Report 19/10/2025

Museum Database System, Group 1704

This report describes and explains the construction of a database system for a museum. The goal for our project was to build a database based on a local museum, more specifically the “Museu Nacional Soares dos Reis” in Porto, but throughout the development we decided to add more entities, improving data analysis and quality.

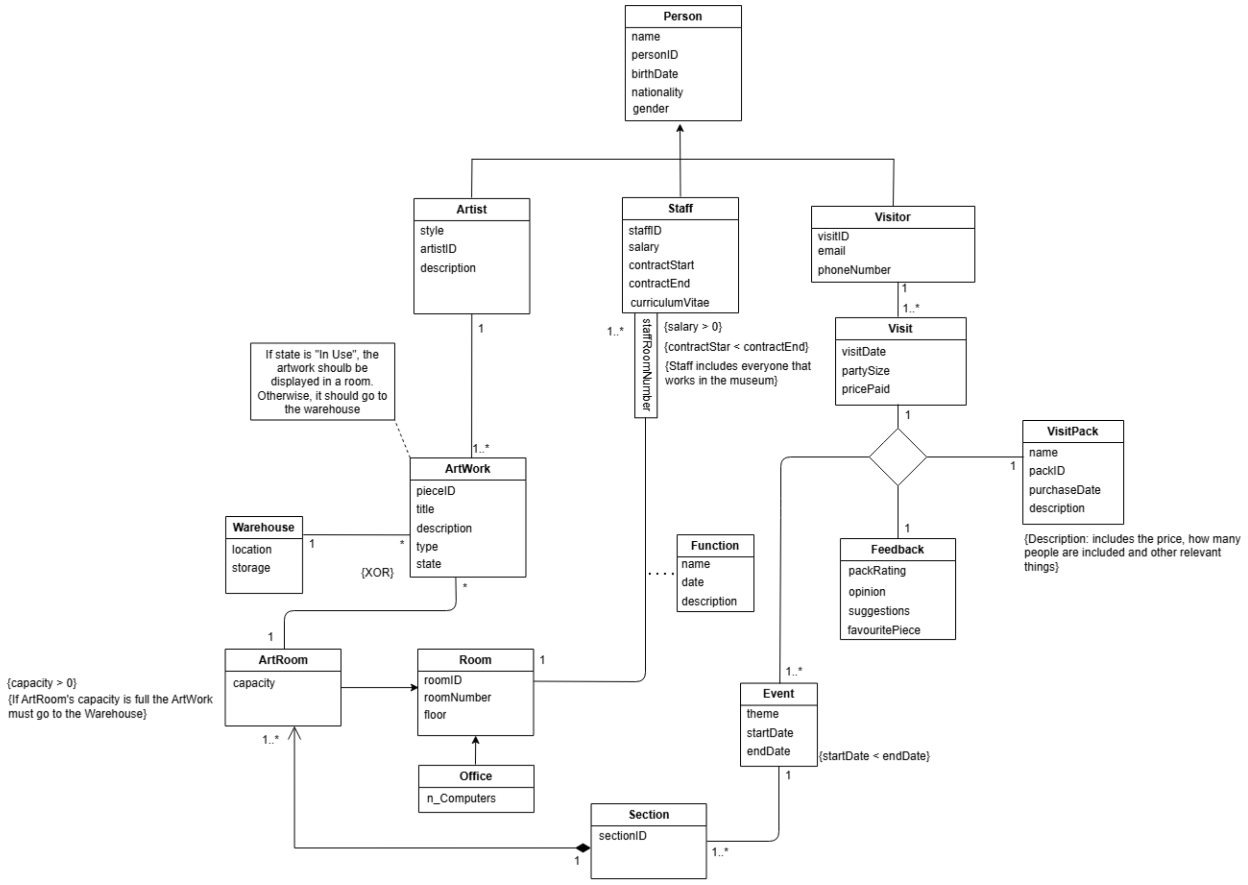
Since we already defined the project domain, let's move on to the data requirements and deconstruction of our museum database. When we think of a museum, the first thing that comes to our mind is probably “Mona Lisa”, which is likely the most famous piece of art ever. Therefore, the museum must store important features about their artworks, including their name and creation background. As a work of art is not born alone, the mind behind the creation must also be presented. So, it is crucial to store information about artists, including their artistic style and relevant background, to properly link artworks to their creators. And, to keep their works clean and correctly disposed of, the staff are essential for managing, maintaining and preserving the museum's collection. Many times, the museum does not have the space to display all the artworks, or some of them need maintenance. So, to prevent that, the museum provides a warehouse for pieces currently not on display and several rooms full of illustrations and sculptures for our guests to visit.

Talking about visiting, our visitors are the pillar of the museum quality, being of our interest to store the date they visit us and if they bring their family or friends. So, to secure the diversity of different pieces we group all the visits in different events, each with a theme and a display period. To separate the events it's important to store the various sections of the museum, where each section has diverse rooms and holds a different event. As a consequence, a visit has a lot of different paths to choose from. And that’s where our museum offers plenty of visit packs, with each variating on price, events and names.

Finally, a good museum is one where people go, enjoy the different styles and pieces and end the journey thinking about coming back. So, for that, and to keep up with people's tastes and various opinions, our job is to improve our functioning and quality by hearing from the source. In this case, the museum has to be able to pile every visitor feedback, where they rate the pack they choose early, their favourite’s pieces and leave a description or a suggestion if they decide so.

In summary, understanding staff’s roles, handling the artworks and enhancing the visitor experience is an essential way that results in a structure that supports data analysis, great for evaluating the requirements, deeper and more precise scope of a database system and optimizing operations for everyone.

Museum Database Model created by us, using draw.io platform.



Museum Database Model created by AI, based on our report.
To do this part, we used two different AI's and gathered all the details that they pointed out about our model in two groups: Strong and Weak points.
The AI's used were <u>Chagpt</u> and <u>Google Gemini</u>.

Strong points:

1 - Correct use of Inheritance with "Person" as a superclass and subclasses "Artist", "Staff" and "Visitor". Avoids redundancy and reflects real-world semantics.

2 - Use of constraints and conditions such as the {XOR} note for artworks being in a room or a warehouse and {startDate < endDate} for events. Strong awareness of data integrity and business rules.

3 - Having "Feedback" and "Visitpack" entities supports user experience analysis and business logic extensions, which make the design richer and more functional.

4 - The correct multiplicity notations and cardinality, such as the composition association, makes this model precise and easy to implement.

Weak points:

1 - The {XOR} is not correct conceptually. The cardinality (*) in the link from "Artwork" to "Room" to "Warehouse" implies that the "Warehouse" cannot be empty, which is incorrect. The same applies to "Room", as the current cardinality (*) does not explicitly state that a "Room" always has, at least, one work of art.

2 - The current cardinality in classes "Feedback" and "VisitPack" are wrong. Right now, a "Visit" can only buy one "VisitPack" which can be false if this visit is a family (bigger group). As a consequence, keeping the cardinality (1) in "Feedback", for a family, it would only be possible to leave one feedback, which is not good if the museum wants to gather information from every client.

3 - There is an attribute in "Feedback" called 'favouritePiece', that represents the visitor's favourite piece. This is a data redundancy problem, because people can spell different names to the same artwork. And also, if the artwork is deleted, this attribute becomes inconsistent.

4 - You have 2 classes, "ArtRoom" and "Warehouse", that have the job of securing the location of an artwork. Since they share their main role, you can create a new class called "Location", where "Location" will be a superclass linked to both "ArtRoom" and "Warehouse" sharing a common attribute 'locationSpot'.

5 - You already wrote {startDate < endDate} for events and contracts, but make sure they are attached to the right entities (Event, Staff, etc.) and not floating notes. Without placement, those constraints are meaningless to UML readers or implementation.

6 - Formally incorrect names in classes and attributes. For example, in "Function" class, maybe "Task" is a better way to qualify what staff does.

Final Museum Database Model with some AI corrections.

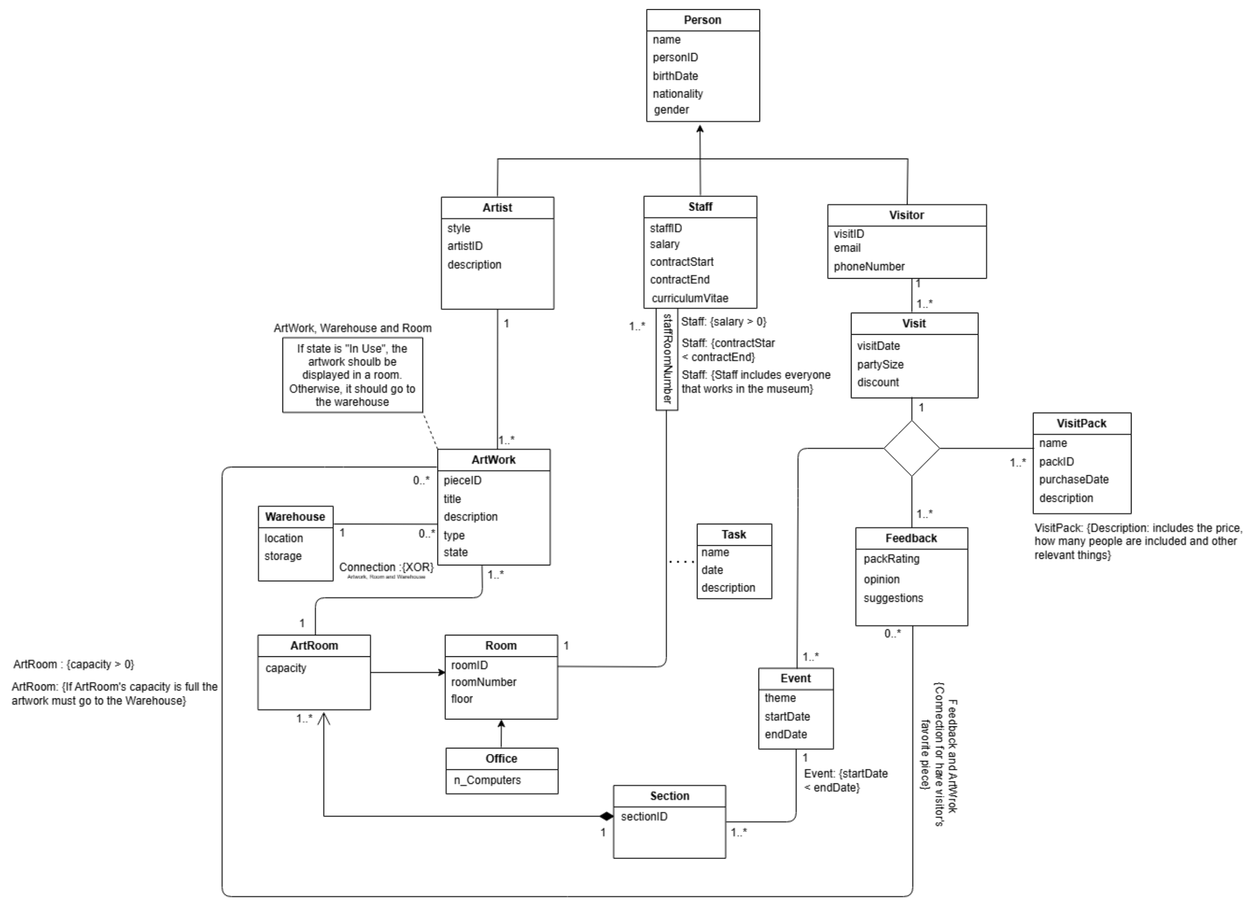
1 - Changed the cardinality between “ArtWork”, “Room” and ”Warehouse”, in these two last classes, from (*) to (0..*) in “WareHouse” and from (*) to (1..*), because a “Warehouse” can be empty and a “Room” has always, at least, one piece.

2 - Changed the cardinality of "Feedback" and “VisitPack”, from 1 to 1..*. A visitor can buy more than one VisitPack at the same time;

3 - We added a connection between “Feedback” and “ArtWork”, because the visitor may or may not specify a favourite piece. It also enables you to get the most often chosen artwork as favourite and if the artwork is deleted, the link stays consistent.

4 - About the superclass “Location”, we think it is a bit unnecessary, as it would only make the model more messy and confusing.

5 - We thought it was better to change the constraints, and now they have the class that they correspond behind them, to place them correctly and make it easier to understand.
Also changed the “Function” name to “Task” and changed the attribute ‘pricePaid’ in “Visit” class to ‘discount’.



Conclusion:

- Url to Initial Model in draw.io :
https://drive.google.com/file/d/1TAw9gA5tupruPjR0oK_DzQCIKcnQ6Fs1/view?usp=sharing
- Url to Final Model in draw.io :
https://drive.google.com/file/d/1CLzcpBGRpzEqA6pte_IXEiYduIdaai40/view?usp=sharing

Group 1704: **19/10/2025**

- Alejandro Zuheros, up202502194
- Rodrigo Pinto, up202403624
- Tiago Rodrigues, up202403626