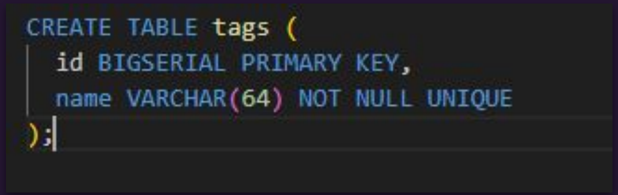
# *Team B2MAGE (MAGE Platform)*

***Team members:*** **Peter B, Vaishnav Manoj, Qasim I, Brian N, MirAhnaf A, Joshua B**

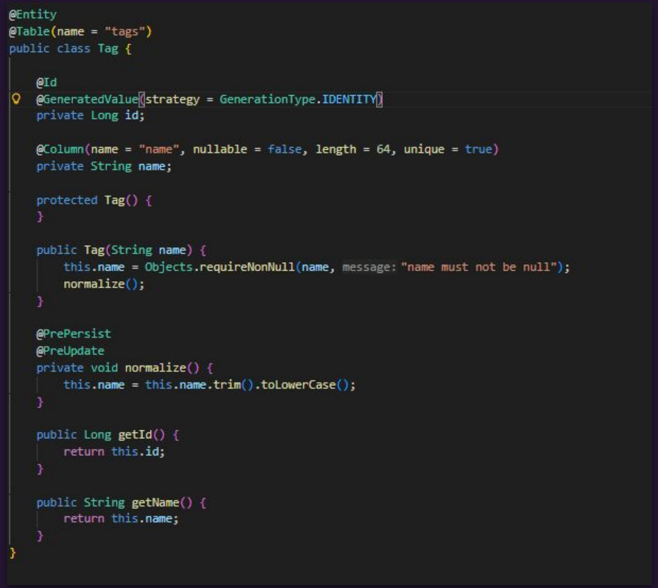
Vaishnav Manoj - 1939

## What I Built



```sql
CREATE TABLE tags (
  id BIGSERIAL PRIMARY KEY,
  name VARCHAR(64) NOT NULL UNIQUE
);
```

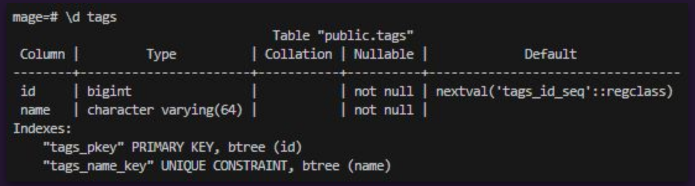
- Created a database table to store music tags (with id and name fields)
- Built a Tag model to represent tags in the application code
- Set up a repository to save and retrieve tags from the database
- Ensured tag names are always formatted consistently (trimmed and lowercase)
- Wrote automated tests to verify the system works correctly (saves tags, prevents duplicates, applies formatting)



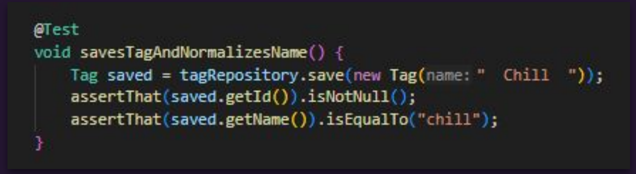
```java
@Entity
@Table(name = "tags")
public class Tag {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(name = "name", nullable = false, length = 64, unique = true)
    private String name;

    protected Tag() {
    }

    public Tag(String name) {
        this.name = Objects.requireNonNull(name, message: "name must not be null");
        normalize();
    }

    @PrePersist
    @PreUpdate
    private void normalize() {
        this.name = this.name.trim().toLowerCase();
    }

    public Long getId() {
        return this.id;
    }

    public String getName() {
        return this.name;
    }
}
```

## How I Tested

- Started postgres + backend with Docker Compose
- Verified the table exists in Postgres (psql: \d tags)
- Ran integration tests and confirmed they pass



```
mage=# \d tags
                              Table "public.tags"
 Column |          Type          | Collation | Nullable |               Default
--------+------------------------+-----------+----------+--------------------------------------
 id     | bigint                 |           | not null | nextval('tags_id_seq'::regclass)
 name   | character varying(64)  |           | not null |
Indexes:
    "tags_pkey" PRIMARY KEY, btree (id)
    "tags_name_key" UNIQUE CONSTRAINT, btree (name)
```



```java
@Test
void savesTagAndNormalizesName() {
    Tag saved = tagRepository.save(new Tag(name: "  Chill  "));
    assertThat(saved.getId()).isNotNull();
    assertThat(saved.getName()).isEqualTo("chill");
}
```

Code: Github  Artifact: link

Website link: link

Research Paper Idea: Real-Time State Distribution for Live Media Platforms - How live media platforms efficiently distribute real-time application state from a host to many viewers using technologies to ensure low-latency synchronization and a smooth shared experience.