Proposed Entity-Relation Diagram for MayAztec project

Profesor: Esteban Castillo Juárez

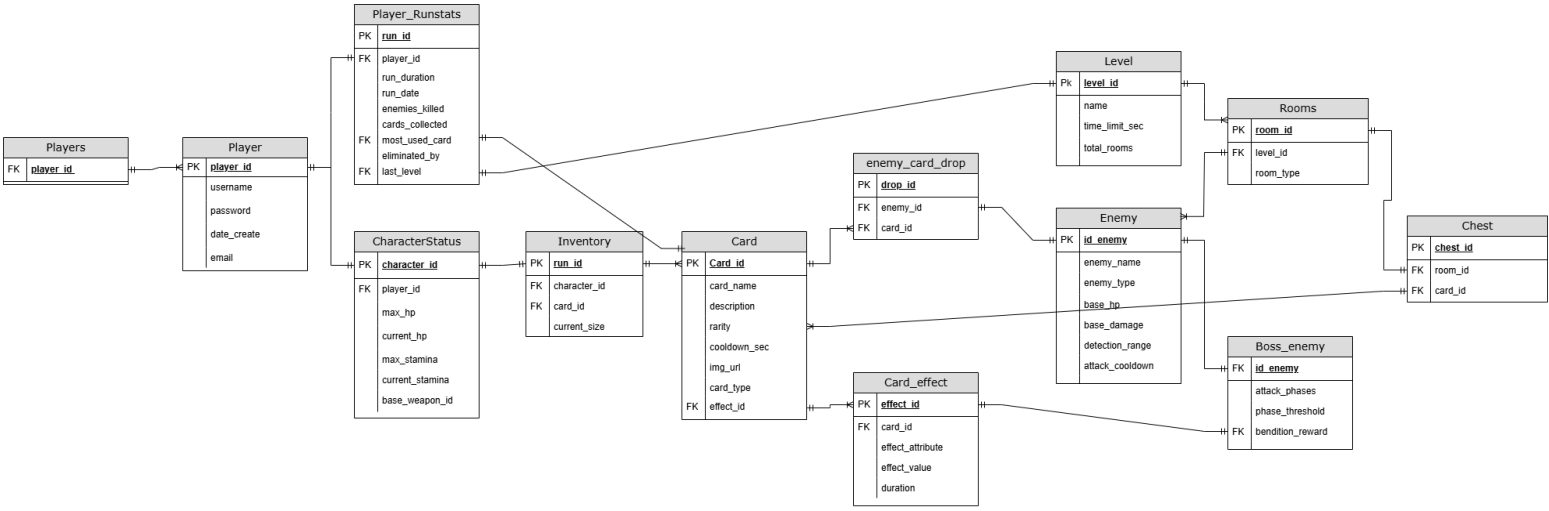
Materia: Construcción de Software para la toma de decisiones

Nombres:

- Mauricio Emilio Monroy González - A01029647
- Héctor Lugo Gabino - A01029811
- Nicolás Quintana - A01785655

Fecha: 14 de marzo del 2025

Justification



In our proposed ER model for our game MayAztec, we proposed 13 Tables with no intermediate tables.

Main entities and their attributes:

Players

- This table will store the information of all the players.
- Relations: Players → Players One-to-Many:
    - Players store the information contained in the Player table, and all the details related to the players.

Player

- This table manages user account information.
- Attributes:
    - player_id (int auto_increment) PK
    - username (varchar)
    - password (varchar)
    - date_create (time_stamp)
    - email (varchar) UK
- Relations:
    - One-to-one with ChatacterStatus:
        - The separation between Player and CharacterStatus allows the account management system to be decoupled from the game state. This makes it easier to restore character statues.

    - One-to-one with Player_Runstats:
        - This relationship allows multiple games to be registered for the same player, implementing the “runs” system, characteristic of roguelites.
        - **Constraints**: the foreign key player_id in Player_Runstats ensures that each statistic belongs to a specific player.

CharacterStatus

- Purpose**:** Store in-game character attributes (Hp, stamina, base weapon) associated with a player.
- Attributes:
    - character_id (PK) auto_increment
    - player_id (FK) fk (Player)
    - max_hp (tiny int)
    - current_hp (tiny int)
    - max_stamina (tiny int)
    - current_stamina (tiny int)
    - base_weapon_id (int) foreigkey to card
- Relations:
    - One-to-one with Player
    - One-to-one with Inventory
        - This is key as this limits the character to just one inventory, not allowing more than the 6 cards in their inventory

Player_Runstats

- Purpose**:** Record the information of each “run” or game attemp. The roguelite is bassesd in successive runs.This table keeps a history of each run analysis.
- Attributes
    - run_id (int) PK
    - player_id (int) FK
    - run_duration (int)
    - run_date (datetime)
    - enemies_killed (tiny int)
    - cards_collected (tiny int)
    - most_used_card (int) FK to card
    - eliminated_by (varchar name of the enemy)

    - last_level (int) FK to level
- Relations:
    - One-to-one with Player
    - One-to-one with Card
    - One-to-one with Level

Inventory:

- Purpose: Store the card that the character has in his “deck” or available slots.
- Attributes:
    - run_id (int) PK
    - character_id (int) FK to CharacterStatus
    - card_id (int) FK to Card
    - current_size (tinyint)
- Relations:
    - One-to-one with CharacterStatus
    - One Mandatory-to-Many Mandatory with Card

Card

- Purpose: To model each lottery card. The games revolve around the cards, these are the power ups of the adventure.
- Attributes:
    - Card_id (int) PK
    - card_name (varchar)
    - description (varchar)
    - rarity (enum(“low_tier, high_tier))
    - cooldown_sec (smallint)
    - img_url
    - card_type (enum(“weapon, transformation, buff))
    - effect_id (int) FK to Card_effect
- Relations:
    - Many Mandatory-to-One Mandatory with Inventory
        - Allows many cards (5 plus base card weapon) to be attributed to only one inventory
    - One Mandatory-to-Many Mandatory with enemy_card_drop
    - Many Mandatory-to-One Mandatory with Chest

    - One Mandatory-to-Many Mandatory with Card_effect:
        - This structure is crucial to implement the Lottery Card system. A card can have multiple effects (damage, transformation, health, stamina), which makes it possible to create complex cards such as transformations that modify several statistics simultaneously.
        - Constraints: The card_effect uses effect_id as a unique identifier for each effect, relating to card_id as a foreign key, allowing a card to have multiple independent effects.

Annotations:

- img_url: This field stores the path to the image that represents the chart graphically in the game.

enemy_card_drop

- Purpose: to define the probabilities of an enemy dropping a specific card.
- Attributes
    - drop_id (int) PK
    - enemy_id (int) FK to Enemy
    - card_id (int) FK to Card
- Relations
    - Many Mandatory-to-One Mandatory with Card
    - One-to-one with Enemy
        - Ensure that the enemy has the chance to drop the correct card (Mariachi enemy has chance to drop mariachi card and no other card)

Card_effect

- Purpose: Allow each card to have multiple effects. Makes complex cards easy to model.
- Attributes
    - effect_id (int) PK
    - card_id (int) FK to Card
    - effect_attribute (varchar)
    - effect_value (int)
    - duration (int -seconds)
- Relations
    - Many Mandatory-to-One Mandatory with Card_effect

- One-to-one with Boss_enemy
  - Gives flexibility for complex cards (e.g., Diablo: +3HP, +2 dmg, etc.).
  - Normalizes the relation and avoids null fields in Card.

Enemy

- To store the data of each enemy. Centralizes the common information of all enemies.
- Attributes:
  - id_enemy (int) PK
  - enemy_name (varchar)
  - enemy_type (enum(“medium”, “heavy”, “boss”))
  - base_hp (tinyint)
  - base_damage (tinyint)
  - detection_range (smallint)
  - attack_cooldown (float)
- Relations:
  - One-to-one with enemy_card_drop:
    - This relationship implements the probabilistic drop system. Each enemy can drop several cards with different probabilities.
    - **Constraints:** Foreign key ensure referential integrity.
  - One-to-one with Boss_enemy:
    - This is a specialization relationship where bosses are a special type of enemy with additional attributes. It allows reusing common attributes while adding specific characteristics.
    - **Constraints**: the **primary key** of Boss_enemy is also a **foreign key** that references Enemy, ensuring that only some enemies are bosses.
  - Many Mandatory-to-One Mandatory with Rooms

Annotations:

- detection_range: Defines the distance at which the enemy can detect the player, crucial for AI systems and enemy behavior.
- attack_cooldown: Represents the time the enemy must wait between attacks, balancing the difficulty of the game.

Level

- Purpose: define game levels. Is a way to persist the configuration of each level required.
- Attributes
    - level_id (int) PK
    - name (varchar)
    - time_limit_sec (float)
    - total_rooms (tinyint)
- Relations:
    - One-to-one with Player_Runstats:
        - In a level a set of statistics is generated to the player, and that is unique for that run.
    - One Mandatory-to-Many Mandatory with Rooms:
        - One level contains many rooms

Annotations:

- time_limit_sec: This field sets a time limit to complete the level, adding pressure to the player and an additional element of challenge.
- total_rooms: Defines the number of rooms in a level, allowing to generate levels of different complexity.
- room_type: Classifies the rooms (combat, boss), allowing to create varied and progressive experiences.

Rooms

- Purpose: store rooms whitin the level, each room its own room-type. A level can have multiple rooms. This facilitates the generation of dungeons and the assignment of enemies to each room.
- Attributes
    - room_id (int) PK
    - level_id (int) FK to Level
    - room_type (enum(“normal”, “boss”))
- Relations:
    - Many Mandatory-to-One Mandatory with Level
    - One Mandatory-to-Many Mandatory with Enemy
    - One-to-one with Chest

Boss Enemy

- Purpose: Separates to empathize with specialization form a normal enemy to one with boss attributes.
- Attributes
    - id_enemy (int) FK
    - attack_phases (tinyint)
    - phase_threshold (int)
    - bendition_reward (int) FK to Card_effec
- Relations
    - One-to-one with Enemy
    - One-to-one with Card_effect:
        - Boss_enemy and bendition_reward: The relationship with Card_effect, the boss reward is a special effect applied to a card, instead of just a new card, it is a card that gives you a permanent effect when you use it. Inside card attributes.

Chest

- Attributes: represent chests in the rooms, indicating which card they contain.
    - chest_id (int) PK
    - room_id (int) FK to Rooms
    - card_id (nt) FK to Card
- Relations:
    - One-to-one with Rooms
    - One Mandatory-to-Many Mandatory with Card
        - This is a treasure system that contains cards as rewards, if a chest appears there can only be one per room. The relationship with Room allows you to place specific treasures in specific rooms, while the relationship with Card defines which cards can appear in the chests.