Building a Discord Game with Postgres

Agenda

- About us
- Motivation behind the Project
- Tools we used
- How our game works
- What we experienced & learned
- Conclusion
- Demonstration

About us

Caspar Klein

Student at the HU Berlin, majoring in Computer Science (4th Semester)

embedded software engineer (working student)

Xenia Kukushkina

Student at the HTW Berlin majoring in Computer Science in Culture and Health (4th Semester)

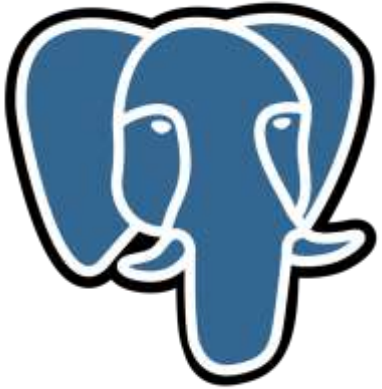
working student in Data Analytics

Motivation behind the project

- originally a Uni-Project for the *Databases Class*
- what was needed was an ERD (Entity-Relationship-Diagram) and to build the database schema and show some sample SQL-queries
- shared interest in gaming
- personal interest in developing games

- so we build a text based RPG (Role Play Game) with basic functionality
  - player combat
  - item gathering

PostgreSQL



- it was given as the database system of choice by the university

I also asked our Professor why they decided to use PostgreSQL

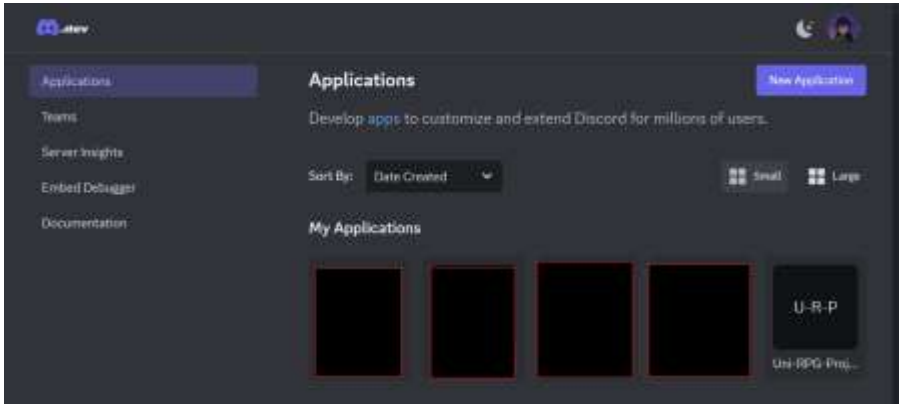
- Broadly used; many use cases thus nice to have for students
- Open Source, no licensing fees
- Can be easily hosted on campus

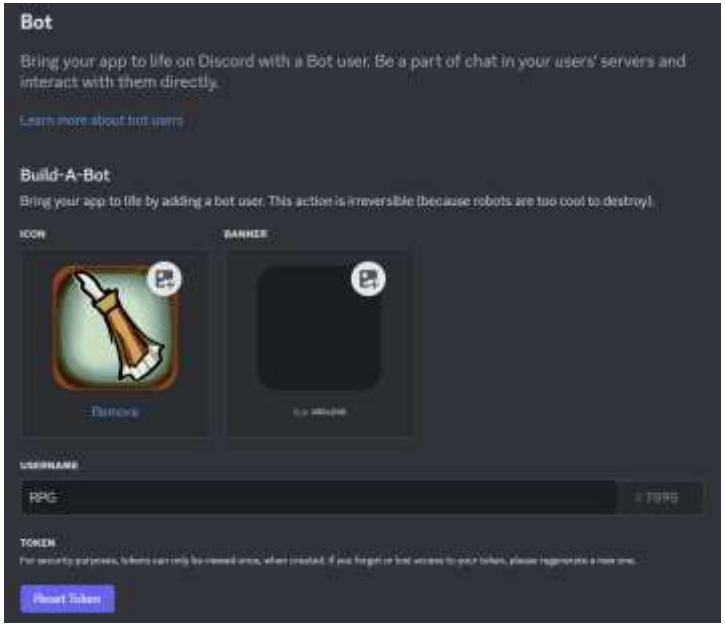
However the University Server has reached EOL

	version text
1	PostgreSQL 12.18 (Debian 12.18-1.pgdg110+2) on x86_64-pc-linux-gnu, compiled by gcc (Debian 10.2.1-6) 10.2.1 20210110, 6...

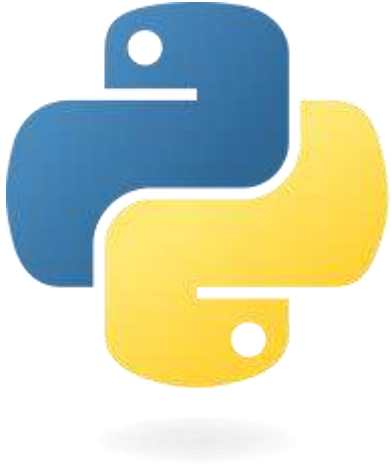
Discord

- easy to use developer portal
- “free” hosting
- “build-in” multiplayer support





Python

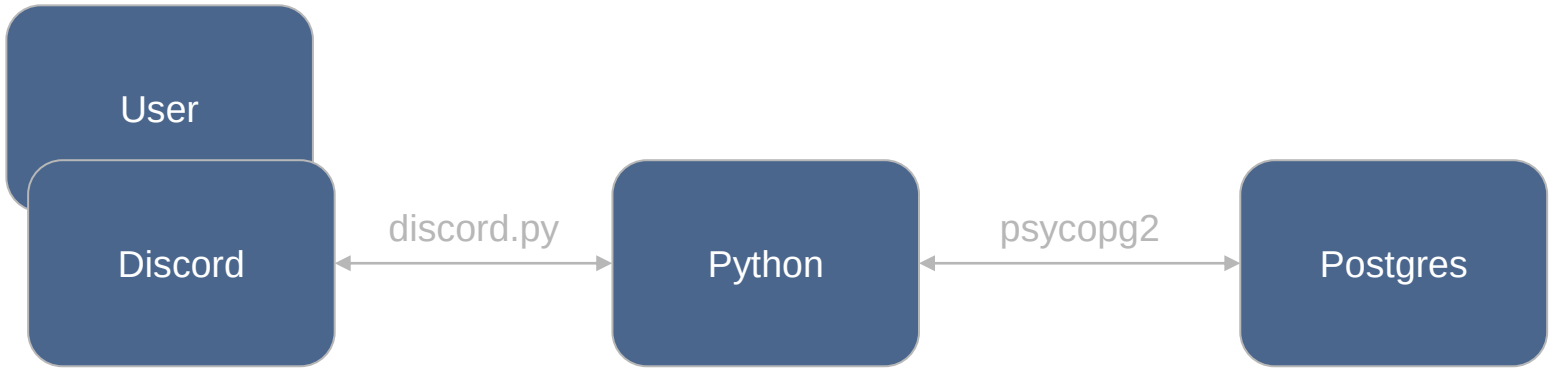


- the programming language most are familiar with
- easy to work with
- nice package library, we used psycopg2 and discord.py

Why psycopg2?

At the start of the project we googled how to connect yourself to the database with python - psycopg2 was recommended the most, so we used it. We know there it the more modern psycopg3 but it was to bothersome to change it :)

API Integrations

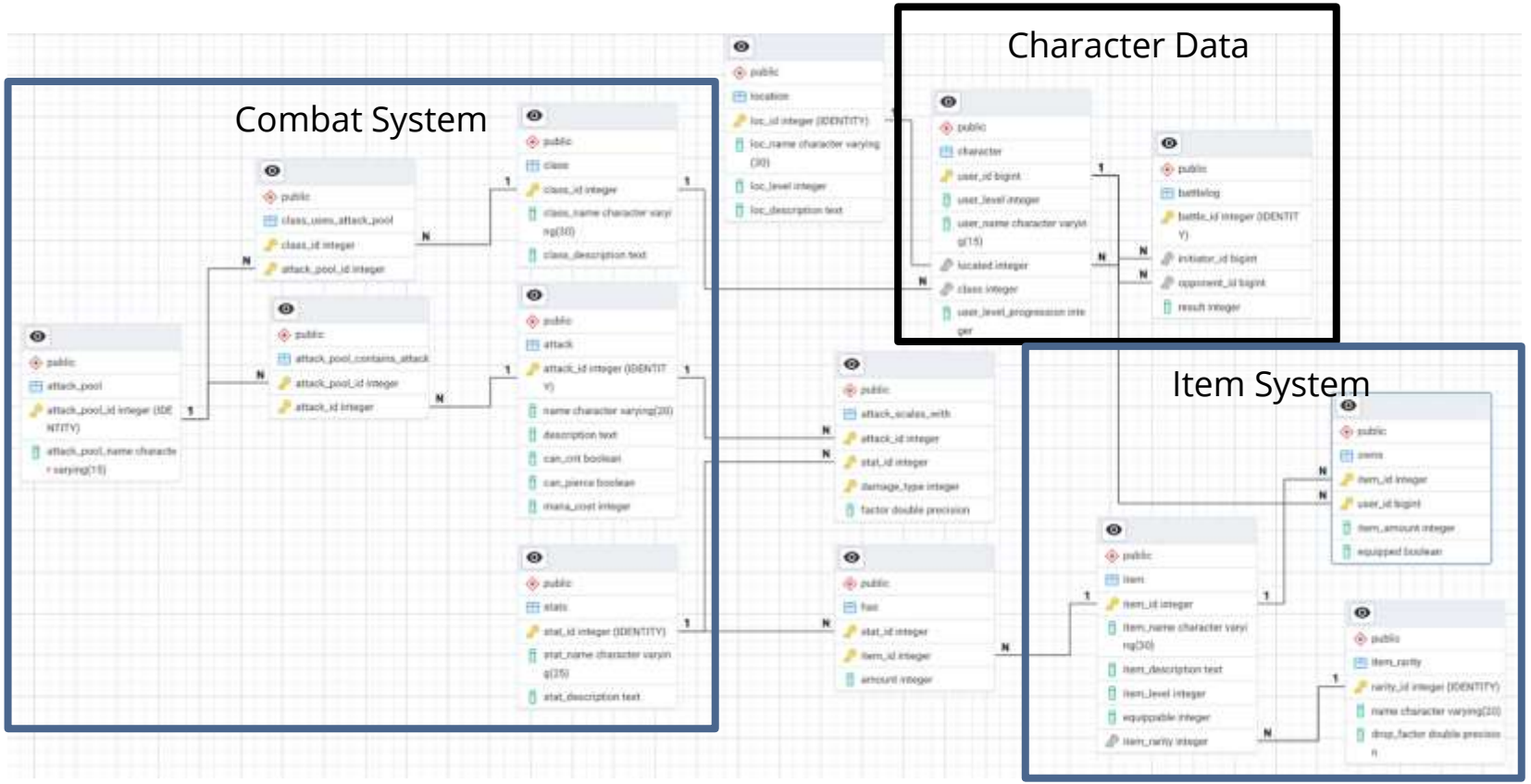


- Handles UI/IO

- Handles Game Logic
- Handles API connections

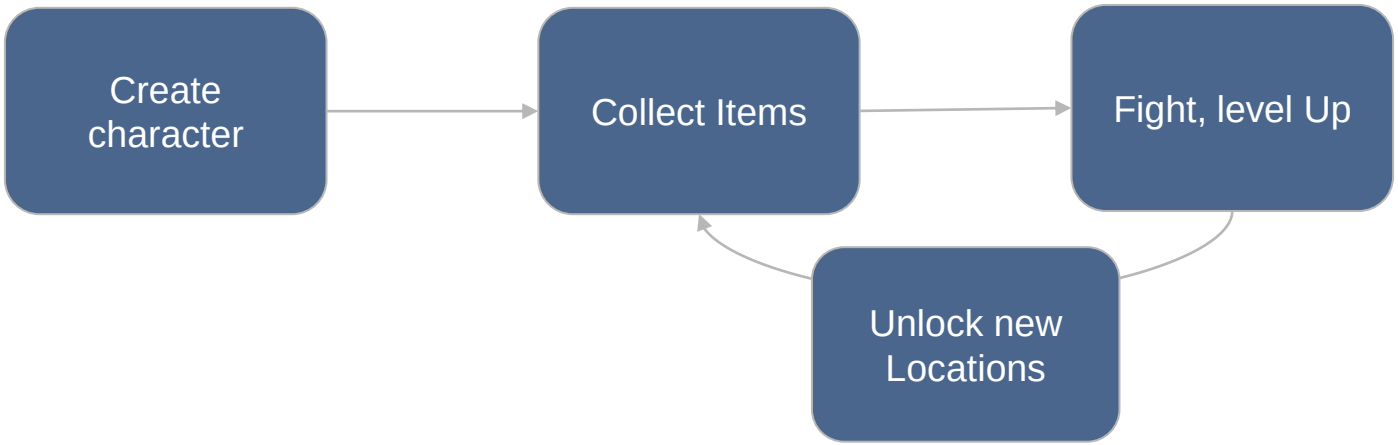
- Manages Gamedata

Database Schema

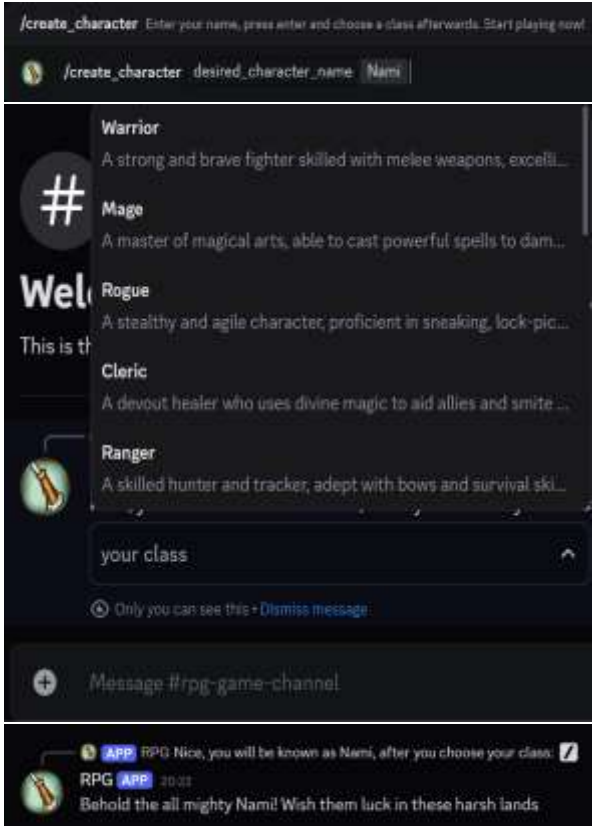


Gameplay

Basic Gameplay loop



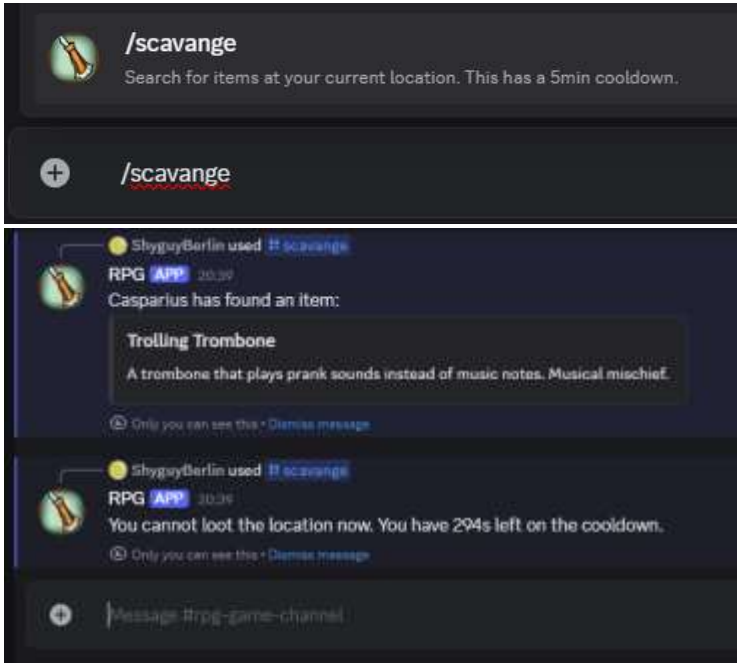
Gameplay - Character Creation




```python
classes=[]
sheet=class_choice_view()
sheet.character_class.plugin=self
sheet.character_class.options=[]
#char_class = ui.Select(placeholder="Your Class",options=[])
#Getting Character Classes

if self.db_cur:
    self.db_cur.execute("SELECT c.class_id,c.class_name,c.class_description FROM class c")
    classes=self.db_cur.fetchall()
for i in classes:
    sheet.character_class.append_option(discord.SelectOption(label=i[1],description=shorten_description(i[2]),value=i[0]))

sheet.add_item(sheet.character_class)

# Save name for later use
data = load_json()
if not "open_creation_prompts" in data:
    data["open_creation_prompts"]={}
data["open_creation_prompts"][str(ctx.user.id)]=desired_character_name
dump_json(data)


await ctx.response.send_message(f"Nice, you will be known as {desired_character_name}, after you choose your class:",view=sheet,ephemeral=True)
```

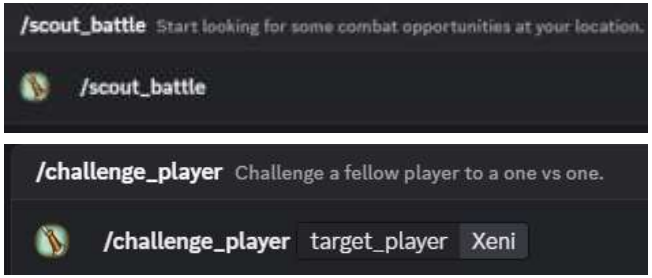
```python
data = load_json()
if "open_creation_prompts" in data:
    if str(interaction.user.id) in data["open_creation_prompts"]:
        name = data["open_creation_prompts"][str(interaction.user.id)]
        del data["open_creation_prompts"][str(interaction.user.id)]
        dump_json(data)
        self.plugin.db_cur.execute(f"INSERT INTO character VALUES (%s,1,%s,0,%s)",(str(interaction.user.id),name,int(interaction.data['values'][0])))
        self.plugin.db.commit()
        weird_announcements=["They are still young, but will conquer the world sooner or later.","Keep an eye on them or they will throw you from your
        await interaction.response.send_message(f"Behold the all mighty {name}! "+weird_announcements[random.randint(0,len(weird_announcements)-1)])
else:
    await interaction.response.send_message("Something went wrong. Maybe refer to an admin if it happens again.")
```

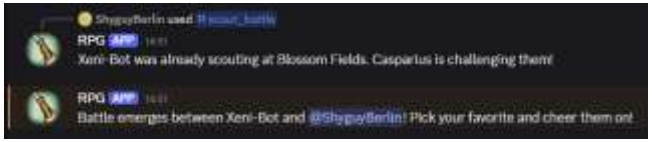
Gameplay - Item Gathering

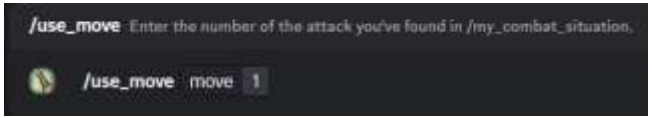


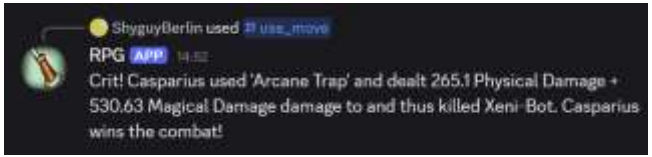

```
if self.db_cur:
    self.db_cur.execute(f"SELECT * FROM valid_item_pools WHERE user_id=%s",(ctx.user.id,))
    db_item_pool=self.db_cur.fetchall()
    for i in range(len(db_item_pool)):
        level_weights=[0.4,0.7,1,0.2]
        print(db_item_pool[i])
        item_pool.append([i,level_weights[min(3,max(0,db_item_pool[i][6]+2))]*db_item_pool[i][5]])
    summed_weight=sum([x[1] for x in item_pool])
    rand_choice=random.random()*summed_weight
    chosen=None
    for i in range(len(item_pool)):
        if rand_choice<item_pool[i][1]:
            chosen=db_item_pool[item_pool[i][0]]
            break
        rand_choice-=item_pool[i][1]
    if chosen:
        print(chosen)
        #The selected item is already present in the players inventory
        if chosen[7]!=None:
            self.db_cur.execute(f"UPDATE owns SET item_amount=%s WHERE item_id=%s and user_id=%s;",(chosen[7]+1,chosen[2],chosen[0]))
            self.db.commit()
        #The selected item is new to this player
        else:
            self.db_cur.execute(f"INSERT INTO owns VALUES (%s,%s,%s,%s);",(chosen[2],chosen[0],1,False))
            self.db.commit()

        item_embed=discord.Embed(title=chosen[3],description=chosen[8])

        await ctx.response.send_message(f"{chosen[1]} has found an item:",embed=item_embed,ephemeral=True)
    else:
        await ctx.response.send_message(f"I, the bot, am not able to randomly select an item, sorry",ephemeral=True)
else:
    await ctx.response.send_message(f"I, the bot, am not able to connect to the database, sorry",ephemeral=True)
```

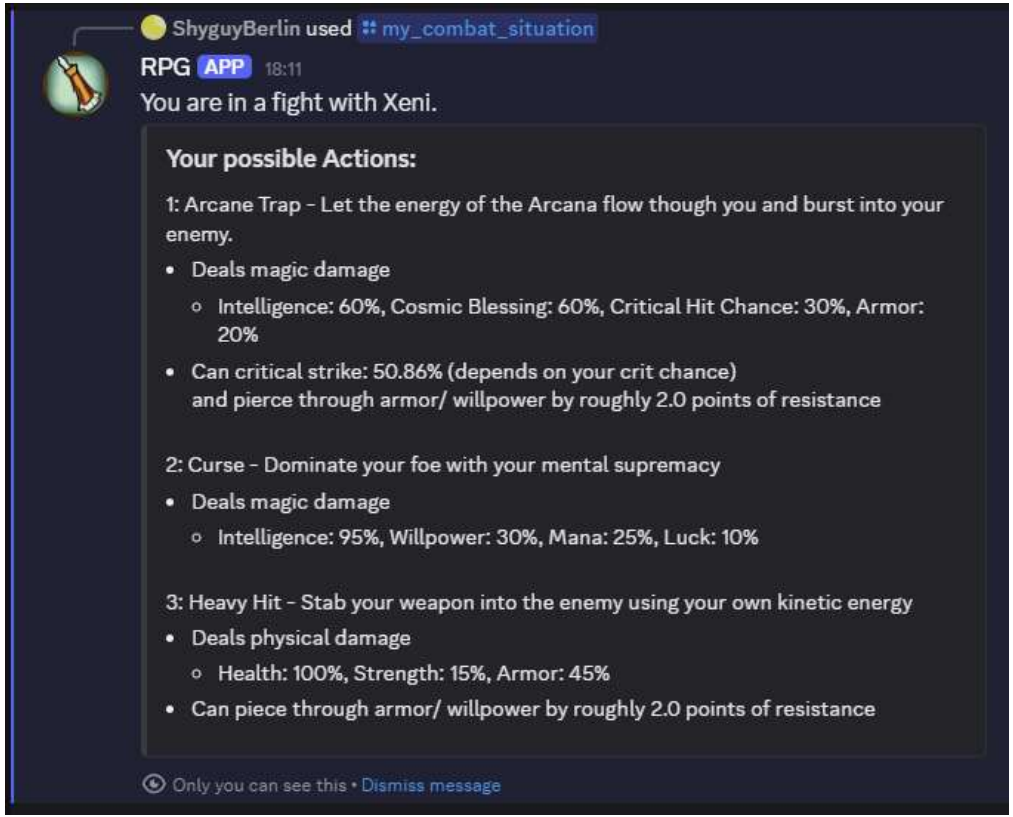
Gameplay - Combat System











Problems and Observations

pgAdmin 4

- beginner friendly
- easy way to create tables
- useful to get an overview of our database schema
- shows all options available to you

Console commands

- more advanced commands (like getting the database dump)

Problems and Observations

- Many available data types

```
                                Table "public.character"
       Column        |         Type          | Collation | Nullable | Default
---------------------+-----------------------+-----------+----------+---------
 user_id             | bigint                |           | not null |
 user_level          | integer               |           | not null | 1
 user_name           | character varying(15) |           | not null |
 located             | integer               |           | not null | 0
 class               | integer               |           | not null | 0
 user_level_progression | integer            |           | not null | 0
Indexes:
    "User_pkey" PRIMARY KEY, btree (user_id)
    "fki_user_loc_id_fkey" btree (located)
Foreign-key constraints:
    "user_class_id_fke" FOREIGN KEY (class) REFERENCES class(class_id)
    "user_loc_id_fkey" FOREIGN KEY (located) REFERENCES location(loc_id)
```

- Nullable fields or default values

```
                                  Table "public.attack"
   Column    |         Type          | Collation | Nullable |              Default
-------------+-----------------------+-----------+----------+-----------------------------------
 attack_id   | integer               |           | not null | generated by default as identity
 name        | character varying(20) |           |          |
 description | text                  |           |          |
 can_crit    | boolean               |           | not null | false
 can_pierce  | boolean               |           | not null | false
 mana_cost   | integer               |           |          |
Indexes:
    "attack_pkey" PRIMARY KEY, btree (attack_id)
```

Problems and Observations

- Read the docs!

- Secure queries

```
self.db_cur.execute(f"SELECT loc_id, loc_name FROM location WHERE loc_level<={player.level} and loc_name='{str(target_location)}'")
if self.db_cur.rowcount==0:
    await no_loc_found()
    return
loc=self.db_cur.fetchone()
```

```
self.db_cur.execute(f"SELECT loc_id, loc_name FROM location WHERE loc_level<=%s and loc_name=%s",(player.level,str(target_location)))
if self.db_cur.rowcount==0:
    await no_loc_found()
    return
loc=self.db_cur.fetchone()
```

Problems and Observations

- Privilege Management

```
                                   List of roles
   Role name    |                         Attributes                         | Member of | Description
----------------+------------------------------------------------------------+-----------+-------------
postgres        | Superuser, Create role, Create DB, Replication, Bypass RLS | {}        |
rpg_bot         |                                                            | {}        |
rpg_bot_testing |                                                            | {}        |
```

```
--
-- Name: TABLE "character"; Type: ACL; Schema: public; Owner: postgres
--

GRANT SELECT,INSERT,DELETE,UPDATE ON TABLE public."character" TO rpg_bot;
```

Conclusion

Overall: fun project to learn how to write a small application with databases

If you want to play around with the bot on discord: [QR/Link]

discord

you can also host it yourself, just follow the readme in the repository

- (https://github.com/xen1i/RPG_Project)

github

Demonstration + Questions