TASKMASTER

APLIKACIJA ZA VOĐENJE EVIDENCIJE O DNEVNIM AKTIVNOSTIMA ZAPOSLENIH

TEHNIČKA DOKUMENTACIJA I UPUTSTVO

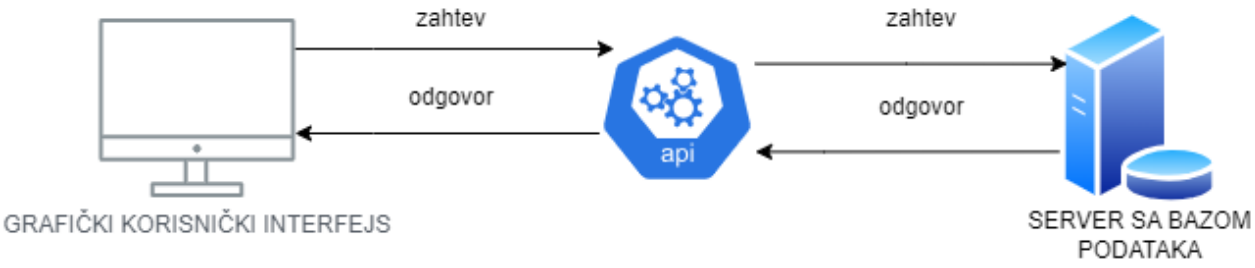
Marko Petrović

Opis aplikacije

Prema zadatom zahtevu o budućoj skalabilnosti aplikacije, za njen razvoj iskorišćen je flutter framework koji je zasnovan na dart programskom jeziku. Aplikacija je zamišljena kao veb aplikacija, i trenutno poseduje sledeće funkcionalnosti:

- Superadmin, admin, pretplatnik i korisnik korisničke uloge sa različitim nivoima pristupa funckijama aplikacije
- Registracija novog pretplatničkog naloga, login i logout.
- Kreiranje novih korisničkih naloga
- Brisanje korisničkih naloga
- Kreiranje novih taskova i dodela korisnika koji su zaduženi za njih, kao i dodela kategorija taskovima
- Brisanje taskova
- Promena trenutne subskripcije u slučaju pretplatnika
- Promena vrednosti subskripcije u slučaju superadmina
- Nephodne validacije pri registraciji, login-u ili kreiranju novih taskova ili korisnika

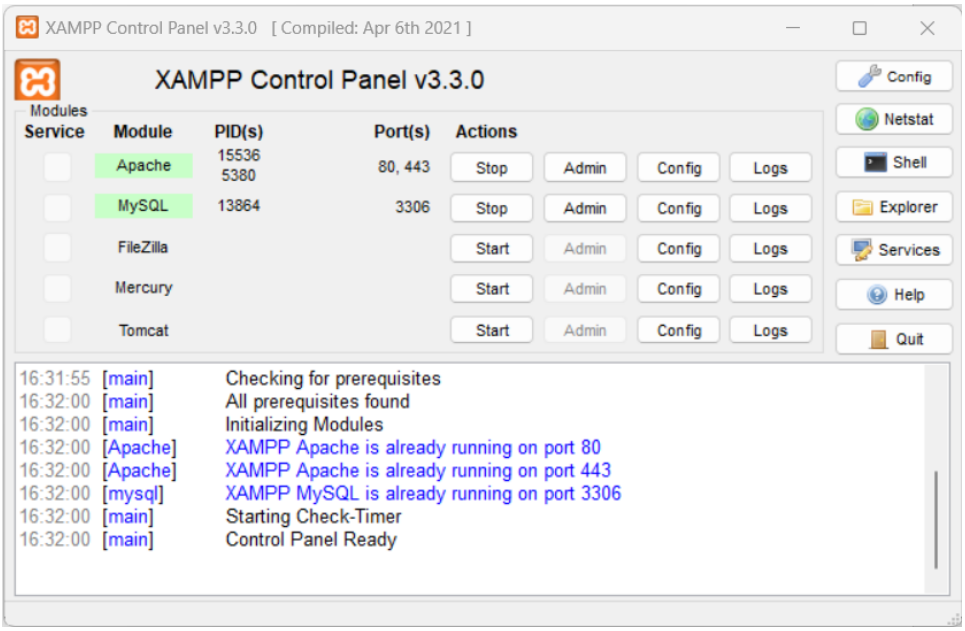
Svi podaci se čuvaju u SQL bazi podataka koja se nalazi na udaljenom* serveru (za potrebe prototipa korišćen je MySql koji je pokrenut lokalno na korisničkoj mašini). Komunikacija između baze podataka i grafičkog korisničkog interfejsa ostvarena je preko različitih API endpoint-a koji su umetnuti na zahtevanim mestima grafičkog korisničkog interfejsa. Različite funkcije koje omogućavaju komunikaciju sa bazom podataka kreirane su pomoću php programskog jezika.



Ilustracija 1- Arhitektura aplikacije

Neophodni koraci pre pokretanja aplikacije

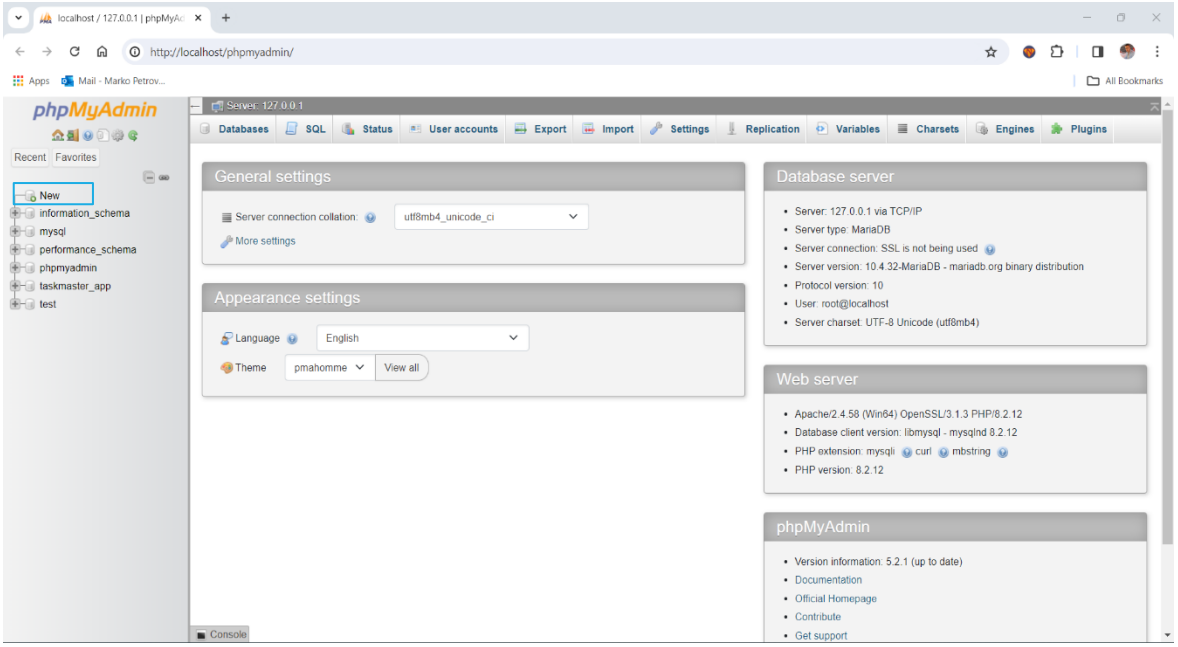
Za testiranje ove aplikacije potrebno je preduzeti nekoliko koraka. Pre svega na našem računaru treba da imamo instalirani XAMPP. Nakon instalacije potrebno je da pokrenemo Apache i MySql:



Ilustracija 2 - XAMPP kontrolna tabla

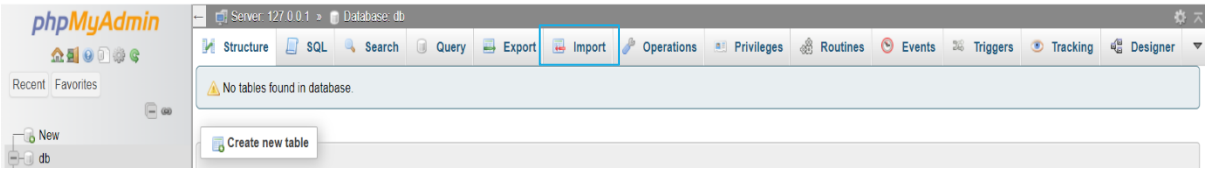
Nakon toga klikom na dugme admin u okviru MySql bićemo preusmereni na stranicu phpMyAdmin gde je sledeći korak import dostavljene baze podataka.

U okviru levog panela biramo stavku New i kreiramo za sada praznu bazu podataka pod proizvoljnim nazivom.



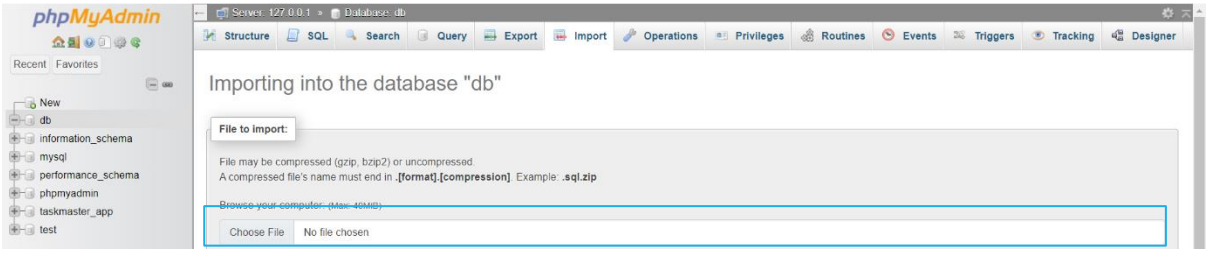
Ilustracija 3 - Kreiranje baze podataka

Sada kada se nalazimo u okviru naše kreirane baze podataka potrebno je da importujemo bazu podataka pomoću .sql datoteke koju smo dobili. Klikom na dugme import u okviru glavnog menija otvara se kartica za importovanje baze.

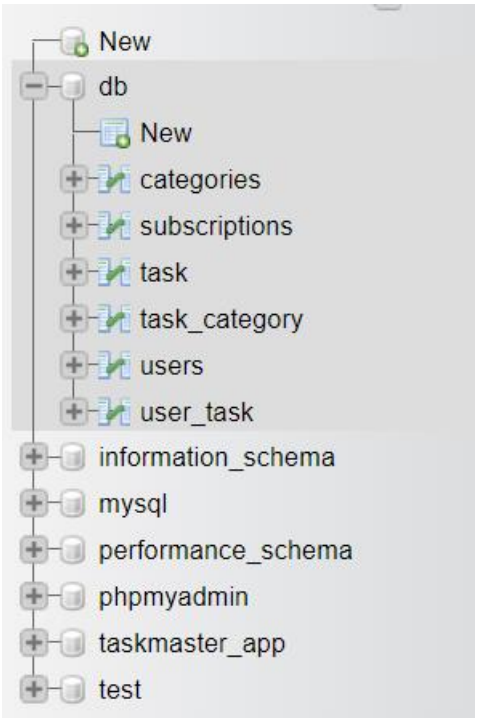


Ilustracija 4 - Importovanje baze

Klikom na dugme import u okviru glavnog menija otvara se kartica za importovanje baze.



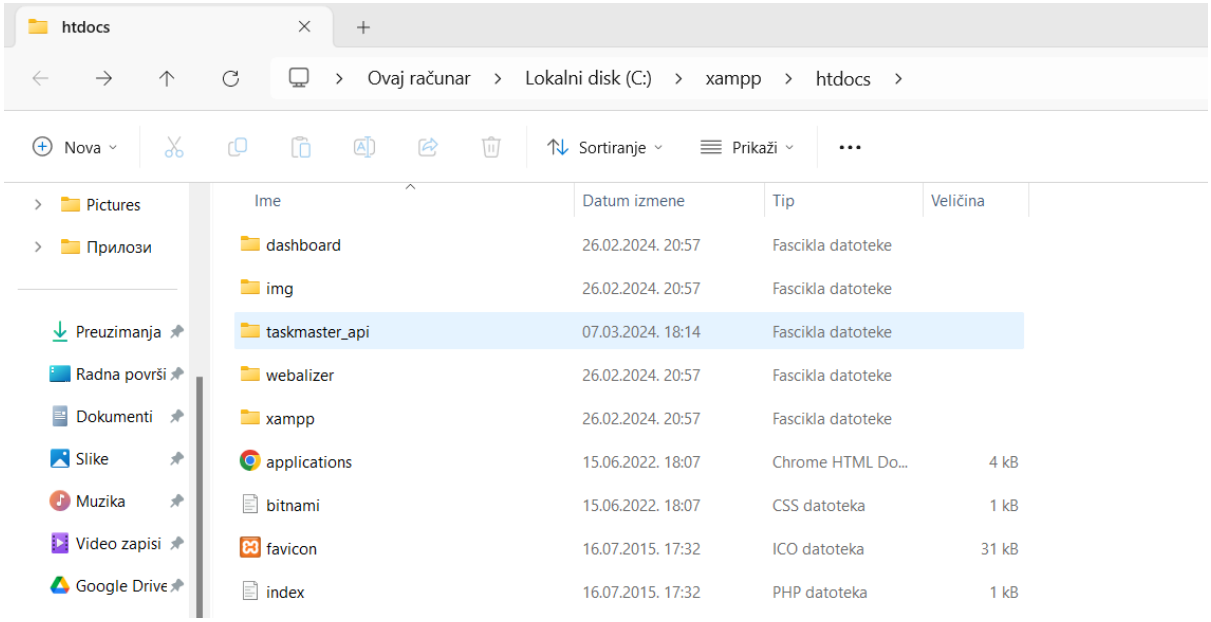
Ilustracija 5 - Importovanje baze



Ilustracija 6 - Kreirana baza podataka

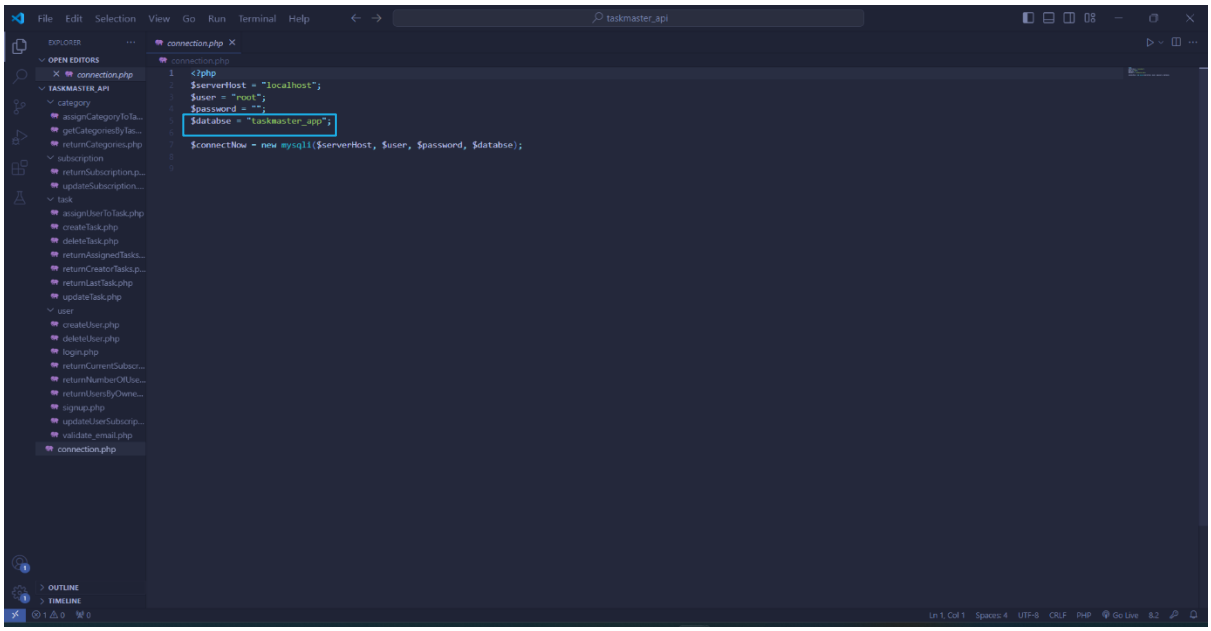
Klikom na Choose File biramo dati .sql fajl i zatim na dnu stranice klikom na dugme Import završavamo proces. Nakon toga naša baza podataka biće popunjena sa odgovarajućim tabelama i nekim predefinisanim podacima u okviru datih tabela.

Sledeći korak jeste da folder taskmaster_api postavimo u okviru XAMPP htdocs foldera. Ovaj folder sadrži sve neophodne php funkcije za komunikaciju sa bazom podataka.



Ilustracija 7 - XAMPP htdocs folder

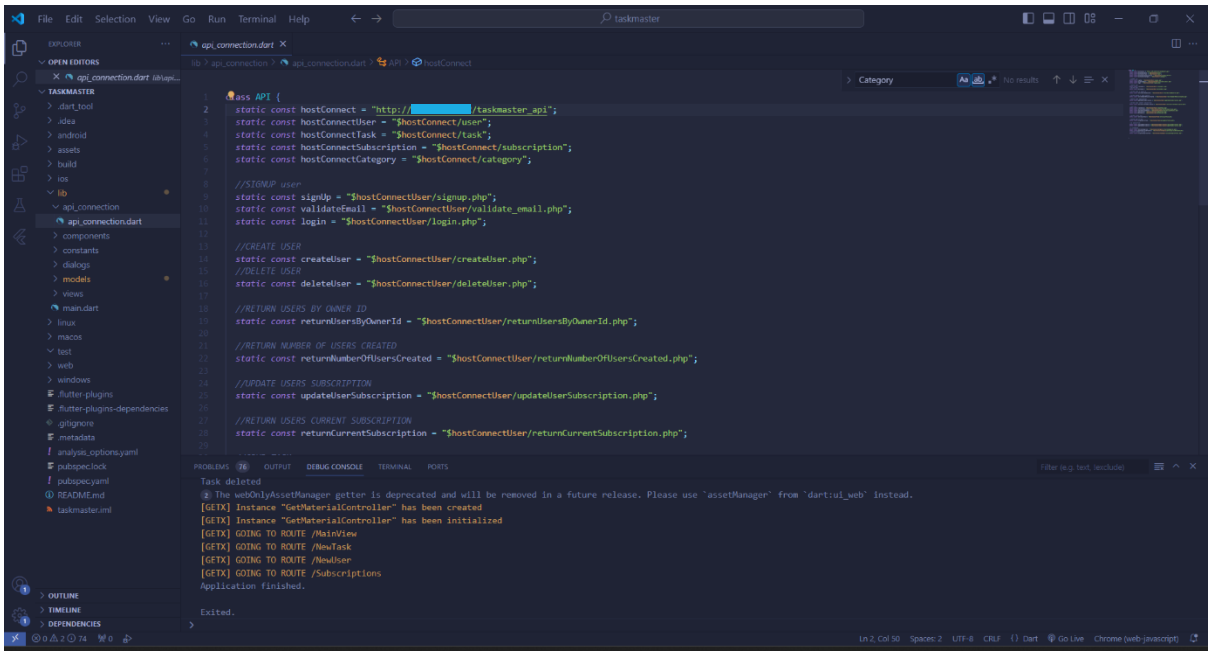
Nakon toga potrebno je da ovaj folder otvorimo u okviru nekog editora (za razvoj ove aplikacije korišćen je Visual Studio Code) i napravimo još jednu izmenu u okviru connection.php fajla.



Ilustracija 8 - Fajlovi datoteke taskmaster_api

U okviru promenljive $database unosimo ime naše kreirane baze podataka i tako ćemo omogućiti komunikaciju aplikacije sa njom.

Nakon ovog koraka potrebno je da u okviru Visual Studio Code-a otvorimo i glavnu datoteku gde se nalazi kod Taskmaster aplikacije. Nakon otvaranja u meniju sa leve strane potrebno je da navigiramo do datoteke api_connection koja sadrži api_connection.dart fajl.

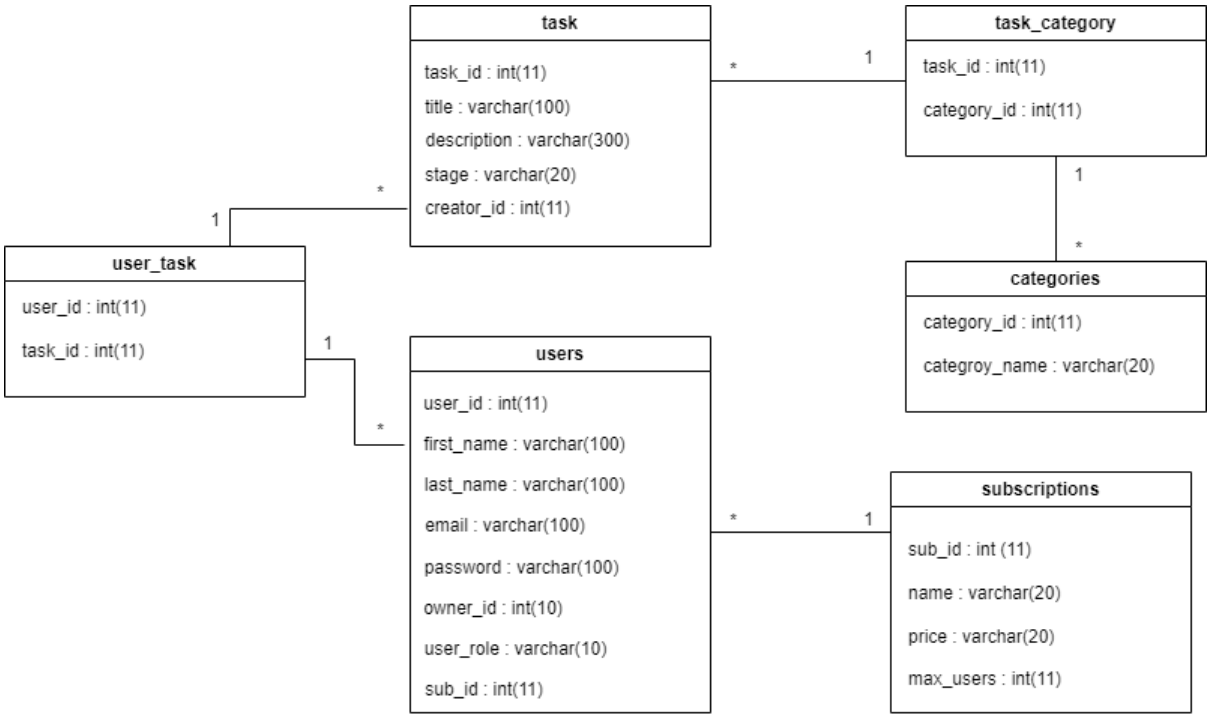


Ilustracija 9 - API fajl

U promenljivoj hostConnect potrebno je da unesemo IP adresu našeg računara. Kako bi smo proverili IP adresu, u komandnom terminalu (ukucati cmd u Windwos pretrazi) mozemo ukucati komandu **ipconfig** i nakon što pritisnemo Enter možemo pročitati i kopirati Ipv4 adresu u polje označeno pravougaonikom. (format : http://xxx.xxx.x.xx/taskmaster_api). Ovime smo obezbedili API endpointe koji će dalje biti pozivani u našem grafičkom korisničkom interfejsu.

Model podataka – Domenski objekti, konceptualni i relacioni model

Pomoću konceptualnog modela opisujemo strukturu sistema. Konceptualni model sadrži konceptualne klase (domenske objekte) i asocijacije između konceptualnih klasa.



Ilustracija 10 - Konceptualni model

Relacioni model

task(task_id, title, description, stage, creator_id)

user_task(user_id, task_id)

task_category(task_id, category_id)

categories(category_id, category_name)

users(user_id, first_name, last_name, email, password, owner_id, user_role, sub_id)

subscriptions(sub_id, name, price, max_users)

Domenski objekti

```dart
class User {
  final int userId;
  final String firstName;
  final String lastName;
  final String email;
  final String password;
  int? ownerId;
  int? subId;
  final String userRole;

  User.ownerUser(this.userId, this.firstName, this.lastName, this.email,
      this.password, this.ownerId, this.userRole, this.subId);

  User(this.userId, this.firstName, this.lastName, this.email, this.password,
      this.userRole, this.subId);

  factory User.fromJson(Map<String, dynamic> json) => User.ownerUser(
        int.parse(json['user_id']),
        json['first_name'],
        json['last_name'],
        json['email'],
        json['password'],
        json['owner_id'] == null ? 0 : int.parse(json['owner_id']),
        json['user_role'],
        json['sub_id'] == null ? 0 : int.parse(json['sub_id']),
      );

  Map<String, dynamic> toJson() => {
        'user_id': userId.toString(),
        'first_name': firstName,
        'last_name': lastName,
        'email': email,
        'password': password,
        'owner_id': ownerId.toString(),
        'user_role': userRole,
        'sub_id': subId.toString()
      };

  Map<String, dynamic> toJsonRegister() => {
        'user_id': userId.toString(),
        'first_name': firstName,
        'last_name': lastName,
        'email': email,
        'password': password,
        'user_role': userRole,
        'sub_id': subId.toString(),
      };

  @override
  String toString() {
    return 'Firstame : $firstName \nLastname : $lastName \nEmail : $email';
  }
}
```

Ilustracija 11 - Domenski objekat User

```dart
class Task extends Equatable {    This class (or a class that this class inherits from) is marked as '@immutable', but one or more of it
  final int id;
  final String title;
  final String description;
  final String? stage;
  final int creatorId;
  List<User>? assignedUsers;
  List<CategoryModel>? assignedCategories;
  // List<CategoryModel> selectedCategories = List.empty(growable: true);

  Task(this.id, this.title, this.description, this.stage, this.creatorId,);
  Task.createTask(this.id, this.title, this.description, this.stage, this.creatorId, this.assignedUsers, this.assignedCategories);

  Task copyWith({
    final int? id,
    final String? title,
    final String? description,
    final String? stage,
    final int? creatorId,
    final List<User>? assignedUsers,
    final List<CategoryModel>? assignedCategories
  }
  ) {
    return Task.createTask(
        id ?? this.id,
        title ?? this.title,
        description ?? this.description,
        stage ?? this.stage,
        creatorId ?? this.creatorId,
        assignedUsers ?? this.assignedUsers,
        assignedCategories ?? this.assignedCategories
        );
  }

  factory Task.fromJson(Map<String, dynamic> json) => Task(
       int.parse(json['task_id']),
       json['title'],
       json['description'],
       json['stage'],
       int.parse(json['creator_id']),
  );

  Map<String, dynamic> toJson() => {
        'task_id': id.toString(),
        'title': title,
        'description': description,
        'stage': stage,
        'creator_id': creatorId.toString()
      };

  @override
  List<Object?> get props => [id, title, description, stage, creatorId];
```

Ilustracija 12 - Domenski objekat Task

```dart
class Stage extends Equatable {
  final int id;
  final String name;
  final Color color;

  const Stage(this.id, this.name, this.color);

  static List<Stage> stages = [
    const Stage(1, 'To Do', Colors.red),
    const Stage(2, 'In Progress', Colors.orange),
    const Stage(3, 'Under review', Colors.yellow),
    const Stage(4, 'Done', Colors.green),
  ];

  @override
  List<Object?> get props => [
    id,
    name,
    color
  ];
}
```

Ilustracija 13 - Domenski objekat Stage

```dart
class Subscription {
  final int subId;
  final String name;
  final String price;
  final int maxUsers;

  Subscription(
      {required this.subId,
      required this.name,
      required this.price,
      required this.maxUsers});

  factory Subscription.fromJson(Map<String, dynamic> json) => Subscription(
      subId: int.parse(json['sub_id']),
      name: json['name'],
      price: json['price'],
      maxUsers: int.parse(json['max_users']));

  Map<String, dynamic> toJson() => {
        'sub_id': subId.toString(),
        'name': name,
        'price': price,
        'max_users': maxUsers.toString()
      };

  @override
  String toString() {
    return "$name with $maxUsers max users to be created, only $price monthly";
  }
}
```

Ilustracija 14 - Domenski objekat Subscription

Ilustracija 15 - Domenski objekat Category

Predefinisani podaci u bazi podataka

U bazi podataka postoje predefinisani korisnički nalozi koji se mogu koristiti za testiranje:

Superadmin nalog:

- Email : mare@gmail.com
- Password : Parcepapira

Admin nalog:

- Email: mika@gmail.com
- Password : Parcepapira

Subscriber nalozi:

- Email: pera@gmail.com
- Password : Parcepapira
- Email: dare@gmail.com
- Password : Parcepapira