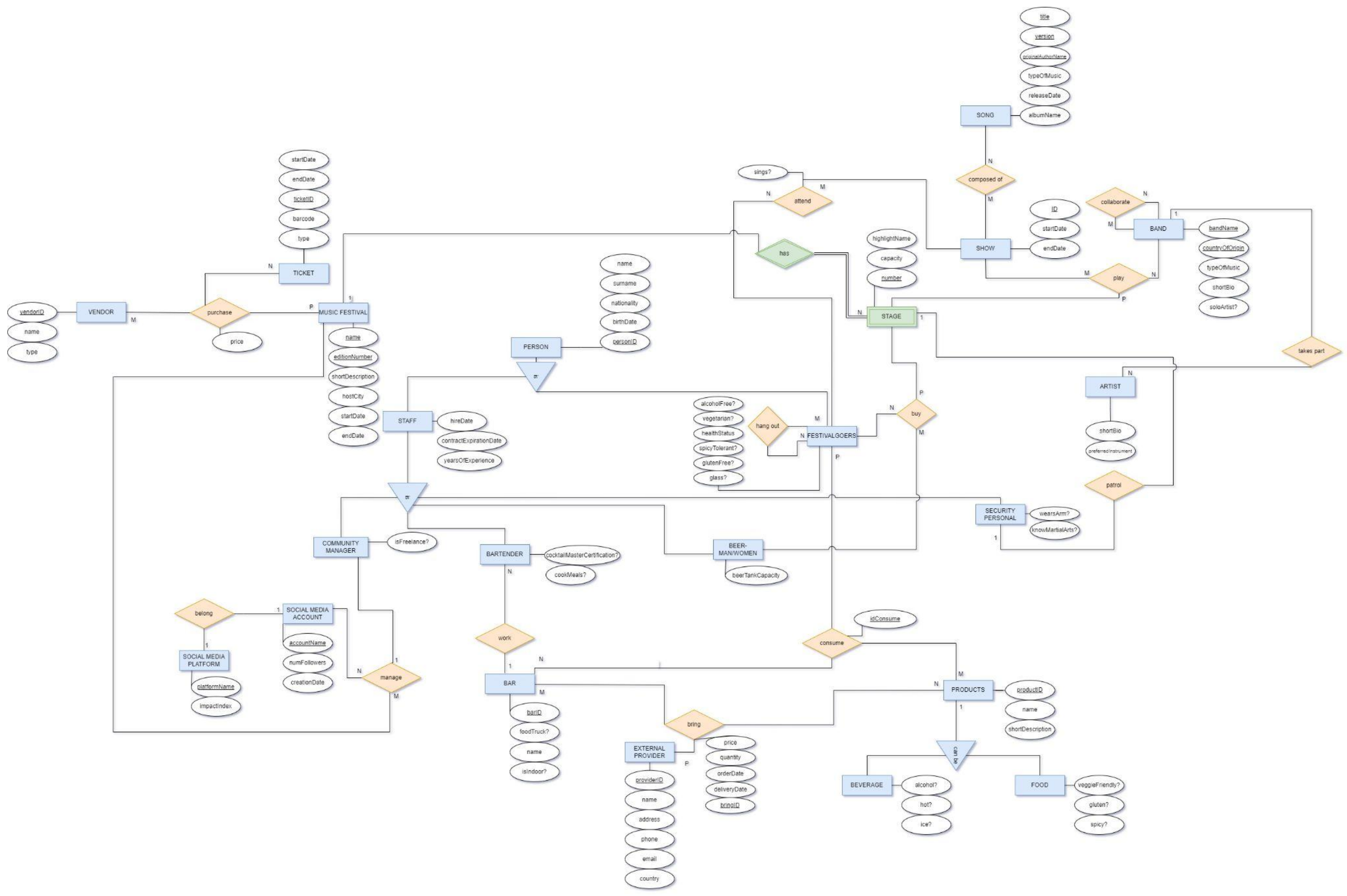
The Conceptual and Relational Model

Maria Beili Mena Dorado (u199138)
Ariadna Prat Bosch (u185150)
Mireia Pou Oliveras (u198721)

Explanation of the justification of assumptions and decisions made.

Model Conceptual



We designed the model conceptual by adding 22 entities with their respective attributes and relations with some of them.

For the entities, we have assigned “Stage” like a weak entity that participates in a relationship with a strong entity, “Music Festival”, because the stage numbers can repeat in

different music festivals, making it necessary to know in which music festival the stage is located. Furthermore, we have established for every one of them a primary key: ID or name; and for others (such as Music Festival, Song, Band, etc.) composite attributes as PK.
For the relationships, we have structure:

- **Binary Relationships:**
    - Music Festival — <has> — Stage, cardinality [1:N]. Every festival will be placed with some stages on it.
    - Festivalgoers — <attend> — Show, cardinality [N:M]. We need to register which festivalgoers are going to see which concerts. This relationship has an attribute, called sings, that denotes if a festivalgoer is singing or not in the Show.
    - Show —<composed of> — Song, cardinality [N:M]. One show is composed of lots of songs and one song can be played in many shows.
    - Artist —<takes part> — Band, cardinality [1:N]. We have considered that one band has several artists.
    - Security Personal —<patrol> — Stage, cardinality [1:1]. A security personal is assigned to a specific stage.
    - Social Media Account —<belong> — Social Media Platform, cardinality [1:1]. We just need a social media account on a platform. It is not necessary to have a lot of social media accounts in relation to music festivals on the same platform. Moreover, we cannot have the same social media account on different platforms.
    - Bartender — <work> — Bar, cardinality [1:N] because one bar has several bartenders and one bartender works in one specific bar.

- **Generalization**:
    - Person → <is> →[Staff, Festivalgoers, Artists]
    - Staff →<is> → [Bartender, Beer-man/woman, Security Personal, Community Manager]
    - Product → <can be> → [Beverage, Food]

  We have made a generalization with the name of the relationship 'is' and 'can be', where the common attributes and relationships are placed and connected to the high-level entity Person, Staff and Product.
  Lower-level entities only have their specific attributes and relationships [Staff, Festivalgoers, Artists], [Bartender, Beer-man/woman, Security Personal, Community Manager], [Beverage, Food]. They inherit all their "parent" entity has.

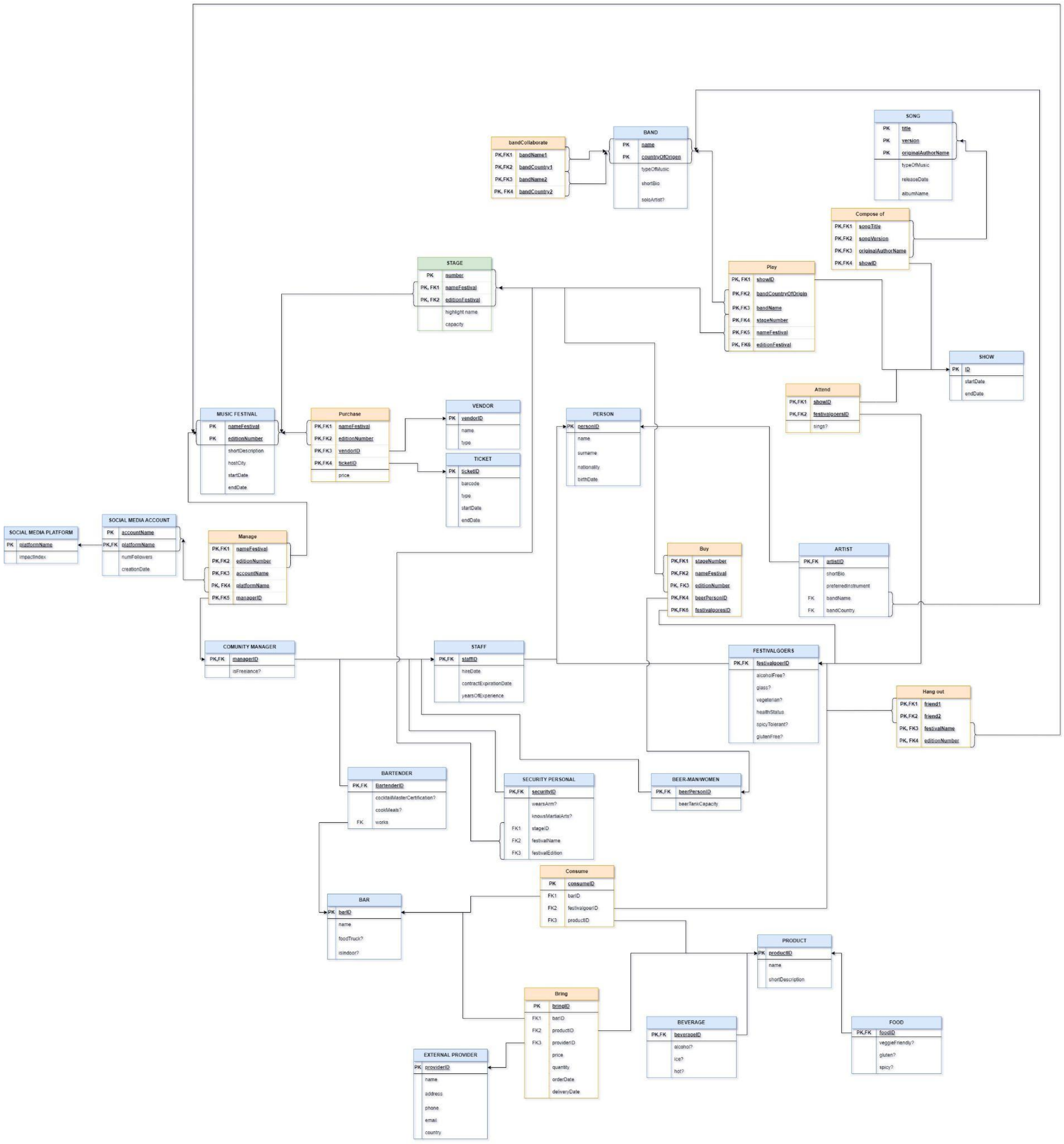
- **Ternary relationship**:
    - bring — [Bar, External Provider, Products] with cardinality [N: M: P], because we need to keep a record of who brings the consumables, where they are brought, and which ones are. Every delivery has a unique ID.

  - consume — [Bar, Products, Festivalgoers] cardinality [N: M: P]
    We need to know which festivalgoers consume what products in a specific bar, because not all the consumables are in the same bar. This relationship has a attribute idConsume because we want to keep track of the IDs.
  - buy — [Beer-man/woman, Festivalgoers, Stage] with cardinality [M:N:P]. Multiple festivalgoers buy multiple beers on multiple stages. Therefore, we need to record which festivalgoer buys which beer on which stage by some beer-man/woman.
  - purchase — [Music Festival, Vendor, Ticket] with cardinality [N:M:P], because a vendor can sell various tickets for different kinds of music festivals. In addition, tickets can be sold by various vendors. This relationship has an attribute price because we want to know what it costs.
  - manage — [Community Manager, Social Media Account, Music Festival] with cardinality [N:M:1] because a manager will manage several social media accounts for several festivals at the same time.
  - play — [Stage, Show, Band] with cardinality [N:M:P]. Bands play their shows on stages. So, this relationship allows us to register what show is playing the band and where.

- **Recursive relationship**:
  - Festivalgoers — <hang out>
  - Band — <collaborate>

  We want to represent the hierarchy in a music festival where the Festivalgoers can go with some friends. So, conceptually anyone who goes to a music festival is a Festivalgoer.
  The same way, we can say that a band can collaborate with another one. Therefore, a collaborative band is considered a Band in the music festival.

Model Relational



All the entities, with their respective attributes are represented in tables. The PKs are bold and all the FKs have arrows pointing to the entity referenced. All the relationships 1:1, 1:N are considered as attributes inside the (N) entity. An example of this is Band - Artist → inside Artist we have an attribute FK of band, and another example is Music Festival - Stage → inside Stage we have the music festival where it is located.

All the other attributes have been placed as new tables. The tables which we created are:

Manage: It has the primary key of a music festival (festival_name, festival_edition) because we need the information about what account name has relation with the music festival. The account can be different depending on the festival. The primary key of social media accounts and platform account (account_name, platform_name) are important because we need the information about the community manager who has access to every account.

Belong: We don't put this relation because we can't put in the case of cardinality [1:1].

Purchase: When we purchase a ticket from any vendor, we need the next information. The festival_name and the festival_edition because a festival can be created in many years. Also we need the id of the vendor and the ticket because a ticket can be sold by many vendors and a vendor can sell various types of tickets. Moreover, we need the price of the ticket.

Work: Is an attribute named works in the entity bartender in this attribute we keep id_bar.

Bring: We need the information about the bar, product and external provider. If we want to get this information, we should have the ID's. Moreover, we need the price and quantity of the product. Also, the deliveryDate and orderDate when the bar asks the product. This relation is identified by a unique ID → bringID.

Consume: We need the information about the bar where the festivalgoer consumes a product. Also, the information about the product and who is the festivalgoer. We need just the ID and create an IDConsume in relation to the buy of the product.

Patrol: Is an attribute of security_personal. In this petrol, we keep the number of the stage where the security_personal is looking out.

Attend: We use the ID of festivalgoer and show. Because we need the festivalgoers who will attend the show. In addition, we need to know if the festivalgoer sings or not in the show.

Compose of: In this relation, we get a set of songs that will be sung in the show. We need the primary key of the songs (songTitle, songVersion, originalAutorName). Moreover, we need the ID of the show.

Has: In the relational model, "has" is ignored. Directly, we relate the stage with a music festival. In addition, the stage has enough information about the music festival with his primary key.

Collaborate: This relation in relational model is named bandCollaborate. We need the main information about the collaboration of two bands. We get the information about these two bands with the bandName and bandCountry of every band.

Play: In this relation, we have the next information about music festival(festival_name, edition_number), band(bandName, bandCountry), show (id_show) and stage (id_stage).

Takes part: We delete this relation in relational model and we do a direct relation where every artist has attributes such as attribute the band_name and band_country.

Buy: In this relation, we keep the information about a festivalgoer buying a beer to a beerman/beerwoman in a specific stage. In this relation, we need the ID's.

Hang out: In this relation, we have the ID of two festivalgoers. Also, the festival_name and festival_edition. The two festivalgoers are related and they enjoy being together.