|  |  |  |
| --- | --- | --- |
|  | | **Departamento de Engenharia Informática**  **TeSP em Programação de Sistemas de Informação**  **Acesso Móvel a Sistemas de Informação**  2016/2017 |
|  |  |  |
|  |  |  |
| **Docentes** | Sérgio Lopes, sergio.lopes@ipleiria.pt | |
|  | David Safadinho, david.safadinho@ipleiria.pt | |

Ficha de Exercícios N/08

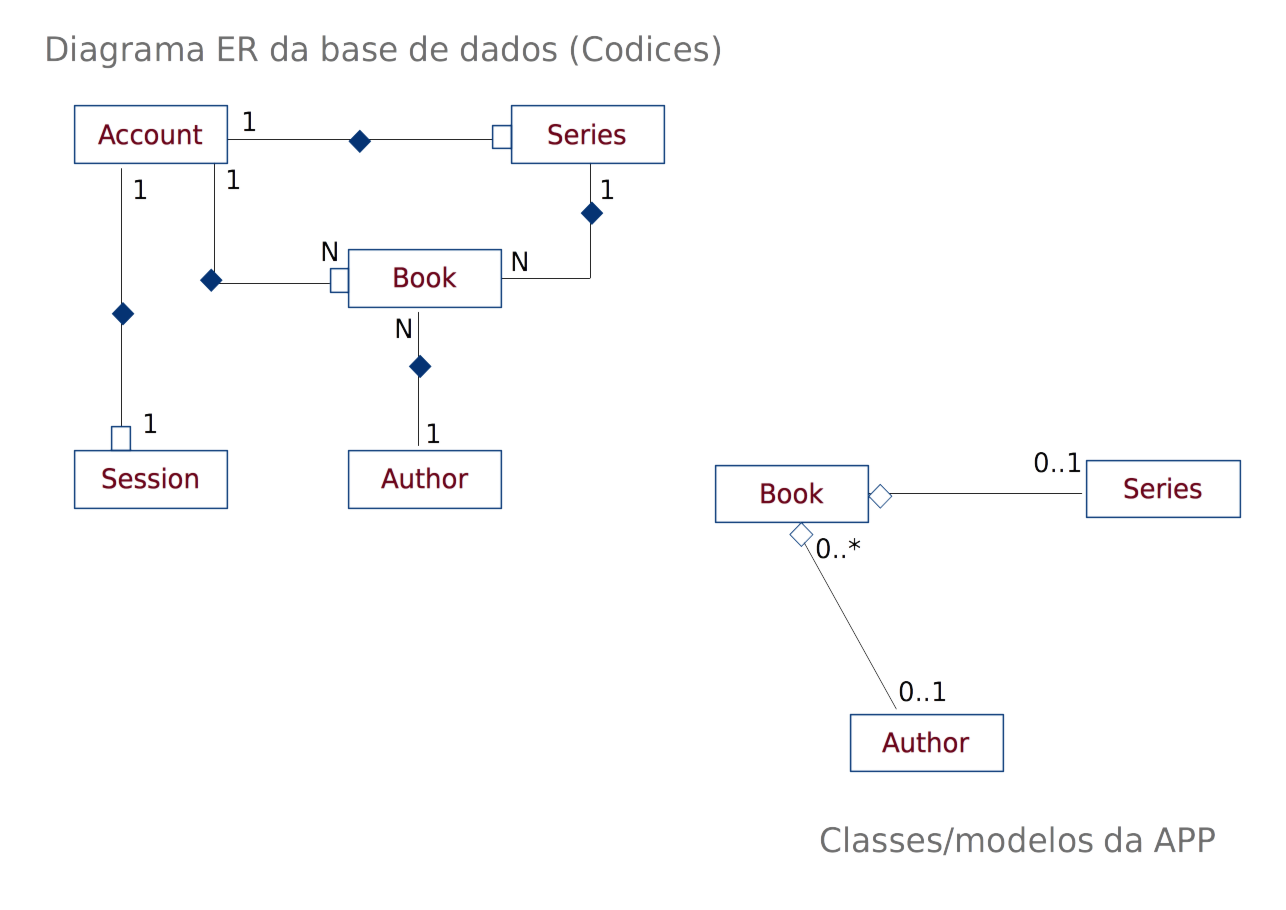
Adicionar acesso a uma API

# *Acesso à API REST do sistema Codices*

Como sistema exemplo para acesso a uma API vamos usar o sistema de gestão de biblioteca pessoal chamado “Codices” (<https://github.com/Knitter/codices)>, e aceder a uma instalação de testes disponível em [www.codicesapp.com](http://www.codicesapp.com), a API estará acessível através do endereço [www.codicesapp.com/rest/v1](http://www.codicesapp.com/rest/v1).

## Diagrama ER e de classes

Com as alterações da ficha anterior o modelo de dados da aplicação e da base de dados da API passa a incluir mais entidades e mais relações, como descrito abaixo.

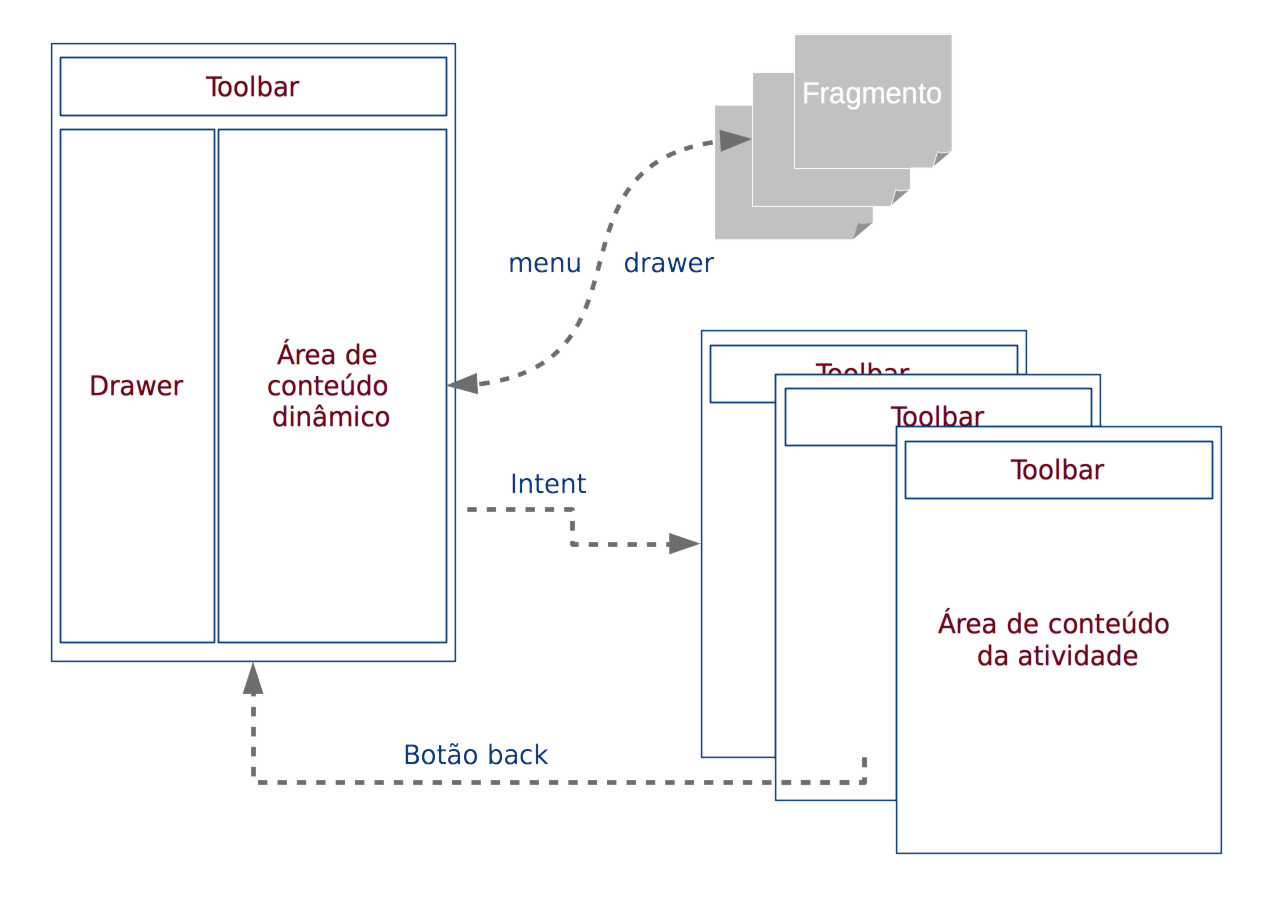


Embora existam relações entre tabelas que dariam origem a tabelas resultantes (ex.: relação entre **Book** e **Author**), essas tabelas não foram criadas por questões de otimização. Assim, existem no servidor apenas cinco tabelas (e igual número de modelos *ActiveRecord*) correspondendo às cinco entidades do diagrama.

Do mesmo modo, algumas das entidades não deram lugar a classes na aplicação, sendo que apenas existem três classes para representar os dados principais. Como a aplicação apenas permite uma conta autenticada não faz sentido manter no dispositivo uma tabela para as contas de utilizadores (**Account**).

## Interface gráfica

A interface gráfica é a mesma que foi criada/alterada na ficha anterior, sendo constituída por uma atividade principal que gere os fragmentos com lista de dados, o menu principal (*drawer*), faz a ponte entre os vários fragmentos e as atividades de edição/criação de dados e mantém uma *toolbar* comum. Existem três entidades, uma para editar/criar cada um dos tipos de dados e existe um painel de opções onde são definidas algumas configurações base para a aplicação.



|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

## Outras alterações

As configurações do AndroidManifest.xml foram alteradas para que possam apresentar corretamente o nome da aplicação e não o nome/título da primeira atividade.

O ficheiro **build.gradle** foi atualizado para conter um número de versão e nome de versão que correspondam às alterações que têm vindo a ser aplicadas e assim respeitar a obrigação de manter um número único que permita ao sistema *Android* validar as atualizações.

Todos os ficheiros XML foram revistos para que fosse aplicada sempre a mesma regra de escrita de código. Nestes ficheiros, os ID dos elementos passam a ser escritos em **snake\_case**, mantendo-se a escrita em **camelCase** para o código Java, facilitando assim a distinção entre os dois ambientes e entre as variáveis de um e de outro.

Já incluída, a biblioteca *Ion* (<https://github.com/koush/ion)> será usada para aceder a API REST, que juntamente com a biblioteca GSON (<https://github.com/google/gson>, incluída como dependência da *Ion*), facilita o acesso e conversão de dados de JSON para objetos Java.

## Recursos da API

Para aceder à API deve ser usado o endereço base [www.codicesapp.com/rest/v1](http://www.codicesapp.com/rest/v1), ao qual deve ser adicionado o recurso final pretendido, de entre os seguintes:

* **/books**, requer autorização
  + GET, lista de livros;
  + POST, criar novo livro;
  + PUT, editar livro existente;
  + DELETE, remover um livro
* **/series**, requer autorização
  + Mesmas ações que para o recurso anterior
* **/authors**, requer autorização
  + Mesmas ações que para o recurso anterior
* **/account**, requer autorização
  + Mesmas ações que para o recurso anterior
* **/account/authenticate**
  + Apenas POST, possibilita a autenticação de um utilizador

Todos os pedidos recebem/devolvem JSON, e para os pedidos com necessidade de autorização é obrigatório o envio de um cabeçalho com o nome **X-CODICESUSER-TOKEN**.

## Conversão JSON – Java

Cada uma das três classes do modelo de dados da aplicação deve ter um conversor que permite converter os dados da API para objetos Java (usando a biblioteca GSON). Para isso é necessário criar três novas classes que implementem a interface **JsonDeserializer** e convertam os dados no método **deserialize()**, ex.:

public class BookDeserializer implements JsonDeserializer<Book> {

@Override

public Book deserialize(JsonElement json, Type typeOfT, JsonDeserializationContext context)

throws JsonParseException {

//Código de conversão

}

# Exercício

Com base na versão **3.1.0** da aplicação disponível em <https://github.com/Knitter/amsi-books/releases/tag/v3.1.0> e usando a documentação da biblioteca *Ion*, adicione à aplicação acesso à API disponível em <http://codicesapp.com/rest/v1> de forma a que a aplicação use diretamente os dados fornecidos por esta API. Comece pelo processo de autenticação e autorização.

Altere o acesso à base de dados para que o servidor da API seja sempre o responsável por validar todas as alterações aos dados, passando a base de dados a ser usada apenas como cache local.

### Passos/alterações a executar:

* Efetuar a autenticação no servidor e obter o ***token*** de acesso para autorização dos pedidos seguintes;
* Obter a lista de livros da API;
* Obter a lista de autores da API;
* Obter a lista de séries da API;
* Criar/editar livros, autores e séries;

Os diagramas seguintes exemplificam, respetivamente, as decisões/fluxo de execução e a sequência inicial de pedidos quando o utilizador tenta ver a lista de livros na aplicação.

|  |  |
| --- | --- |
|  |  |