Tasky

System Design Document

26.04.2021

Oğuz Kaan Yazan
Berke Biten

IŞIK UNIVERSITY
COMPUTER SCIENCE AND ENGINEERING

SYSTEM DESIGN DOCUMENT

1. Introduction

1.1 Purpose of the System

Most of the people work on multiple tasks every day in their job, school or even in home. Working on these tasks is getting harder if you try to handle it with a pen and paper. Also, for the people who are responsible for managing a team, it is really hard to monitor the project, tasks and the team. The purpose of the task management system Tasky is making easier and more efficient working and managing the projects and tasks. Users will easily create, assign and prioritize tasks, set deadlines, track how much time spent on tasks and visualize the projects and tasks with the Tasky.

1.2 Design Goals

a) Performance:

Response time is an important issue for us. The response time of the system should be minimum in order to ensure smooth usage for our web and mobile application.

b) Dependability:

The system should be robust against the invalid user inputs. In order to do that we validate all the user inputs in the front-end and back-end of the system. We put validations for the data that is sent by the user in the web and mobile application. After the data proceed this validation, we again validate the data in the server.

The system should be reliable. We control the data sent by users and roles of the user before updating the any data in the system in order to prevent undesirable behaviors of the system.

The system should be available 24/7. Users can access and make operations whenever they want.

c) End-User:

Utility and usability of the system are main goals of the system. Any of the user's projects should be manageable with the Tasky system. We ensure very useful interfaces for project and task management processes to the users who has different kinds of roles. Also, with the simplicity of the web and mobile applications, users can do the operations very quickly in the Tasky system.

1.3 Definitions, Acronyms, and Abbreviations

JWT Token: A token that is created when the user logged into the system. It provides the authorization of the user for web api requests.

Authentication: Register and login functions.

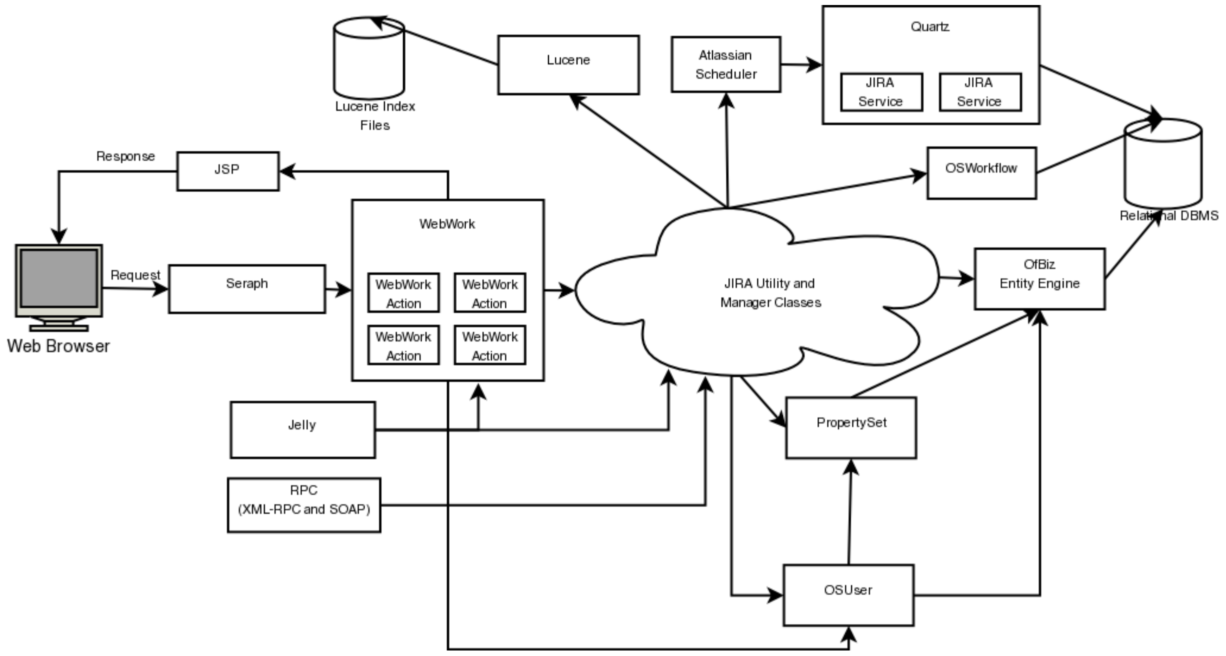
Client side: Web and mobile application of the Tasky System.

Front-end: Any screen/page that is interact with the user.

1.4 References

https://developer.atlassian.com/server/jira/platform/architecture-overview/

Current Software Architecture



As a current software we examined the Jira web application. Jira web application is written in Java. It runs on web browsers. Jira uses WebWork 1 for web requests. It has a MVC architecture. HTML that is created after web requests are created by JSP. WebWork actions created by the user is forwarded to Jira Utility and Manager classes. Then with the help of intermediate layers such as OSWorkflow, PropertySet etc. relational database of the Jira system is updated.

2. Proposed Software Architecture

2.1 Overview

This section describes how system decomposed into layers and subsystems. Functions of the subsystems and relationship between subsystems are also described in this section.

2.2 System Decomposition

System is divided into three layers. First layer is interface. In this layer we have two subsystems.

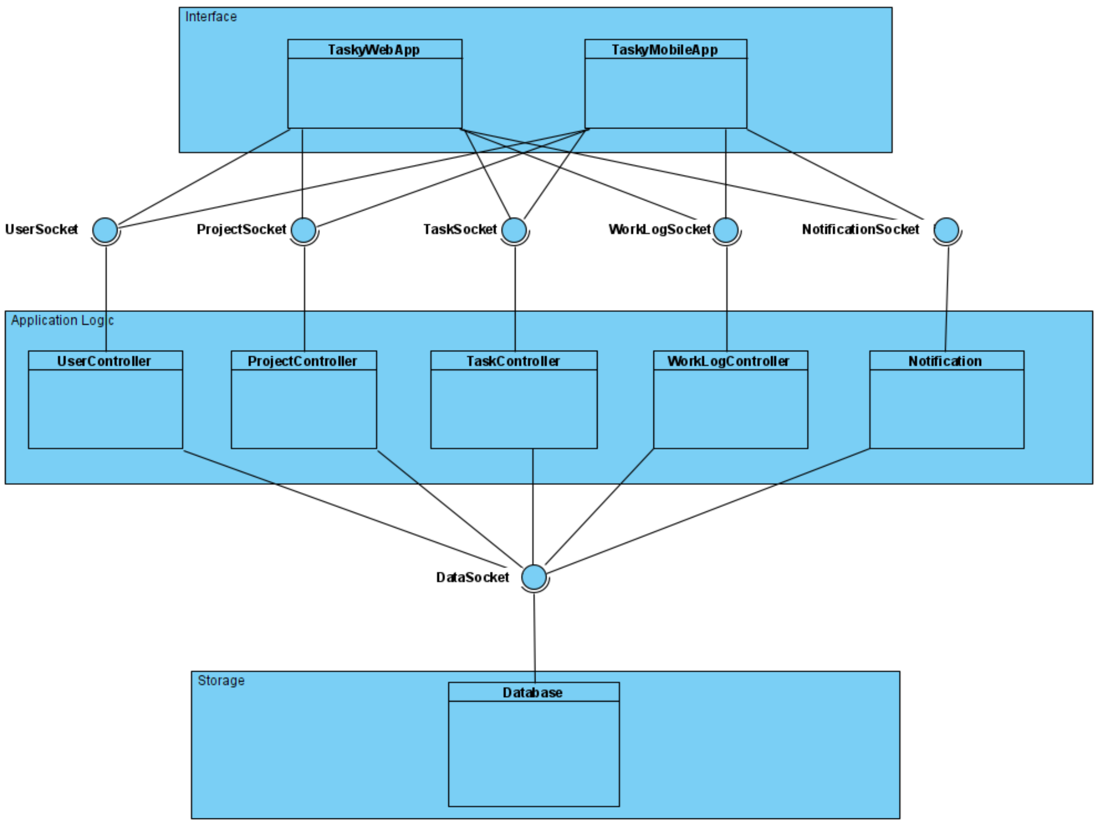
1) **Tasky Web App:** Web application of the Tasky system which runs on internet browsers. All of the user interfaces and interactions in the Tasky web application belongs to this subsystem.
2) **Tasky Mobile App:** Mobile application of the Tasky system which runs on smart phones. All of the user interfaces and interactions in the Tasky mobile application belongs to this subsystem.

Second layer of the Tasky system is application logic layer. This layer contains the server side controllers. Functions of this layer’s subsystems control and validate the data that is sent by user. There are 5 subsystems in this layer.

1) **User Controller:** This subsystem is responsible of the user data. All operations related to user data such as profile and user preferences run on this subsystem. This subsystem also handles the login and register functions of the system. It creates a JWT token for the user and check its authorization in each API call.
2) **Project Controller:** This subsystem handles the operations that is related to project.
3) **Task Controller:** This subsystem handles the operations that is related to task.
4) **Work Log Controller:** This subsystem handles the operations that is related to work log.
5) **Notification Controller:** This subsystem handles the sending notification to the users. Sending e-mail and mobile notification operations run on this subsystem.

Third and final layer of the system is storage layer. All the data of the system is stored and handled in this layer. This layer has a single subsystem.

1) **Database:** This subsystem stores all the data. It also makes insert, update and delete operations.



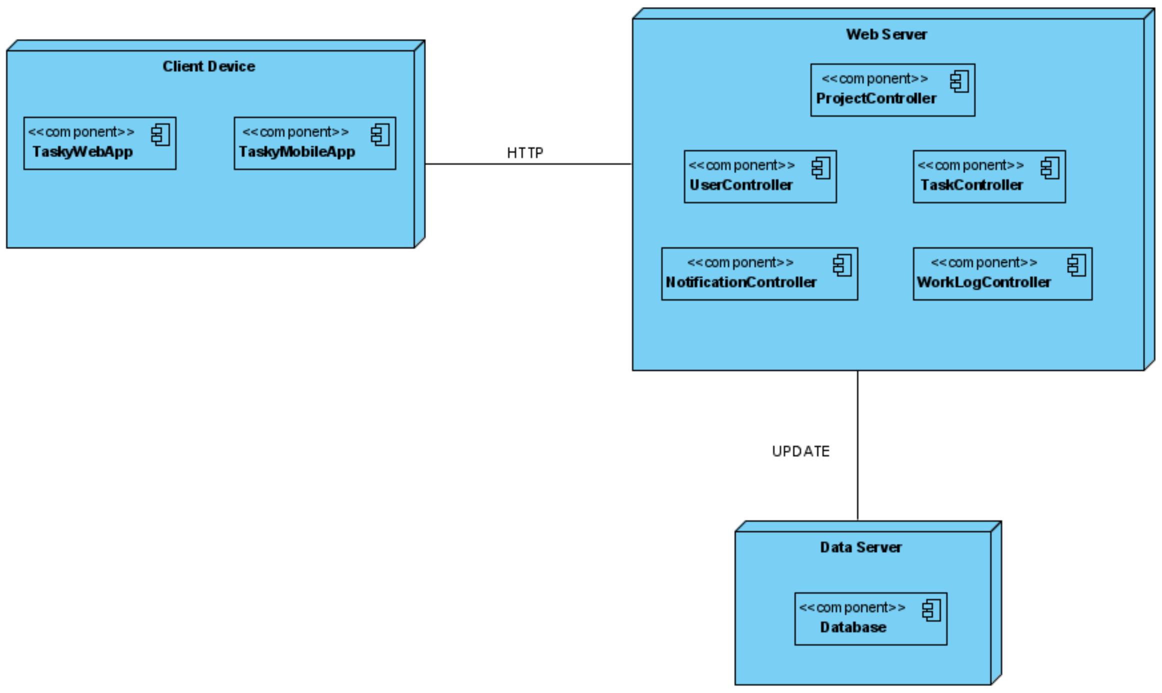
2.3 Hardware/Software Mapping

There are three hardware components in the system.

First one is client device. Client device is the device that our web or mobile application run. It contains the components of Interface layer. It can be a computer or a smart phone.

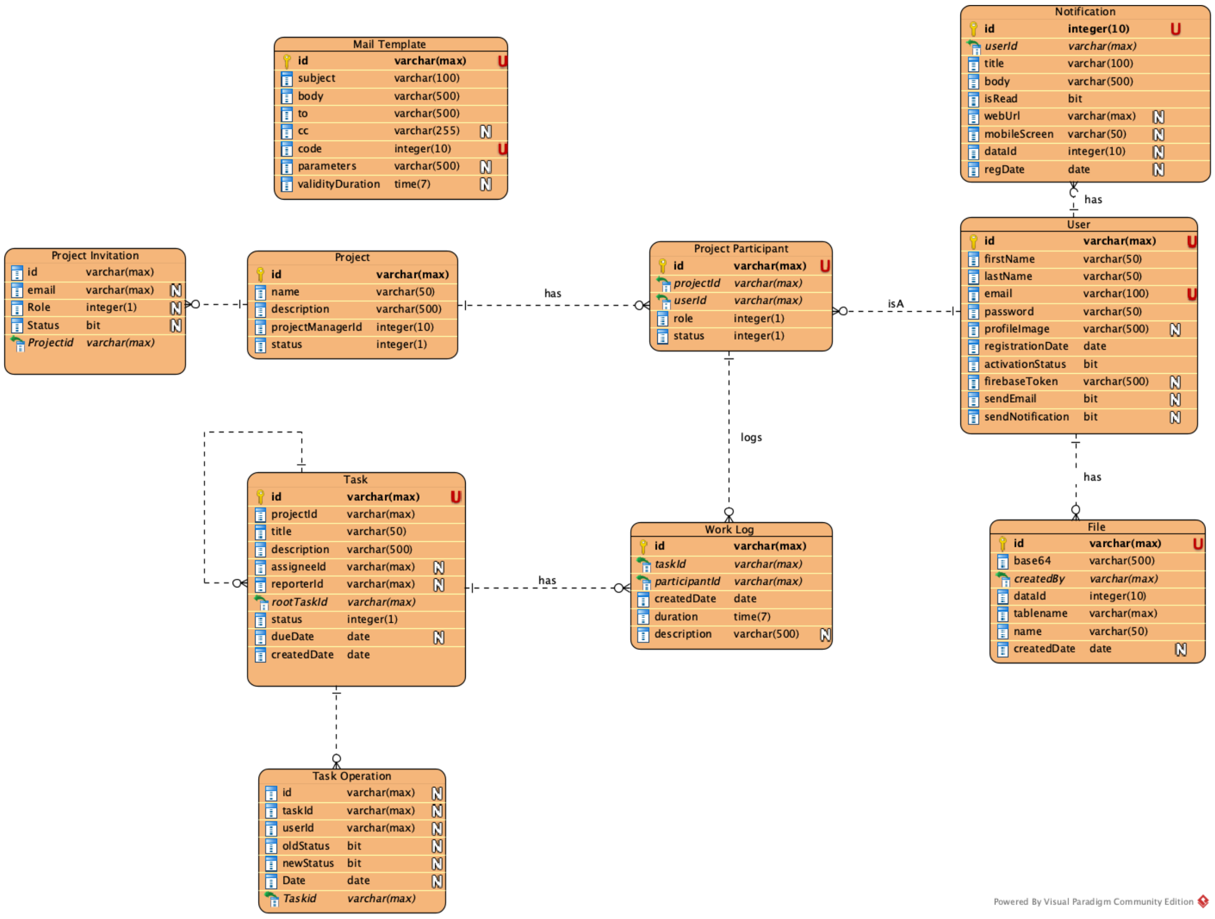
Second hardware component is web server. It contains the components of application logic layer. The operations are done in this hardware. And the

Final hardware component is data server. It contains the database component. All of the dynamic data of the system is stored in this hardware component.


Client Device
<<component>> TaskyWebApp
<<component>> TaskyMobileApp
HTTP
Web Server
<<component>> ProjectController
<<component>> UserController
<<component>> TaskController
<<component>> NotificationController
<<component>> WorkLogController
UPDATE
Data Server
<<component>> Database

2.4 Persistent Data Management

Data of the Tasky system is stored in relational database which is run on the Azure server. System will keep the backup of the database in order to prevent any data loss. The password of the users will be kept as encrypted for avoiding the security problems. There will be 12 different tables in the database.



2.5 Access Control and Security

Tasky system has two different user roles. There are registered users who is registered and have an account in the system and guests who has no account. But each user has different kinds of roles within the single project. Projects has team members, watchers and project manager. Access Control matrix is shown below. We have a role system in the client side of the web and mobile applications in order to avoiding security issues. Also, we use jwt token authentication in the server side. When the user logged in to the system, they get a jwt token and use this token in the web api calls. Each token has 1-hour expiration time.

	System	**User**	**Preference**	**Project**	**Task**	**Work Log**	**Notification**	**Mail**
System							create()	create()
Registered User	logout()	editProfile() viewProfile()	manage()	create()			receive()	receive()
Guest	login() register()							receive()

Project Manager				getDetail()
				getTaskList()
				getWorkLogs()
				getBoard()
				getFiles()
				update()
				delete()
				getReport()
				addParticipant()
				removeParticipant()
					getDetail()
					getFiles()
					getWorkLogs()
					create()
					update()
					delete()
						getDetail()
						getFiles()
						create()
						update()
						delete()		

Team Member				getDetail()
				getTaskList()
				getWorkLogs()
				getBoard()
				getFiles()
					getDetail()
					getFiles()
					getWorkLogs()
					create()
					update()
					delete()
						getDetail()
						getFiles()
						create()
						update()
						delete()		

Watcher				getDetail()
				getTaskList()
				getWorkLogs()
				getBoard()
				getFiles()
					getDetail()
					getFiles()
					getWorkLogs()
						getDetail()
						getFiles()		

2.6 Global Software Control

Tasky system has event-driven global control flow. Callback functions of the subsystems monitor the interactions of the user with the components. When an interaction occurred, this interaction is handled in the related subsystem. Interactions that are made at the same time by different users does not affect data consistency.

2.7 Boundary Conditions

Prerequisites

Users should have an internet connection in order to use Tasky system.

Start-up and shutdown

Start-up and login: When the user logged into the system, all necessary data related with the user will be shown in the Tasky mobile/web application.

Error Behaviour:

- If any error occurs because of the user, system will not allow processing of the operation.
- If any network failure occurs, user's connection to the system will be interrupted.

3. Subsystem Services

We have 3 layers and 8 subsystems in the proposed system architecture.

TaskyMobileApp and TaskyWebApp subsystems which are in the interface layer are represents the user interfaces in the mobile and web application. These subsystems use services provided by application logic layer subsystems.

Application Logic layer subsystems provides some services to the interface layer and use services provided by storage layer.

UserController subsystem provides following services:

- register()
- login()
- logout()
- editProfile()
- viewProfile()
- managePreferences()

ProjectController subsystem provides following services:

- createProject()
- updateProject()
- deleteProject()
- getProjectDetail()
- getWorkLogs()
- getBoard()
- getTaskList()
- getProjectReport()
- addParticipant()
- removeParticipant()
- getFiles()

TaskController subsystem provides following services:

- createTask()
- updateTask()
- deleteTask()
- getTaskDetail()
- getFiles()
- getWorkLogs()

WorkLogController subsystem provides following services:

- createWorkLog()
- updateWorkLog()
- deleteWorkLog()
- getWorkLogDetail()
- getFiles()

NotificationController subsystem provides following services:

- createNotification()
- sendNotification()

There is only database subsystem in storage layer. It provides a service to the application logic layer.

Database subsystem provides following services:

- getData()
- updateDatabase()

4. References

1. Bruegge B. & Dutoit A.H.. (2010). *Object-Oriented Software Engineering Using UML, Patterns, and Java*, Prentice Hall, 3rd ed.