Faculdade de Engenharia da Universidade do Porto

Conceptual Modeling

Building a database for a fashion event

(First Submission)

Database Project 2024/25 - L.EIC

Group 707

Students & Authors

Leonor Matias up202303772@up.pt

Matilde Nogueira de Sousa up202305502l@up.pt

Gabriela de Mattos Barboza da Silva up202304064@up.pt

20 de Outubro de 2024

Resumo

This document presents the UML diagram our group developed for managing a high-fashion runway event, alongside an in-depth explanation of the entities, relations involved, and our reasoning and methods used for organizing the diagram during the initial stage of database development.

Additionally, we discuss the impact of AI input on our work after submitting it for feedback, and whether or not we decided to implement its suggestions.

Keywords: Database, UML Diagram, Fashion, Conceptual Modeling

Index

I. Introduction

As a group of friends, we embraced this project as an opportunity to let our common interests shine through our academic responsibilities. From the outset, we wanted the theme to be beauty or fashion related. After some deliberation, we settled on creating a database for a high-fashion runway event, analogous to the well-known *Fashion-Week*, iconic and indispensable to the glamorous world of fashion.

II. Context Description

High fashion runway shows, often held semi-annually in major cities like New York, Paris, London, and Milan, serve as a pivotal platform for showcasing new collections from top designers and brands, attracting media attention, and significantly shaping trends and fashion culture. The complexity and dynamic nature of these shows make it necessary to have a well-structured system, in order to organize and manage various aspects, highlighting relationships and interactions among different entities. To put it briefly, key aspects include designers presenting unique pieces and collections, models showcasing them on the runway, technicians ensuring smooth execution, and spectators attending the event.

Main Entities

Designers

Designers are important, as they create the fashion pieces and consecutively, the collections to be further displayed. The system tracks their personal and professional details, such as name, nationality, and contact information. Each designer can create multiple pieces, which are then associated with a collection.

Collections

Fashion collections, defined by the brand and season, contain multiple pieces. Each piece is described by material, color, and type, aiding in managing inventory and tracking trends.

Brands

Brands, which are associated with collections, shape the showcased designs and have attributes like origin, representative, and logo, maintaining their identity during events.

Runway Show

Runway shows are key events where these collections are displayed. Each brand holds a single runway show at each edition of the event.

Models

For models, the system keeps track of their personal data and links them to the runway through a class “Model_Runway”, that discloses specifically which model is participating in each runway, avoiding duplicate data. Each model can participate in multiple runway shows, and a runway show features more than one model. Because of this, we figured creating this class would be the best way to deal with this many-to-many relationship. This class is further refined with an attribute that indicates the order of each model’s appearance on the runway.

Event

The diagram focuses on the Event, defined by location, organizing entity and budget. Sponsors are critical for financial support, and the system tracks their contributions, agreements, and resources. The relationship between an event and its sponsors is one-to-many, as an event typically has multiple sponsors. However, whether sponsors support other events is irrelevant to the database and not tracked.

Supporting Entities

The system also manages various support roles. Styling and local technicians provide assistance with models styling and technical aspects of shows, using their skills and equipment to ensure smooth operations.

Ticketing is managed through tracking ticket types and statuses, through the class “Runway_Espectator”, helping monitor spectator attendance. Influencers and media professionals, as part of the audience, are tracked by their reach, like social media followers and platforms they’re working for (e.g., TV, magazines).

III. UML Model (1st Approach)

Our initial model was developed based on our preliminary brainstorming for the project, without the involvement of AI.

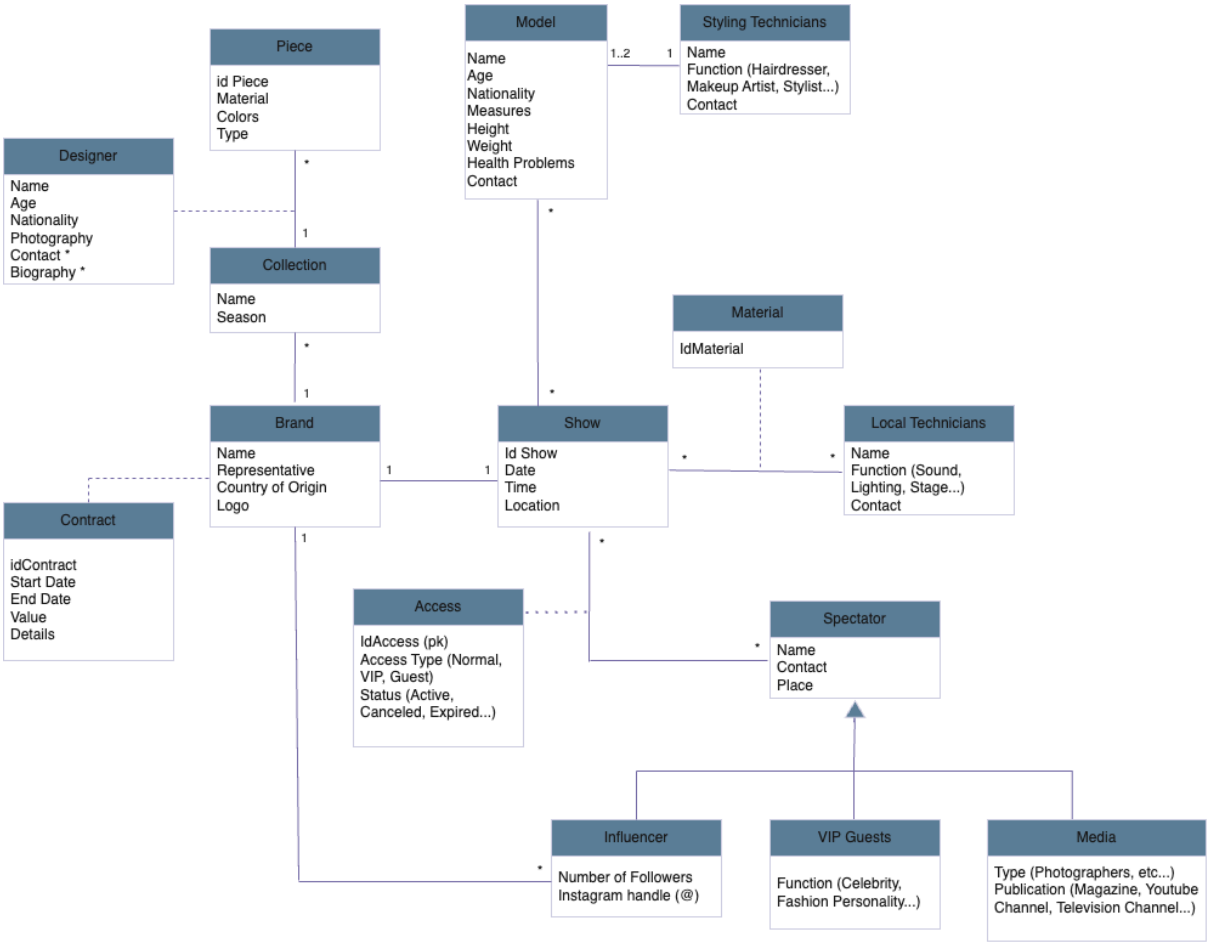

Image 1 - Initial UML Model

Runway - Brand / Brand - Collection

The relationship between the runway show and the brand is structured such that each show represents a brand, and each brand has multiple collections that vary by season. With that said, we have established a one-to-one relation between the brand and the show, and a one-to-many relation between the brand and the collection.

Collection - Piece

Each collection is composed of various pieces, with designers responsible for creating these pieces. A collection can thus involve contributions from several designers. Therefore, the relationship between the brand and the collection is a many-to-one. Additionally, an association class called *Designer* is introduced to represent the designers responsible for creating the pieces within each collection.

Runway - Model / Model - Styling Technicians

The relation between *Runway* and *Model* is based on the fact that a single fashion show features multiple models, and a model may participate in more than one show. Additionally, styling technicians are responsible for preparing models for the show. Each technician is assigned to one or two models, and a model may have multiple technicians associated with her, as different technicians specialize in various areas of styling.

As a result, we establish a many-to-many relationship between the fashion show and the model as well as a many-to-one relationship between the *Model* and the *Styling Technicians*.

Runway - Spectator

As for access to the fashion show, there are several access types (e.g., VIP, general admission, and guest), which determine the type of spectator attending. The class *Spectators* has some subclasses, such as *Influencers* and *Media*, as they have distinct characteristics compared to other attendees. Furthermore, influencers are related to the brand, as they attend the fashion shows invited by a specific brand, with the nuance that they're usually hired to create content regarding the event and brand they were invited by. (One of the reasons we created a class *Contract*: This class serves to represent the contractual agreements between the brand and various stakeholders involved in the fashion show, including designers, influencers, and technicians).

As a result, we establish a many-to-many relationship between the fashion show and the spectator, with an association class called *Access* to represent different types of entry. Additionally, we defined *Influencer*, *VIP Guest* and *Media* as subclasses of *Spectator.*

IV. AI Feedback

The AI tool we chose to assist us with this project GPT-4, developed by OpenAI. We presented the first draft of our UML model and asked for revisions and suggestions for any relevant changes. Some of the changes it suggested are as follows:

Many-to-Many relations

Firstly, ChatGPT told us to avoid many-to-many relations, which were a serious concern in our UML diagram draft. We acknowledged this as a bad practice and, to ensure correctness, we created new classes (such as *Model_Runway*) that act as a proof method for managing repeated information.

Discard Spectator Class

It also proposed that we discard the class *Spectators*, saying it was an unnecessary generic class, and we could just replace it directly by its specific types like Influencer, Media,

Frontline, etc. We chose not to take this suggestion, as we found it helpful to use generalization to improve readability.

New Class “TicketStatus”

Additionally, ChatGPT recommended adding a new class to specify the status of the show’s tickets, referring to things like "Paid," "Reserved," or "Canceled," but we considered this a deviation from our main goal, which is not ticket management. We figured that keeping the status as an attribute was enough.

Surrogate Keys for all entities

Finally, it suggested we add surrogate keys to all entities. We found this unnecessary, as there are classes (e.g. Model), where just the attribute name should be unique enough to identify them, given that our project focuses exclusively on the high-fashion world.

V. Our final solution

This is the final UML Diagram improved after taking some of the suggestions made by AI, as well as some optimisations made fully by us.

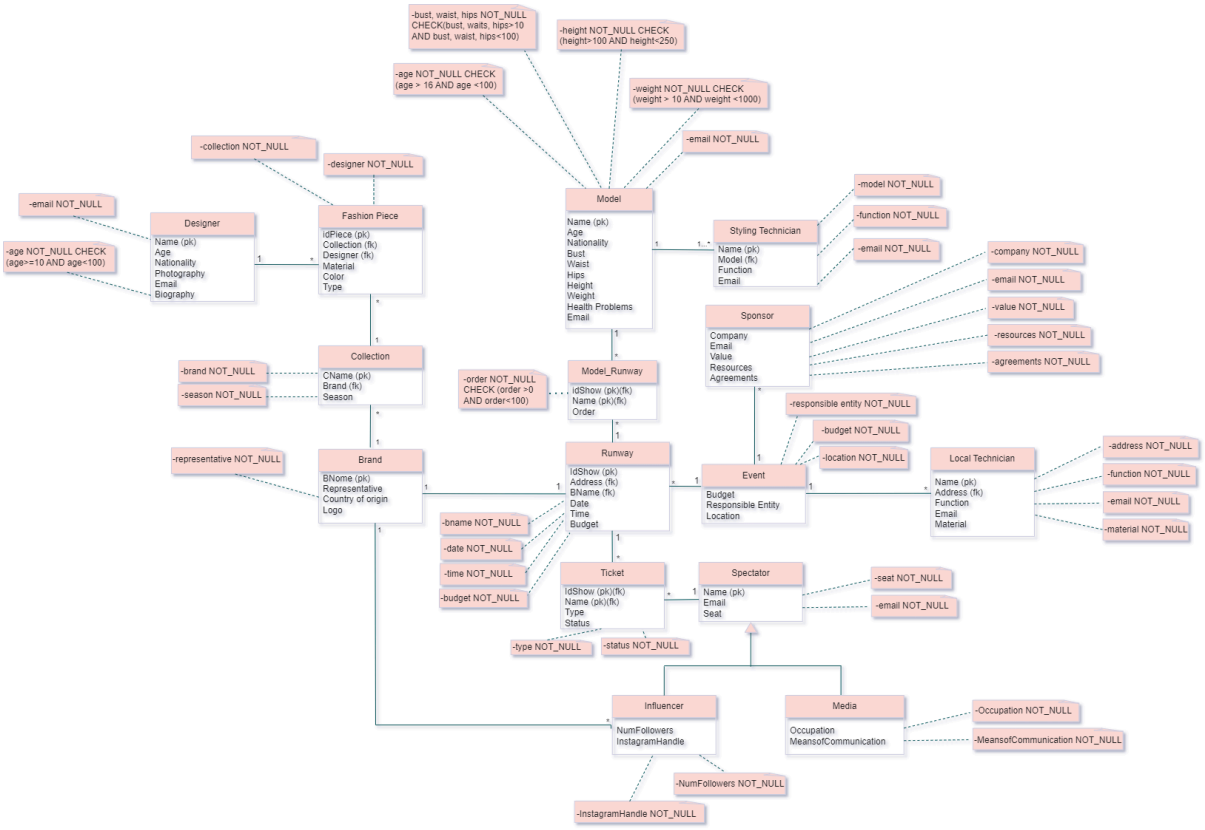

Image 2 - Final UML Diagram

AI Suggestions

Runway - Model / Runway - Spectator

As discussed in the previous section (IV. AI Impact - Many-to-Many relations), we added the class *Model_Runway* to eliminate the many-to-many relationship we had between *Model* and *Runway*. This approach helps to clearly define the connections between models and fashion shows, ensuring a more organized structure. It also allows us to identify specifically what model is performing on what runway, avoiding duplicate data.

We also omitted the *Access* association class and added a *Ticket* class, as it appeared to make more sense for the overall structure and functionality of the diagram. Therefore, we establish a many-to-one relationship between the fashion show and the ticket, as each show can have multiple tickets associated with it. Additionally, we establish a many-to-one relationship between the ticket and the spectator, meaning that each ticket is linked to a specific spectator, while a spectator can hold multiple tickets for different shows.

Other Improvements

By working on the database, and getting more immersed both in our theme and in database design, we made some optimisations “by our own”, regardless of the suggestions given by AI. We also added constraints to each class directly in the diagram.

Runway - Event / Event - Sponsor

We created the *Event* class to represent the actual event (with multiple runway shows) rather than just the *Runway*. We then linked it to the *Runway* class through a many-to-one association, as the fashion week hosts multiple shows. Additionally, we introduced a *Sponsor* class, which is linked to the *Event* class through a many-to-one relationship, reflecting that each event can have multiple sponsors.

We also removed the *Material* association class and incorporated it as an attribute into the *Local Technicians* class, as we believe this approach enhances the fluidity of the conceptual modeling.

Designer - Piece

Furthermore, we established a many-to-one relationship between the *Designer* class and *Piece*, rather than it being an association between the pieces and the collection. This change clarifies that multiple pieces can be designed by a single designer, streamlining the representation of the design process within the model. We also discarded the *Contract* association class linked to the *Brand*, as we determined it to be unnecessary information for our goal with this model.

Spectator - Influencer / Spectator - Media

Finally, we removed the *VIP Guest* class as a subclass of *Spectator*, as it does not possess enough unique characteristics to warrant a specific subclass. Consequently, we retained only *Influencer* and *Media* as subclasses of *Spectator*, since they have distinct attributes that differentiate them from other attendees.