

**Faculdade de Engenharia da Universidade do Porto**



## **Base de dados aplicada à gestão de frota de Companhias Aéreas**

SEGUNDA SUBMISSÃO

Antero Morgado [up202204971@fe.up.pt](mailto:up202204971@fe.up.pt)

Tiago Loureiro [up200706716@fe.up.pt](mailto:up200706716@fe.up.pt)

Hugo Cruz [up202205022@fe.up.pt](mailto:up202205022@fe.up.pt)

Projeto realizado no âmbito da Licenciatura em Engenharia Informática  
e Computação para a Unidade Curricular de Bases de Dados

Professor das aulas Teórico-Práticas: Michel Celestino Paiva Ferreira

19-10-2023

# Índice

Introdução .....	3
1. Modelo Conceptual Modificado .....	4
2. Ferramenta de Inteligência Artificial Generativa .....	5
3. Modelo Relacional .....	6
3.1. Modelo Relacional feito a partir do Modelo Conceptual .....	6
3.2. Retificações usando Inteligência Artificial .....	7
4. Análise de Dependências Funcionais e Formas Normais.....	9
4.1. Dependências Funcionais .....	9
4.2. Forma Normal de Boyce Codd .....	10
4.3. Terceira Forma Normal.....	10
4.4. Retificações usando Inteligência Artificial .....	11
5. SQLite.....	12
5.1. Desenvolvimento das Tabelas .....	12
Usando Inteligência Artificial .....	12
5.2. População das Tabelas.....	14
Usando Inteligência Artificial .....	14
Referências Bibliográficas .....	15

## Introdução

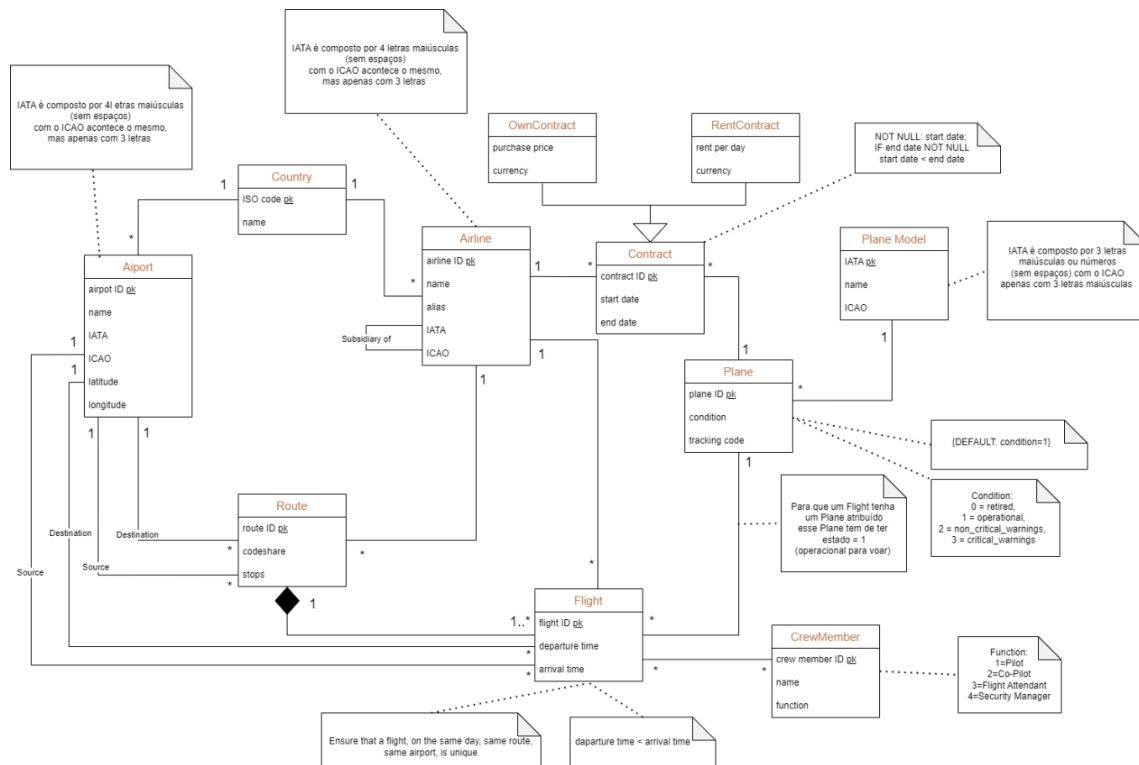
Este projeto foi realizado no âmbito da unidade curricular Bases de Dados, do segundo ano da Licenciatura em Engenharia Informática e Computação, da Faculdade de Engenharia da Universidade do Porto.

O projeto tem como objetivo a implementação de uma base de dados, sendo para isso necessária a criação de um modelo conceptual (UML), obter um modelo relacional que lhe seja correspondente e, por fim, implementar esse modelo numa base de dados SQLite.

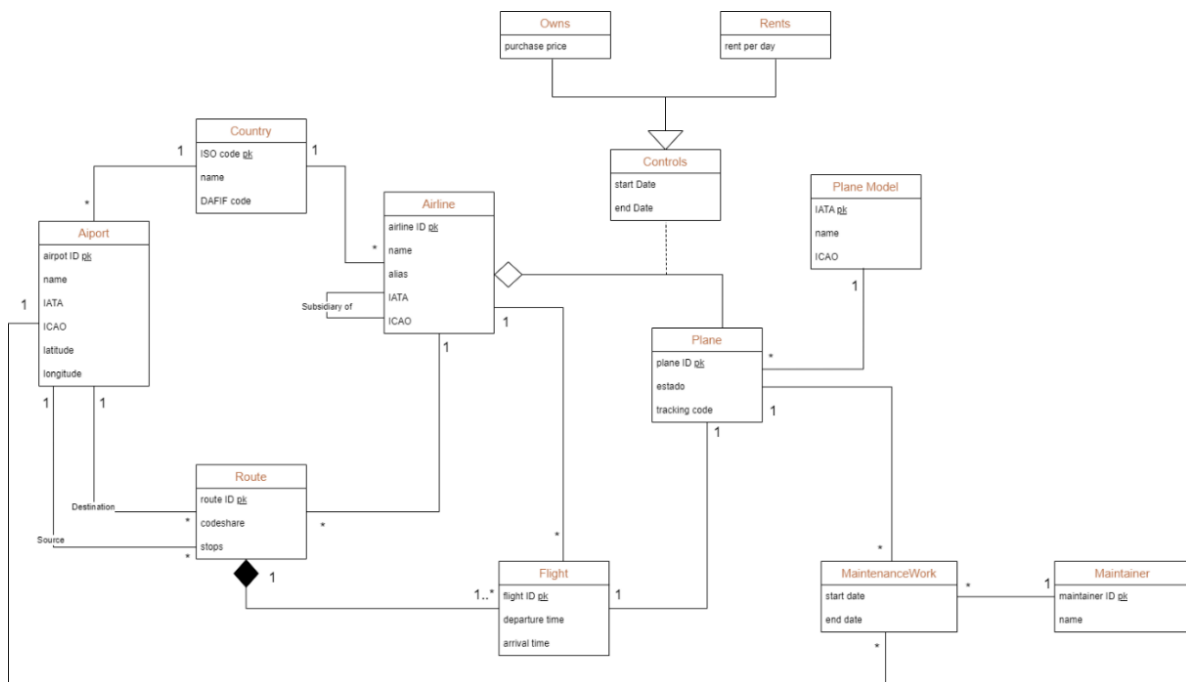
O tema escolhido foi a gestão de frotas de aviões por parte de companhias aéreas e, nesta segunda submissão do projeto, não será abordado a fundo a criação do modelo conceptual (UML), visto já ter sido tema da primeira submissão, mas serão sim tratados no presente relatório o modelo relacional e se o mesmo está, ou não de acordo com as formas normais e também a implementação em SQLite, desde a criação das tabelas à sua população com tuplos.

## 1. Modelo Conceptual Modificado

Após a receção do feedback do Professor Michel Ferreira o grupo alterou o modelo conceptual entregue na primeira parte e chegou ao seguinte diagrama UML:



É possível ver o diagrama a partir do qual foram feitas as alterações na seguinte imagem:



## 2. Ferramenta de Inteligência Artificial Generativa

No presente projeto, é solicitado que, em diversas etapas da concepção da base de dados, o trabalho seja discutido com uma ferramenta de inteligência artificial generativa. Nesta segunda etapa, como é baseada em texto e não em um diagrama UML, optou-se por usar o ChatGPT3.5.

A escolha do ChatGPT3.5 deve-se a vários fatores. Em primeiro lugar, é uma das IAs mais populares do momento, o que garante que está bem testada e tem um bom histórico de desempenho. Em segundo lugar, é uma ferramenta muito útil no mundo da programação e das ciências da computação.

A ferramenta pode ajudar a gerar código, escrever documentação, gerar dados de teste e corrigir erros. Isso pode economizar tempo e esforço para os programadores, e pode levar a bases de dados mais precisas e robustas.

## 3. Modelo Relacional

### 3.1. Modelo Relacional feito a partir do Modelo Conceptual

Abaixo poderá ver o modelo relacional:

**Airport** (id, name, iata, icao, latitude, longitude, iso\_code->Country)

**Route** (id, codeshare, stops, source\_id->Airport, destination\_id->Airport, airline\_id->Airline)

**Country** (iso\_code, name)

**Airline** (id, name, alias, iata, icao, parent\_id->Airline, iso\_code->Country)

**Flight** (id, departure\_time, arrival\_time, source\_id->Airport, destination\_id->Airport, plane\_id->Plane, airline\_id->Airline, route\_id->Route)

**FlightCrewMember** (flight\_id->Flight, crew\_member\_id->CrewMember)

**CrewMember** (id, name, function)

**Plane** (id, condition, tracking\_code, iata->PlaneModel)

**PlaneModel** (iata, name, icao)

**Contract** (id, airline\_id->Airline, plane\_id->Plane, start\_date, end\_date)

**OwnContract** (contract\_id->Contract, purchase\_price, currency)

**RentContract** (contract\_id->Contract, rent\_per\_day, currency)

### 3.2. Retificações usando Inteligência Artificial

Como solicitado, foi fornecido o modelo ao ChatGPT e foi feita a seguinte pergunta:

*In the context of a Database to be used for Airplane Fleet Management.  
Do you see any errors in this Database Relational Model? Do you have any suggestions to improve it?*

Ao que respondeu:

*Your relational model seems quite comprehensive for airplane fleet management. However, I have a few observations and suggestions for potential improvements:*

**1. Normalization and Relationships:**

- Ensure that the relationships are well-established and normalized to avoid redundancy and maintain consistency.
- Consider breaking down the CrewMember table if there are specific roles or types of crew members. For instance, separating pilots from cabin crew might offer more specific information and easier management.
- Explore whether the PlaneModel table could be related to the Plane table in a more structured way to maintain details about various models and their individual characteristics.

**2. Data Integrity:**

- Implement constraints like foreign key constraints and indexes to maintain data integrity and improve query performance.
- Consider adding constraints for time, especially in Flight and Contract tables, to avoid overlapping schedules or conflicting data.

**3. Aircraft Maintenance and History:**

- Consider adding a table to track maintenance history for planes. It could contain maintenance type, date, cost, and any other relevant details.
- Including a table for incidents or issues related to flights or planes could provide a historical record and aid in analysis and improvement.

**4. Contracts and Financials:**

- For contracts, consider if there are specific terms or conditions that could be better represented in separate tables, especially if they vary greatly among contracts.
- Financial information such as revenue generated per flight, operating costs, or profit margins could be added to provide a holistic view of the airline's operations.

**5. Data Consistency:**

- Ensure consistency in data types across tables (e.g., currency formats, date formats) to avoid confusion or errors in data manipulation.

*Remember, the effectiveness of the model highly depends on the specific needs and intricacies of the airline's operations. Adapting the model to suit those needs precisely is crucial for its success.*

E o grupo decidiu que:

- Manter o nível de detalhe, por ser irrelevante para o caso de uso ou por não acrescentar nenhum desafio novo a nível de aprendizagem. Como o CrewMember tem especialização por função, pois não teriam atributos novos. O PlaneModel podia ter as características, como VelocidadeMax, Peso, etc. mas é irrelevante para o caso de uso;
- Implementar as restrições;
- Concordou-se com a ideia de Maintenance, mas devido a feedback da primeira submissão, optou-se por retirar;

- Concordou-se que num caso de uso de produção, ter os dados dos contratos e custos seria importante para apoiar a tomada de decisão de qual avião usar. No entanto deixar simples e deixar subentendido que poderia ser expansível para quaisquer atributos foi o escolhido;
- O grupo concordou com a consistência de dados, e estão todos com o mesmo type. A nível de aplicação será importante todos os Datetime serem guardados em UTC para comparação.



## 4. Análise de Dependências Funcionais e Formas Normais

### 4.1. Dependências Funcionais

#### **Airport:**

Airport(id) -> name, iata, icao, latitude, longitude, iso\_code

#### **Route:**

Route(id) -> codeshare, stops, source\_id, destination\_id,  
airline\_id

#### **Country:**

Country(iso\_code) -> name

#### **Airline:**

Airline(id) -> name, alias, iata, icao, parent\_id, iso\_code

#### **Flight:**

Flight(id) -> departure\_time, source\_id, destination\_id,  
arrival\_time, plane\_id, airline\_id, route\_id

#### **Plane:**

Plane(id) -> condition, tracking\_code, iata

#### **PlaneModel:**

PlaneModel(iata) -> name, icao

#### **Contract:**

Contract(id) -> airline\_id, plane\_id, start\_date, end\_date

#### **OwnContract:**

OwnContract(contract\_id) -> purchase\_price, currency

#### **RentContract:**

RentContract(contract\_id) -> rent\_per\_day, currency

#### **CrewMember:**

CrewMember(id) -> name, function

## 4.2. Forma Normal de Boyce Codd

As dependências funcionais fornecidas estão na forma normal de *Boyce-Codd* (BCNF) porque todas são não triviais e cada dependência funcional tem uma superchave na parte da esquerda.

Uma dependência funcional é trivial se o determinante for um subconjunto do dependente. No caso das dependências funcionais fornecidas, nenhum determinante é um subconjunto do dependente. Portanto, todas as dependências são não triviais.

Além disso, todas as dependências funcionais têm uma superchave na parte da esquerda. Uma superchave é um conjunto de atributos que identifica exclusivamente cada tuplo de uma relação. No caso das dependências funcionais fornecidas, cada superchave identifica exclusivamente cada tuplo da tabela correspondente. Portanto, todas as dependências são superchaves.

Assim sendo, as dependências funcionais fornecidas estão na BCNF porque todas são não triviais e têm uma superchave na parte da esquerda.

## 4.3. Terceira Forma Normal

As dependências funcionais fornecidas estão na terceira forma normal (3NF) porque todas são não triviais e cada dependência funcional tem uma chave candidata na parte da esquerda.

Uma dependência funcional é trivial se o determinante for um subconjunto do dependente. No caso das dependências funcionais fornecidas, nenhum determinante é um subconjunto do dependente. Logo, todas as dependências são não triviais.

Além disso, todas as dependências funcionais têm uma chave candidata na parte da esquerda. Uma chave candidata é um conjunto de atributos que pode ser usado para identificar exclusivamente cada tuplo de uma relação. No caso das dependências funcionais fornecidas, cada chave candidata identifica exclusivamente cada tuplo da tabela correspondente. Portanto, todas as dependências são chaves candidatas.

Em suma, as dependências funcionais fornecidas estão na 3NF porque todas são não triviais e têm uma chave candidata na parte da esquerda.

## 4.4. Retificações usando Inteligência Artificial

Como solicitado, com as dependências funcionais acima, perguntou-se ao ChatGPT:

*In the context of an Airline Fleet Management system.  
Please look at these Functional Dependencies and confirm if any error exists.*

Ao que respondeu:

*Looking at the functional dependencies you've provided, they seem well-structured in terms of linking the attributes to their respective identifiers. However, a couple of considerations could be made:*

*1. **Normalization:** In the context of database design, ensuring normalization might be helpful. Some relations might have redundancies or transitive dependencies that can be optimized further.*

*2. **Attribute dependencies:** The functional dependencies seem reasonable based on the descriptions given. However, without seeing the actual data and considering the practical use cases, it's challenging to identify if there might be any specific errors or improvements needed in the design.*

*For instance, consider the relationship between `OwnContract` and `RentContract`. If they both share the same `contract\_id`, it might create issues or inconsistencies if a contract is both owned and rented simultaneously. Depending on the business logic, this might need to be handled differently.*

*Additionally, the relationship between `PlaneModel` and `Plane` seems to be based on the `iata` code, but in reality, a plane might have different models, or a model might be used for multiple planes. This could be a potential area for refinement depending on the specific requirements of the airline fleet management system.*

*Overall, these dependencies seem generally aligned with their corresponding entities, but without deeper knowledge of the system's requirements or the specific data, it's challenging to definitively confirm their accuracy.*

Com base neste feedback, foi considerado que:

- Sobre o OwnContract e RentContract, já se tinha identificado o mesmo e questionado o professor. A consistência só poderá ser garantida através de um trigger. A nível conceptual, está corretamente descrito (há um contract\_id e não um own\_contract\_id).

- Sobre a tabela ser PlaneModel, mas depois a key ser IATA, o ChatGPT está correto. Após a consulta de links, efetivamente um só IATA, pode ter Models com pequenas diferenças. Como um Boeing 787 mais recente ou com um motor ligeiramente diferente. Mas a nível de características, deverá ser semelhante, e como o neste caso de uso é escolher que avião pode fazer um voo nacional / continental / intercontinental, este nível de detalhe é o correto. Também o dataset usado, o OpenFlights, só tem informação de IATA / ICAO para descrever os aviões.

## 5. SQLite

### 5.1. Desenvolvimento das Tabelas

O código SQLite que cria as tabelas pode ser encontrado no ficheiro create1.sql. É deixado aqui apenas um exemplo demonstrativo da criação da tabela Airport:

```
CREATE TABLE AIRPORT (
  ID BIGINT,
  NAME VARCHAR(50),
  IATA CHAR(3),
  ICAO CHAR(4),
  LATITUDE DOUBLE,
  LONGITUDE DOUBLE,
  ISO_CODE CHAR(2),
  PRIMARY KEY (ID),
  FOREIGN KEY (ISO_CODE) REFERENCES COUNTRY (ISO_CODE)
);
```

### Usando Inteligência Artificial

Submeteu-se a seguinte pergunta sobre o schema ao ChatCPT:

*With the context of an Airline Fleet Management system.  
Can you see any errors in the SQL Schema below?*

Ao que respondeu:

*The schema seems mostly well-structured for an airline fleet management system, but there are a few points to consider:*

- 1. Plane Condition:** The `CONDITION` field in the `PLANE` table is defined as `TINYINT`. It might be beneficial to have a reference table or enum defining what these values represent (e.g., 0 for "poor condition," 1 for "good condition," etc.), for clarity and maintenance.
- 2. Flight and Route Relationship:** The uniqueness constraint `UNIQUE(SOURCE\_ID, ROUTE\_ID)` in the `FLIGHT` table might cause issues. It's a good idea to ensure that this constraint aligns with the business logic, as having multiple flights with the same source and route might be allowed.
- 3. Ownership Relationship:** The `OWN\_CONTRACT` and `RENT\_CONTRACT` tables both reference the `CONTRACT` table with a foreign key. This structure assumes that a contract can either be an ownership contract or a rental contract, exclusively. If a contract can be both, consider a different approach or a different structure.
- 4. ICAO and IATA:** The constraints for `ICAO` and `IATA` might be a bit strict. For instance, IATA codes for airports are three characters long, but the constraint is set to accept only alphanumeric characters. Similar considerations apply to `ICAO`.

*5. **Data Consistency:** It's essential to ensure that data types and constraints align with real-world scenarios. For instance, the `CONDITION` of a plane is represented as a `TINYINT`, but the meaning and range of these values should be documented clearly.*

*6. **Foreign Key Operations:** Depending on business logic, cascading deletes and updates should be thoroughly considered. For instance, cascading updates or deletes on the `FLIGHT` table might have significant implications on related tables like `FLIGHT\_CREW\_MEMBER`.*

*Overall, while the schema is comprehensive, it's crucial to ensure that it accurately reflects the business logic and relationships within the airline fleet management system. Testing with sample data and considering various use cases could help validate and refine this schema further.*

Com base neste feedback, decidiu-se:

- Esclarecer melhor quais são algumas possíveis alternativas de Condition, portanto adicionou-se uma nota no Modelo Conceptual que as descreve melhor. A nível de schema de DB, não é possível descrever, e também é melhor descrever as mesmas a nível da Aplicação;
- Embora o ChatGPT apenas tenha avisado para ter cuidado, efetivamente apercebeu-se que o Unique não estava completo. Pois numa Route (de Porto a Hong Kong), há diversas paragens e **dias diferentes**, por exemplo. Corrigiu-se para ser UNIQUE(ROUTE\_ID, SOURCE\_ID, DEPARTURE\_TIME);
- Confirma-se que o pretendido é ser mutuamente exclusive;
- ICAO e IATA tem a restrição esperada e que comprovamos com os dados fornecidos pelo OpenFlights;
- Duplicado do ponto 1;
- Os cascading deletes foram revistos e confirma-se que estão como esperado. Eliminando uma rota, é desejado que deixe de haver o flight e a crew que fez esse voo.

## 5.2. População das Tabelas

À semelhança da criação das tabelas o código que as vai popular também vai estar num ficheiro chamado populate1.sql criado à mão e um populate2.sql que contém informação retirada de <https://openflights.org/data.php>. É deixado aqui um exemplo demonstrativo do código, a encontrar em populate1.sql, do código que vai popular algumas tabelas:

```
PRAGMA foreign_keys = ON;

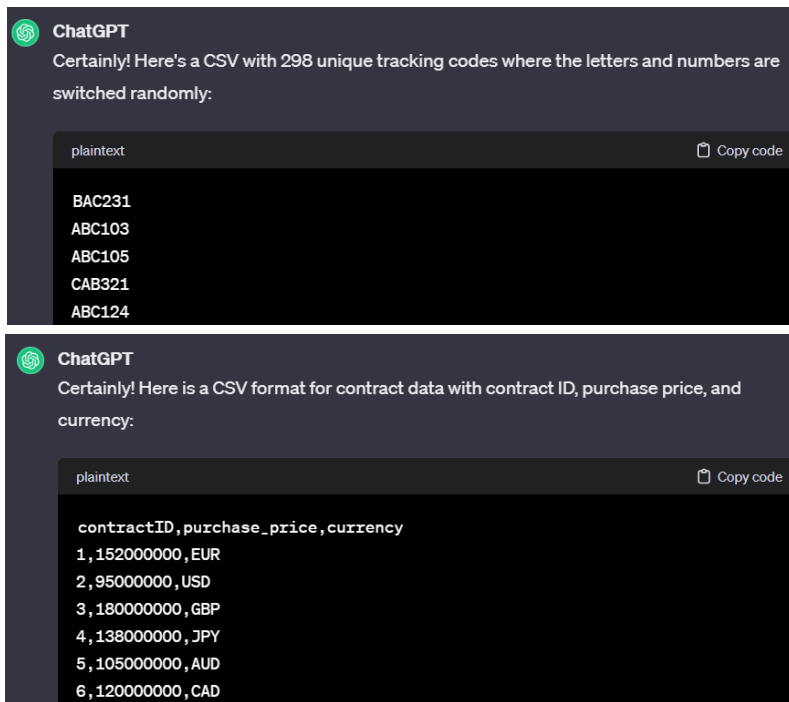
INSERT INTO COUNTRY VALUES ('FR', 'FRANCE');
INSERT INTO COUNTRY VALUES ('PT', 'PORTUGAL');
INSERT INTO COUNTRY VALUES ('GB', 'United Kingdom');

INSERT INTO PLANE_MODEL VALUES ('32Q', 'Airbus A321neo', 'A21N');
INSERT INTO PLANE_MODEL VALUES ('744', 'Boieng 747-400', 'B744');

INSERT INTO PLANE VALUES (1, 1, "ABC123", "32Q");
INSERT INTO PLANE VALUES (2, 1, "DEF456", "32Q");
```

## Usando Inteligência Artificial

Para a população das tabelas, o ChatGPT foi usado para gerar de certos dados como nomes para os crew members assim como as suas funções, geração de códigos aleatórios, por vezes tendo os mesmos que respeitar dependências com outras tabelas. Nas seguintes imagens é possível ver algumas repostas obtidas.



## Referências Bibliográficas

OpenAI. "GPT-3.5 - A powerful language model." OpenAI, 2023. Web. 18 Nov. 2023.

OpenFlights. "OpenFlights Data." Accessed November 18, 2023.

<https://openflights.org/data.php>.

## Participação

Todos os três membros do grupo participaram ativamente na realização do projeto tendo todos dado contribuições necessárias para o projeto.