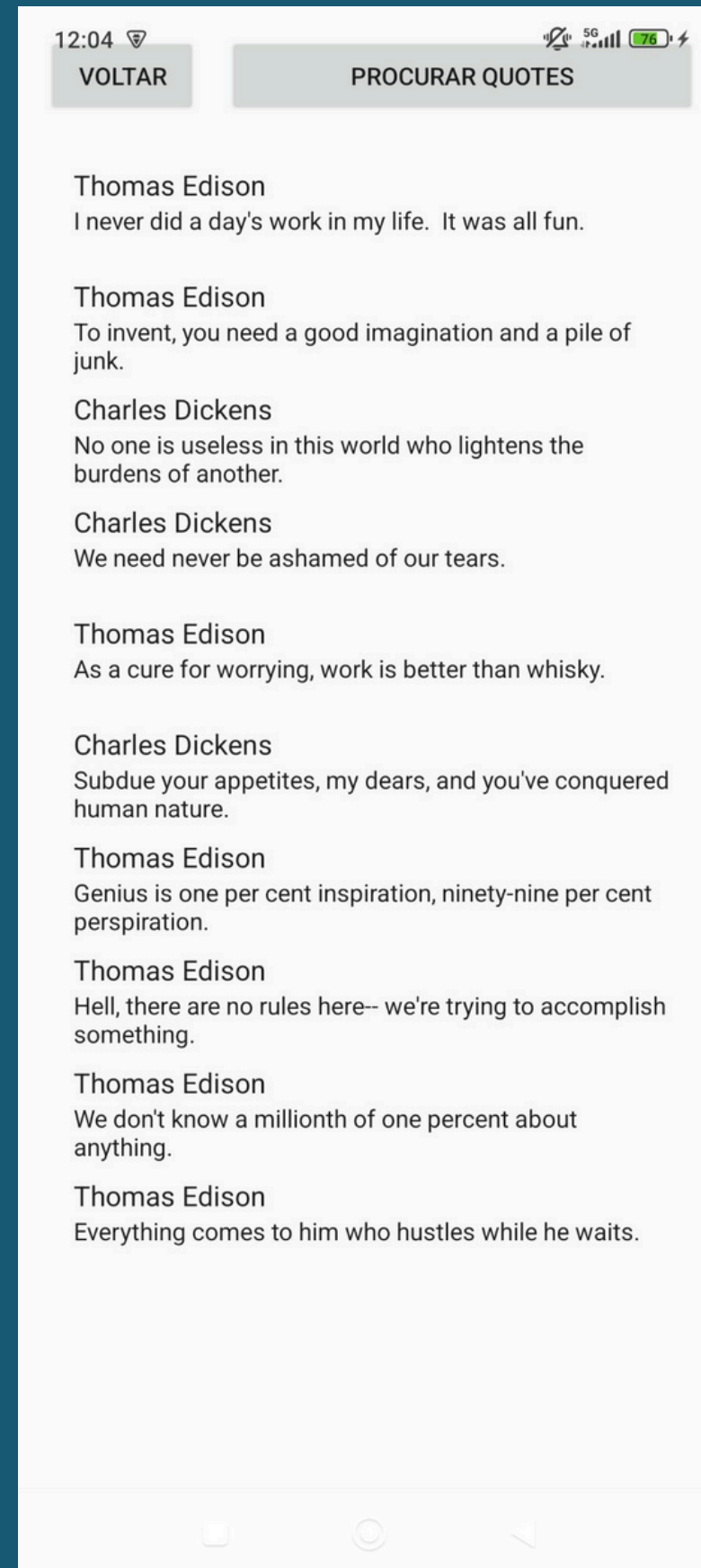
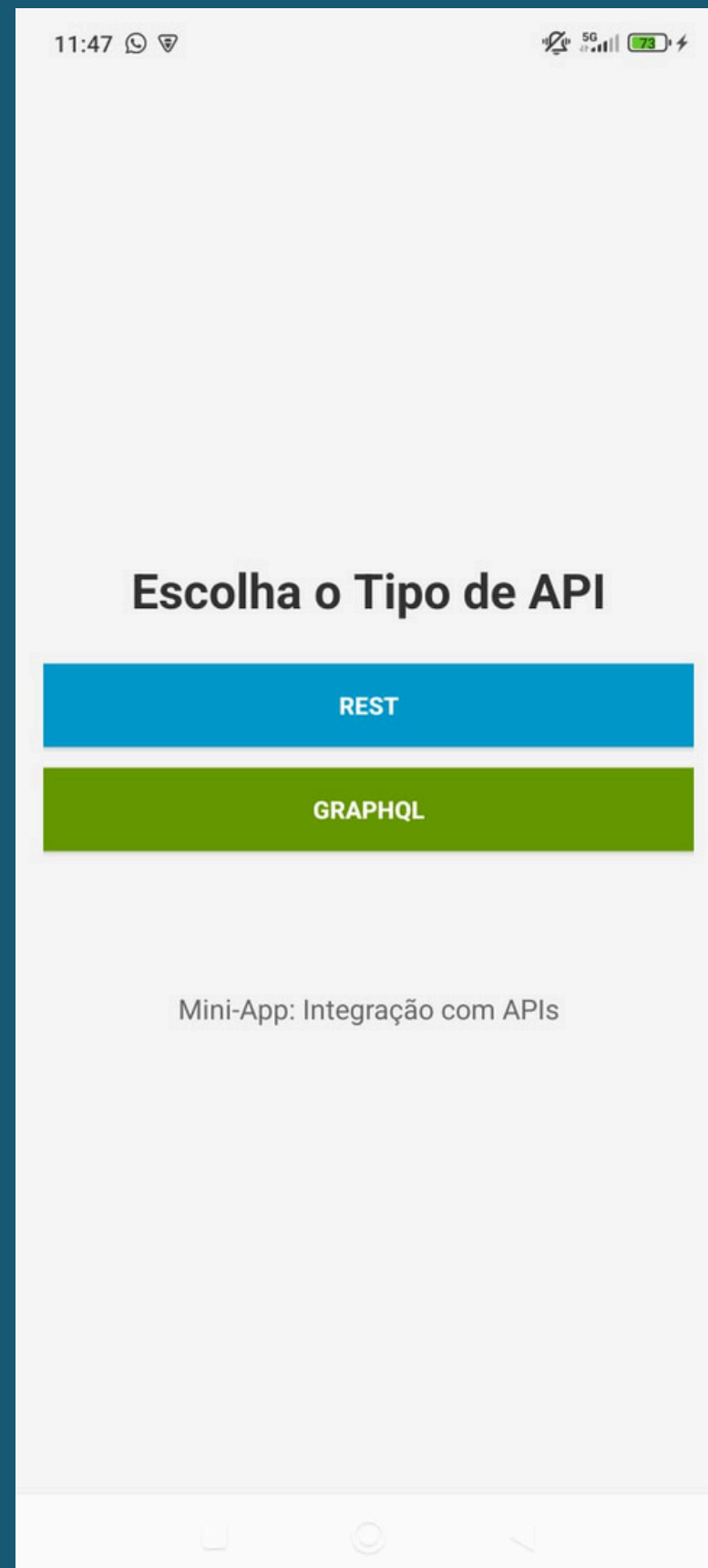
    public static ApiService rest() {
        if (restRetrofit == null) {
            restRetrofit = new Retrofit.Builder()
                .baseUrl(ApiService.BASE_URL)
                .addConverterFactory(GsonConverterFactory.create())
                .build();
        }
        return restRetrofit.create(ApiService.class);
    }

    public static GraphQLApi graphql() {
        if (graphqlRetrofit == null) {
            graphqlRetrofit = new Retrofit.Builder()
                .baseUrl("https://countries.trevorblades.com/")
                .addConverterFactory(GsonConverterFactory.create())
                .build();
        }
        return graphqlRetrofit.create(GraphQLApi.class);
    }
}
```

```
public interface ApiService {
    String BASE_URL = "http://api.quotable.io/";

    @GET("quotes")
    Call<QuoteResponse> getQuotes(
        @Query("maxLength") int maxLength,
        @Query("limit") int limit
    );
}
```

Layout



Obrigado!
Grupo7
