Wireframing Relational Data for an MTG Deck Builder

By Nahid Sekander

Project Idea

I am aiming to create a system that manages players, decks, and cards. Being a fan of Magic: The Gathering, the website will allow users to create different decks by adding cards from a list of MTG cards through a database. Users can also view which cards belong to specific decks. The admin can update information about players, decks, and cards.

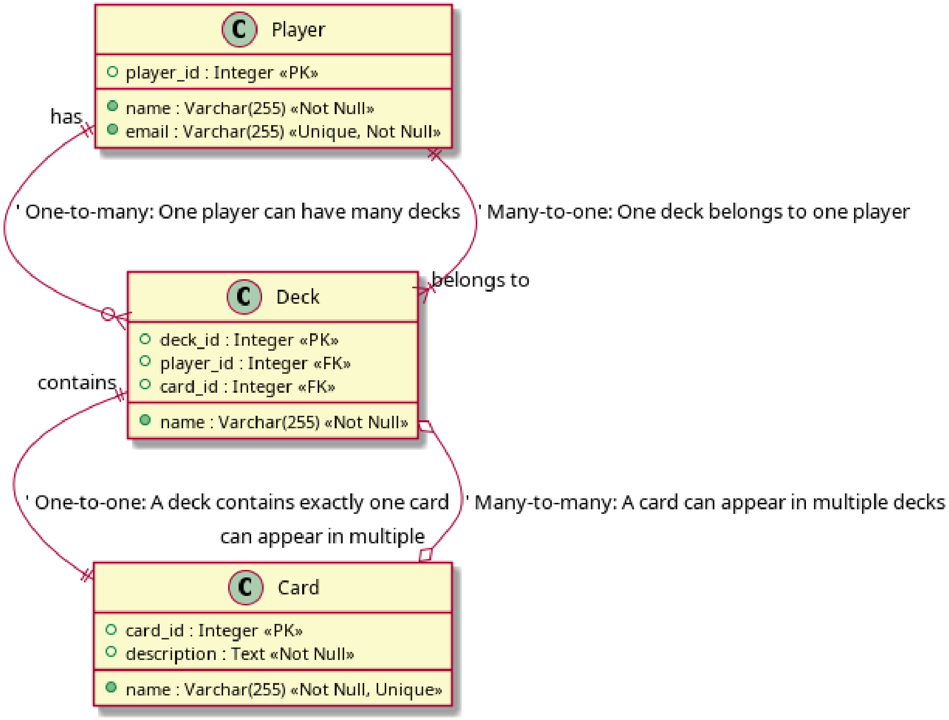
Relational Database

Player to Deck: A **Player** can have many **Decks** (one-to-many relationship).

Deck to Player: A **Deck** belongs to one **Player** (many-to-one relationship).

Deck to Card: A **Deck** contains exactly one **Card** (one-to-one relationship).

Card to Deck: A **Card** can appear in multiple **Decks** (many-to-many relationship).

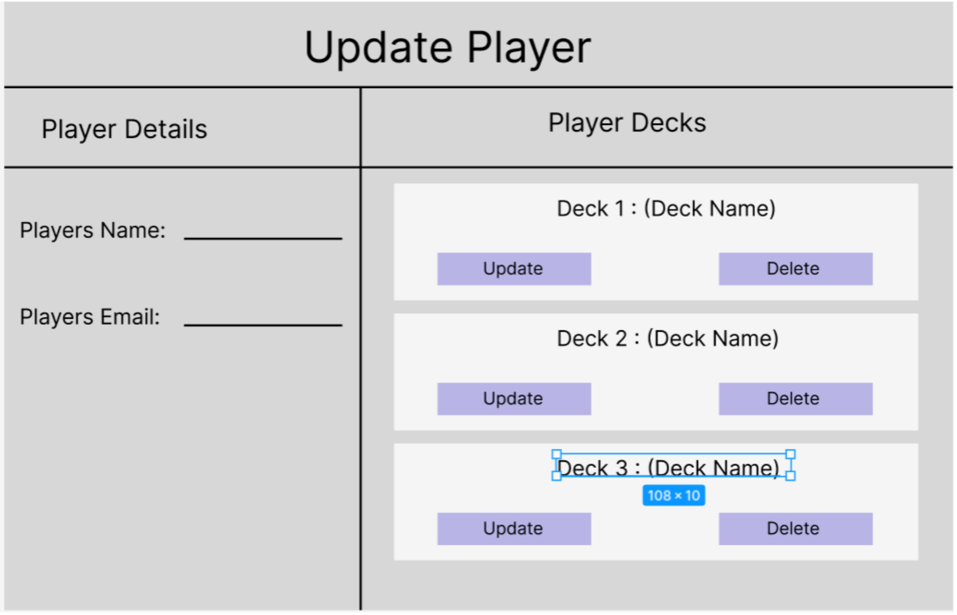

C Player
○ player_id : Integer «PK»
● name : Varchar(255) «Not Null»
● email : Varchar(255) «Unique, Not Null»
has
' One-to-many: One player can have many decks
' Many-to-one: One deck belongs to one player
belongs to
C Deck
○ deck_id : Integer «PK»
○ player_id : Integer «FK»
○ card_id : Integer «FK»
● name : Varchar(255) «Not Null»
contains
' One-to-one: A deck contains exactly one card
' Many-to-many: A card can appear in multiple decks
can appear in multiple
C Card
○ card_id : Integer «PK»
○ description : Text «Not Null»
● name : Varchar(255) «Not Null, Unique»

- **Player Entity**:
  - `player_id` is the **Primary Key** (`PK`), an integer.
  - `name` is a **Varchar(255)** and **Not Null**.
  - `email` is a **Varchar(255)**, **Unique**, and **Not Null**.
- **Card Entity**:
  - `card_id` is the **Primary Key** (`PK`), an integer.
  - `name` is a **Varchar(255)**, **Unique**, and **Not Null**.
  - `description` is a **Text** field, **Not Null**.
- **Deck Entity**:
  - `deck_id` is the **Primary Key** (`PK`), an integer.
  - `name` is a **Varchar(255)**.
  - `player_id` is a **Foreign Key** (`FK`) pointing to the **Player** table.
  - `card_id` is a **Foreign Key** (`FK`) pointing to the **Card** table, ensuring that each deck contains exactly one card.

Wireframes

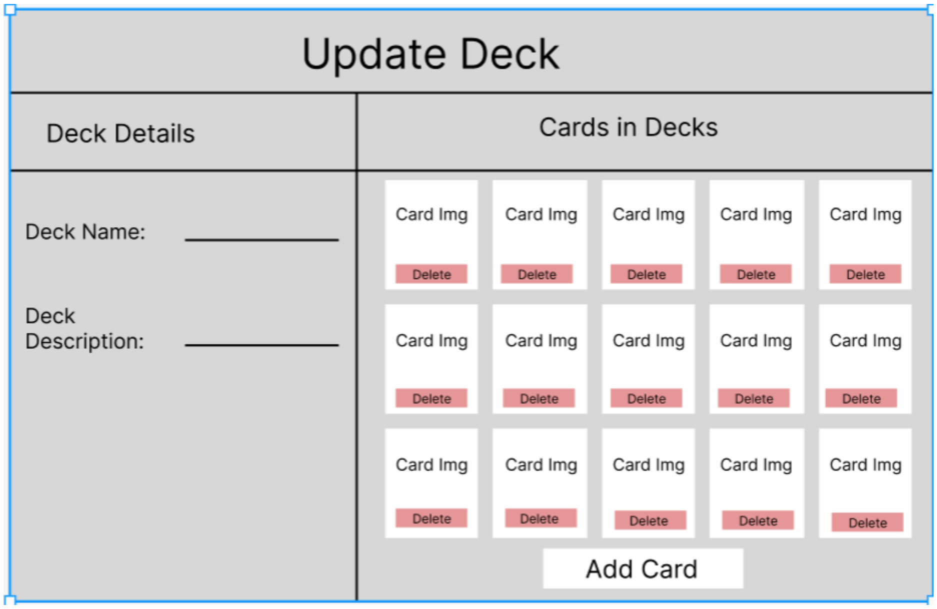
Update Player Wireframe

Players can have multiple decks. In this interface, you can assign new decks to a player or remove existing ones.



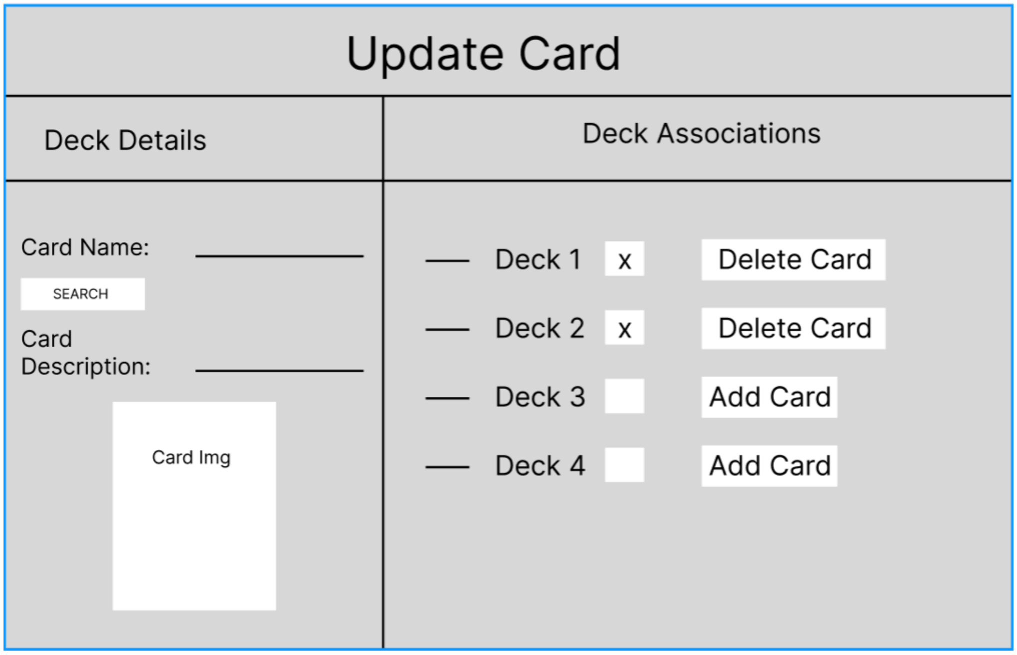
Update Deck Wireframe

The left column displays information about the player's deck, such as the deck name and description, which can be updated. The right column lists relationships (e.g., cards like “Lightning Bolt” and “Llanowar Elves” in the deck). Cards can be deleted from the deck and new cards can be added which will take you to the Update Card window.



Update Card Wireframe

Cards can belong to multiple decks, reflecting a Many-Many relationship. The update view allows users to select or deselect decks for a card, like assigning “Lightning Bolt” to Nahid’s Aggro Deck or Nahid’s Control Deck.



Final Notes on Wireframes

API Methods (MVP)

Core Functionality

- **GET**: `ListPlayers`, `ListDecks`, `ListCards`, `ListCardsForDeck`, `ListDecksForPlayer`
- **POST**: `AddPlayer`, `AddDeck`, `AddCard`, `AddCardToDeck`
- **DELETE**: `DeletePlayer`, `DeleteDeck`, `DeleteCard`, `RemoveCardFromDeck`
- **PUT**: `UpdatePlayer`, `UpdateDeck`, `UpdateCard`

Additional Features

- Deck previews with images of featured cards.
- Card artwork add/remove functionality.
- Card filtering by type, colour, or set.
- A “format” entity (e.g., Standard, Commander) with associated restrictions.
- Player avatars or deck cover images.
- Integration with MTG API for automatic card updates.
- Search functionality for players, decks, or cards.
- Featured decks on the homepage.