

# Task Management System for Teamwork V1.2

Cohort49-2

Vladimir Romaikin

---

Create an application for managing projects and tasks within a team. The application should support user registration and authentication, creating projects, adding participants to projects, assigning tasks with deadlines and priorities, as well as commenting on tasks and exchanging messages between participants. An important part of the project is tracking the progress of tasks and projects.

# Requirements

to the implementation of the application

- Ensuring validation of input data to prevent errors.
- Development should follow the principles of a three-layer architecture, dividing the application into a data access layer, business logic, and user interface.
- It is important to ensure testing of the developed functionality to confirm its correct operation.



# Functional

## applications

- User registration and authentication.
- Project and task management, including creation, assignment and tracking.
- Ability to comment on tasks (additional task: exchanging messages between users).
- Tracking progress on tasks and projects.

# Work items in the application

## Basic elements

### User

#### FIELDS TO FILL IN

1. Id. (Id)
2. Name. (Name)
3. Surname (Surname)
4. Login (Login)
5. Password Hash (passwordHash)

### Project

#### FIELDS TO FILL IN

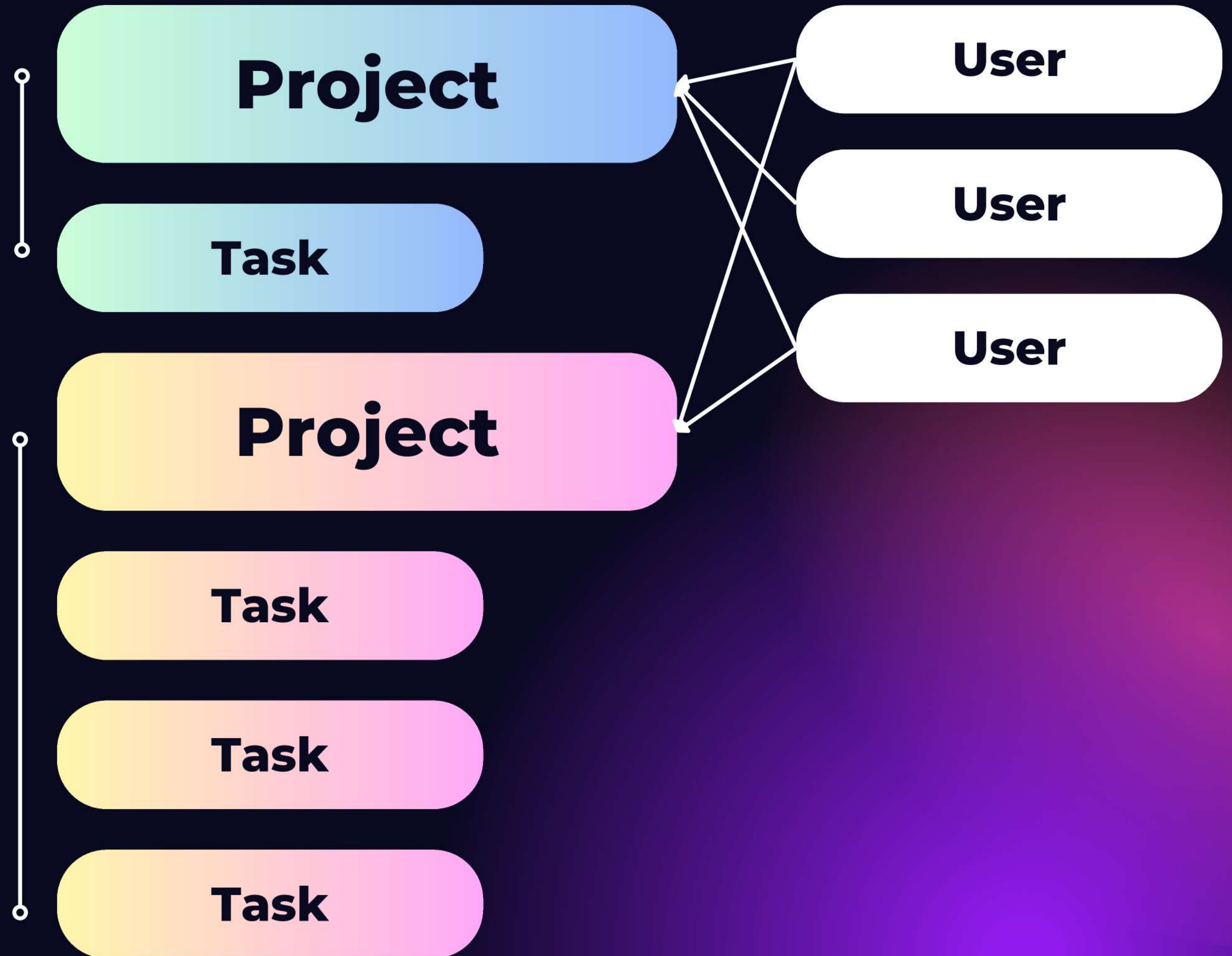
1. Id (IdProject)
2. Title (Title)
3. Description (Description)
4. Date of creation (Created)
5. Date of completion (Deadline)
6. Priority (Priority)
7. Status (Status)
8. Responsible (Executor)
9. Related tasks (Tasktask)

### Task

#### FIELDS TO FILL IN

1. Id (IdTask)
2. Title
3. Description
4. Created
5. Deadline
6. Priority
7. Status
8. Executor

# Structure of the hierarchy of interaction of application entities





# Methods and classes

## Classes

-----Main classes-----

User class  
Project class  
Task class

-----Repositories-----

UserRepository class  
ProjectRepository class  
TaskRepository class

-----Service classes-----

ProjectService class  
TaskService class  
UserService class

-----Additional classes-----

Comment class  
Message class  
Validation class  
ProgressTracking class  
TestJava class



## Methods

Class User

**Objective:** Create a class to represent a user in the system.

### Methods:

- Constructor for all fields
- Getters and setters for all fields (Except password variable)
- toString() for all fields

# Methods and classes

## Methods Class Project

- **Functions:** There must be access to manage this data - adding, updating and receiving information.
- **Relationships:** A project can include many tasks, and each task is linked to a project. The project also has a relationship with performers.

### Methods:

- Constructor for all fields
- Getters and setters for all fields
- Method addTask(taskTask)
- Method removeTask
- Method editTask
- Method toString
- Methods equals and hashCode



## Methods Class Task

- **Functions:** The Task class provides basic capabilities for working with tasks: creating, editing, and displaying information.

### Methods:

- Constructor for all fields
- Getters and setters for all fields
- editTasktask method
- toString method
- equals and hashCode methods



# Methods and classes

## Methods

### Class UserRepository

- **Functions:** responsible for storing and managing users in the system. This storage is used to create, search, update and delete users. We use the Map collection with a key in the form of a user identifier (id) and a value in the form of a User object.
- **Methods:**
  - addUser(User user) method
  - getUserById(String id) method
  - method updateUser(User updatedUser)
  - deleteUser(String id) method
  - getAllUsers() method



## Methods

### Class ProjectRepository

- **Functions:** Project storage. The key is a unique project identifier (id), and the value is a Project object. The Map collection allows you to quickly find projects by their unique identifiers, as well as add and delete projects.
- **Methods:**
  - addProject(Project project) method
  - getProjectById(String id) method
  - method updateProject(Project updatedProject)
  - deleteProject(String id) method
  - getAllProjects() method



# Methods and classes

## Methods

### Class TaskRepository

- **Functions:** Task storage. The key is a unique task identifier (id), and the value is a Task object. This allows you to quickly find tasks by their ID, as well as add, update and delete tasks.
- **Methods:**
  - addTask(Task task) method
  - getTaskById(String id) method
  - method updateTask(Task updatedTask)
  - deleteTask(String id) method
  - getAllTasks() method
  - getTasksByStatus(String status) method



## Methods

### Class ProjectService

- **Functions:** This class is responsible for the business logic related to projects. It interacts with the ProjectRepository to perform operations such as adding, updating, deleting and searching for projects. ProjectService provides the business logic layer.
- **Methods:**
  - Constructor
  - createProject(Project project) method
  - getProjectById(String projectId) method
  - getAllProjects Method
  - method updateProject(Project updatedProject)
  - Method deleteProject(String projectId)
  - Method addTaskToProject(String projectId, Task task)
  - getProjectTasks(String projectId) method
  - Method addParticipant(String projectId, User user)
  - getParticipants(String projectId) method
  - Method removeParticipant(String projectId, User user)



# Methods and classes

## Methods

### Class TaskService

- **Function:** Provides storage and management of devices in the system. This storage is important for future, location, discovery and deletion. users. The collection map has a key between the identifier and password (id) and a value in the form of a User object.
- **Methods:**
  - Constructor
  - Method createTask(Task task)
  - Method getAllTasks()
  - Method getTaskById(String taskId)
  - Method deleteTask(String taskId)
  - Method updateTask(String taskId, Task updatedTask)
  - Method assignTaskToUser(String taskId, User user)
  - Method getTasksByUser(String userId)
  - Method getTasksByProject(String projectId)
  - Method updateTaskStatus
  - Method addCommentToTask
  - Method getCommentsForTask(String taskId)
  - Method searchTasks(String keyword)
  - Method deleteCommentFromTask
  - Method updateCommentForTask



## Methods

### Class UserService

- **Functions:** The UserService class contains methods for managing users, such as adding, deleting, updating, searching, and authenticating. It interacts with the UserRepository repository, which manages the storage of user data.
- **Methods:**
  - Constructor
  - Method createUser
  - getUserById
  - getUserByLogin
  - deleteUser
  - updateUser
  - authenticateUser
  - getAllUsers



# Methods and classes

## Methods Class Comment

- **Functions:** Create and add comments. Display comments.
- **Variables:**
  - id
  - text (comment text)
  - author (refers to the User object)
  - timestamp (records the time the comment was created)
- **Methods:**
  - Constructor
  - List<Comment> comments
  - addComment(Comment comment) method
  - removeComment(String commentId) method
  - method findCommentById(String commentId)
  - toString method



## Methods Class Message

- **Functions:** The Message class provides a framework for exchanging messages between users.
- **Variables:**
  - String id
  - User sender
  - User recipient
  - String content
  - LocalDateTime sentTime
- **Methods:**
  - Constructor
  - Getters and setters for all fields
  - toString method



# Methods and classes

## Methods

### Class ProgressTracking

- **Functions:** Track task progress. Track project progress. Methods for getting and setting progress.

#### **Variables:**

- taskProgress
- projectProgress

#### **Methods:**

- setTaskProgress
- getTaskProgress
- setProjectProgress
- getProjectProgress
- calculateProjectProgress



## Methods

### Class Validation

- **Functions:** Various methods for data validation, such as checking for correct format, string length, and other restrictions.

#### **Methods:**

- Checking that a string is not empty and does not consist only of spaces.
- Checking that a string has a length within certain limits
- Checking for login uniqueness
- Checking that the project execution date is not earlier than the creation date
- Checking for unwanted characters
- Checking for the presence of a user, project or task
- Checking for a match between password and password confirmation
- Checking for a minimum password length



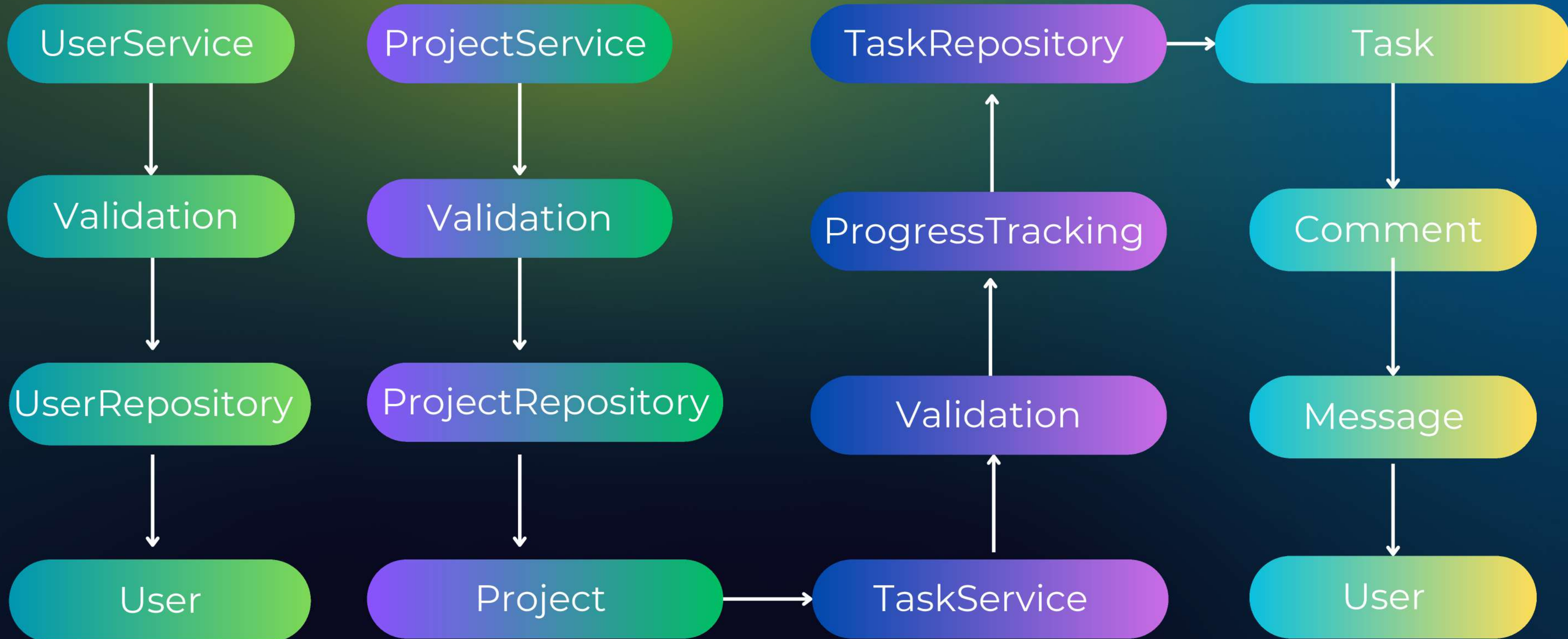
# Methods and classes

## Methods

### Class ProgressTracking

- **Features:** Tests for other methods: You can add tests to test all the methods of the classes, such as editing and deleting objects.

# Class interaction diagram





# UserService

- **UserRepository:**
- createUser(User user) → Call method addUser(User user)
- getUser(String login) → Call method findUserByLogin(String login)
- **Validation:**
- validateUser(User user) → Method for checking the correctness of user data

# ProjectService

- **ProjectRepository:**
- createProject(Project project) → Call method addProject(Project project)
- getProjectById(String id) → Call method findProjectById(String id)
- deleteProject(String id) → Call method removeProject(String id)
- **TaskService:**
- assignTaskToProject(Task task, Project project) → Method for assigning a task to a project
- **Validation:**
- validateProject(Project project) → Method for checking the correctness of project data

# Class interaction diagram



# TaskService

- **TaskRepository:**
- createTask(Task task) → Call method addTask(Task task)
- getTaskById(String id) → Call method findTaskById(String id)
- deleteTask(String id) → Call method removeTask(String id)
- updateTask(Task task) → Call method updateTask(Task task)
- **ProgressTracking**
- updateTaskProgress(String taskId, int progress) → Method to update task progress
- **Validation:**
- validateTask(Task task) → Method for checking the correctness of task data

# Comment

- Related to Task:
- Methods for adding and removing comments to tasks

# Message

- Related to User:
- Methods for sending and receiving messages between users

# Class interaction diagram

# Progress Tracking

- Related to Task and Project:
- Methods for tracking the progress of tasks and projects



# Three-layer architecture

## Project implementation scheme

**01**

---

User interface

**02**

---

Business logic

UserService  
ProjectService  
TaskService

**03**

---

Data layer

UserRepository  
ProjectRepository  
TaskRepository

### Stages of implementation

1. Initial development: creating entities, repositories, services.
2. Integration: combining all layers of the application.
3. Testing: checking the operation of each layer and the entire system.
4. Security and validation: adding data validation.
5. Finishing touches.

**Validation and tracking**

# Project stages and their distribution











## Project participants:

- 1. Parkhomenko Denys
- 2. Romaikin Volodymyr
- 3. Ganina Kira
- 4. Katasonov Oleksantr
- 5. Ivanova Kseniia

Project stage	Responsible	Duration	Start	Completion
Defining requirements	Bce	3 days	01.10.2024	03.10.2024
Architecture design	Parkhomenko Denys	5 days	04.10.2024	08.10.2024
Entity Modeling	Ganina Kira	4 days	09.10.2024	12.10.2024
Data Layer Development	Romaikin Volodymyr	5 days	13.10.2024	17.10.2024
Service Layer Development	Katasonov Oleksantr	6 days	18.10.2024	23.10.2024
UI Layer Development	Ivanova Kseniia	6 days	24.10.2024	30.10.2024
Integration of all layers	Parkhomenko Denys	3 days	31.10.2024	02.11.2024
Security implementation	Romaikin Volodymyr	4 days	03.11.2024	06.11.2024
Development of validation	Katasonov Oleksantr	4 days	07.11.2024	10.11.2024
Logging and progress	Ganina Kira	4 days	11.11.2024	14.11.2024



# Project stages and their distribution

Project stage	Responsible	Schedule of execution
Defining requirements	Bce	
Architecture design	Parkhomenko Denys	
Entity Modeling	Ganina Kira	
Data Layer Development	Romaikin Volodymyr	
Service Layer Development	Katasonov Oleksantr	
UI Layer Development	Ivanova Kseniia	
Integration of all layers	Parkhomenko Denys	
Security implementation	Romaikin Volodymyr	
Development of validation	Katasonov Oleksantr	
Logging and progress	Ganina Kira	

# **Thank you for your attention**

**You can also check out this project on my GitHub.**