VIA University College

SLIT

Heterogeneous system

Project Report - Project management tool
Third semester – Software Engineering

Supervisors	Students
Jakob Knop Rasmussen	Andrei Balanuta (281045)
Jan Otto Munch Pedersen	Lukas Vaisnoras (280107)

Characters with spaces
51,654

Contents

List of tables and figures

Abstract

The project's purpose is to develop a secure Project Management Tool for "Galaxis" Software Engineering team using 3-Tier Architecture design pattern.

SLIT Project management tool is a custom software for "Galaxis" team project manager that has the objective to optimize current team's workflow. In order to achieve working stream optimization, on the base of team's management and cooperation issues were stated the main requirements for this product. The requirements went through a long software development process that is described in this documentation.

The software has the purpose to ease project manager's work. Information management in a project is highly important and having a custom tool is necessary for implementing different tasks. The software provides various functionalities that include communication between team members, task management, liabilities overview and other project management features.

The 3-Tier architecture design was used in order to implement an application with three independent parts. Each tier has it's own purpose and configuration as well as it's own method of processing information. The system is built to be consistent and ready for future updates. Furthermore, software is using different security methods that went through all three steps : analysis, design and implementation. Using security protocols made software to avoid a big amount of threats and risks.

Additionally, development process has involved different tools and methods in order to achieve most of the requirements accomplishment. Even though it didn't pass all test cases, it's still a functional application that can be used for project management objectives.

Introduction

Nowadays, information is the most valuable resource on the planet and every day its weight is increasing exponentially in a way that humans cannot predict. People that work on various projects have certain milestones and assignments to be done which usually are used to measure their degree of fulfillment. Therefore, effective enterprise management is crucial for every company or team's economy in terms of money and time. Thus, in order to reach a strong operation control of project execution, various tools are created that boost information processing and labor management. Undoubtedly, in terms of information processing, computers are way more superior over humans. Since human brain has its own limits, technologies evolve faster as the time goes and using them as extension is the only way to overcome data management struggle. After all, task management is a time-consuming and rigorous business, thus not estimating it leads to various risks and complexity increment.

To continue with, project management tools are used every day around the world and people who use them achieve serious goals. Obviously, there are lots of tools on the market to choose from and selecting the right one can be a real contest. Without a doubt, when teams determine what tools to use, they search for one that conforms to their needs and requirements. However, rarely when a project management tool can automate most of the project work. Therefore, people came with a solution to use custom appliances that work specifically under their needs. In most cases, software developers are using lots of tools in order to achieve automation control especially when development process is complex and contains a lot of data.

The Software Engineering team named "Galaxis" is involved in different projects and every day they process and use lots of information for developing applications and delivering high quality services. Currently, the team in suffering due to lack of control and poor connectivity between members. Also, product manager struggle with team leading and task hierarchy regulation. Finding a proper tool exactly for their demands was an operation without success and a real difficulty.

Consequently, "Galaxis" project manager is researching how they can achieve better goals by improving management approach. Thus, a decision was made to obtain a custom project management tool in order to fix certain weak points. The project manager has stated that his team is in demand of a software that will serve as an instrument that will provide a clear overview or every project's insight. Also, the tool must improve the interconnection between team members and optimize project manager's every day workflow. Additionally, the tool should allow team members to work remotely on different tasks, this becomes powerful when there is no possibility to be physically in the office or some other places.

Since application is a project management tool, it is reliable only for project management purposes. Therefore, it won't provide any additional functionality other than described in the requirements section stated by the customer. The software will support only English language and won't provide any audio/video content since it's not a customer's demand.

The project purpose is to design and develop a distributed system that is secure and fulfills customer's requirements. A system that will help the project manager to coordinate tasks and help to plan and delegate work all in one or multiple channels using chat and task management functionality. In contrast, the system must be consistent and easy to expand using 3 Tier Architecture design.

Requirements

Generally, requirements are the key factor to understand customer's needs. Therefore, a list of functional and non-functional requirements was made in order to develop a well-functioning custom software. Importantly, each requirement represents a single functionality and has its unique purpose. As a result, they were used as cutoffs in project development process and also as a manual.

Functional requirements

1. As a user I want my account to be created on the base of my personal information (First name, last name, date of birth, profile picture, username, gender, password).
2. As a user I want to be able to log in/log out of the system.
3. As a user I want to be able to edit my personal information.
4. As a user I want to be able to create a project and become its administrator.
5. As an administrator I want to be able to invite people from my project by username.
6. As an administrator I want to be able to kick out members from my project.
7. As an administrator I want to be able to create/delete tasks.
8. As an administrator I want to be able to create/delete subtasks.
9. As an administrator I want to be able to set task priority by color labels.
10. As an administrator I want to assign project members to tasks.
11. As an administrator I want to set task's status as compete.
12. As a user I want to be able to leave a comment for a task.
13. As an administrator I want to create a channel for department's communication.
14. As an administrator I want to invite members to channel.
15. As an administrator I want to be able to kick out members from channel.
16. As a user I want to be able to send text messages, images, code snippets, docx, doc, pdf, xls, csv etc.
17. As a user I want to be able to chat privately with another user.
18. As a user I want to be able to see current progress of a task.
19. As a user I want to be notified about future tasks deadline, messages, project invitations.
20. As a user I want to have access to project tasks and message history.

Non-functional requirements

1. The system should be heterogeneous and can be accessed via browser.
2. Browser application must be responsive.
3. The system should be implemented in java and C# as a 3 Tier architecture design.
4. The system should store information in a MySQL Database.

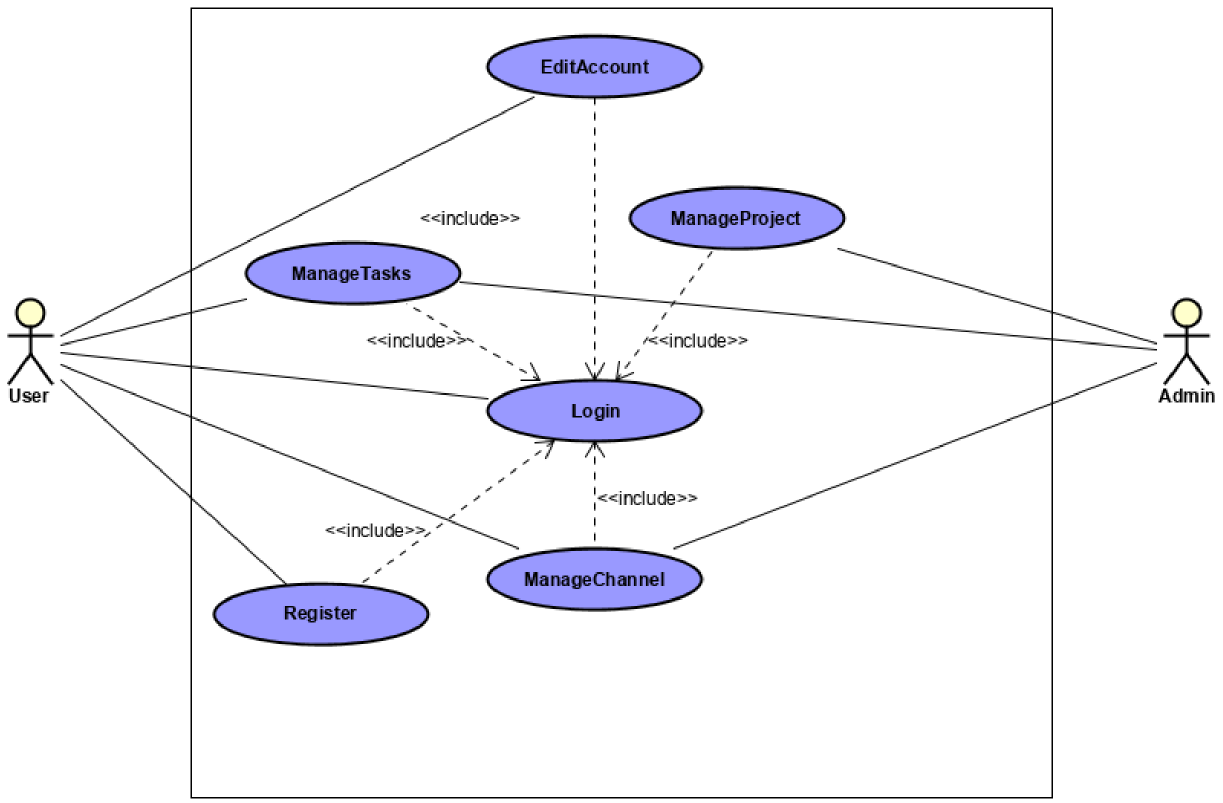
Analysis

This chapter has the purpose to explain the problem domain definition and to expand software requirements into different diagrams that define software functionality. Moreover, analysis content is highly important for next development process steps, thus focusing on this step is primordial in terms of understanding how users will consume the application in different situations. UML standard will be used for all below diagrams.

Use Case diagram

Use case diagram is serving as a behavioral diagram type, it is used to analyze system's actions in various scenarios. In addition, each use case shows a single path for achieving a certain goal, thus it's important not to underestimate each requirement's value in order to build a consistent use case diagram for later needs. The way users will interact with the system is highly essential to investigate considering requirement's linkage.

Figure 1 - Use case diagram (Appendix H)



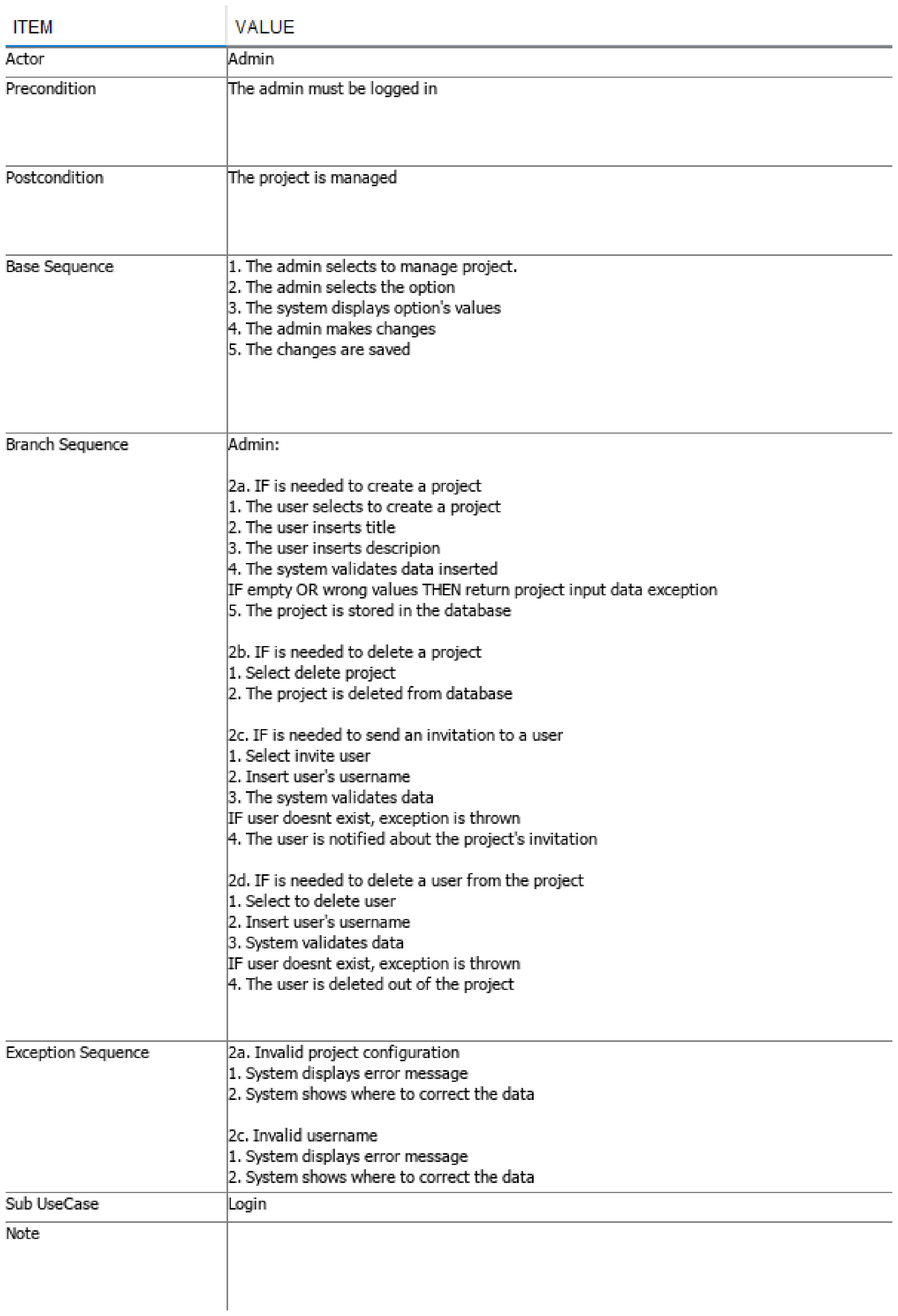
Considering the *Figure 1*, there are only two types of actors that have different functionalities. Also, can be seen that each use case includes login use case. This signifies that before performing a certain action, the user must be logged in for authorization objectives. It's important to mention that the user becomes an admin when the project is created and managing project process is started. Another important point is that when admin leaves the project – he becomes a simple user. Considering everything mentioned above, it's essential that admin has more functionality than a simple user while managing project and they both share some functionality. All in all, as mentioned that a use case is showing a single path for achieving a certain goal – it implies more sub functionality as described in next section.

Use case description

This section covers every scenario that can be performed in a particular use case.

Considering *Figure 2*, Manage Project use case involves different options and capabilities for all project admins that wish to create a project. First use case scenario describes what actions are necessary to perform in order to create a project. Importantly, the project will not be created if a step is not performed or throws an error as a result of system validation described in exception sequence. The same principle is followed by the rest use case scenarios. Also, it's mentioned in the diagram that the admin must be logged in before executing any use case scenarios.

Figure 2 - Use case description (Appendix C)



ITEM	VALUE
Actor	Admin
Precondition	The admin must be logged in
Postcondition	The project is managed
Base Sequence	1. The admin selects to manage project. 2. The admin selects the option 3. The system displays option's values 4. The admin makes changes 5. The changes are saved
Branch Sequence	Admin: 2a. IF is needed to create a project 1. The user selects to create a project 2. The user inserts title 3. The user inserts descripion 4. The system validates data inserted IF empty OR wrong values THEN return project input data exception 5. The project is stored in the database 2b. IF is needed to delete a project 1. Select delete project 2. The project is deleted from database 2c. IF is needed to send an invitation to a user 1. Select invite user 2. Insert user's username 3. The system validates data IF user doesnt exist, exception is thrown 4. The user is notified about the project's invitation 2d. IF is needed to delete a user from the project 1. Select to delete user 2. Insert user's username 3. System validates data IF user doesnt exist, exception is thrown 4. The user is deleted out of the project
Exception Sequence	2a. Invalid project configuration 1. System displays error message 2. System shows where to correct the data 2c. Invalid username 1. System displays error message 2. System shows where to correct the data
Sub UseCase	Login
Note	

Figure 3 - System activity diagram (Appendix G)

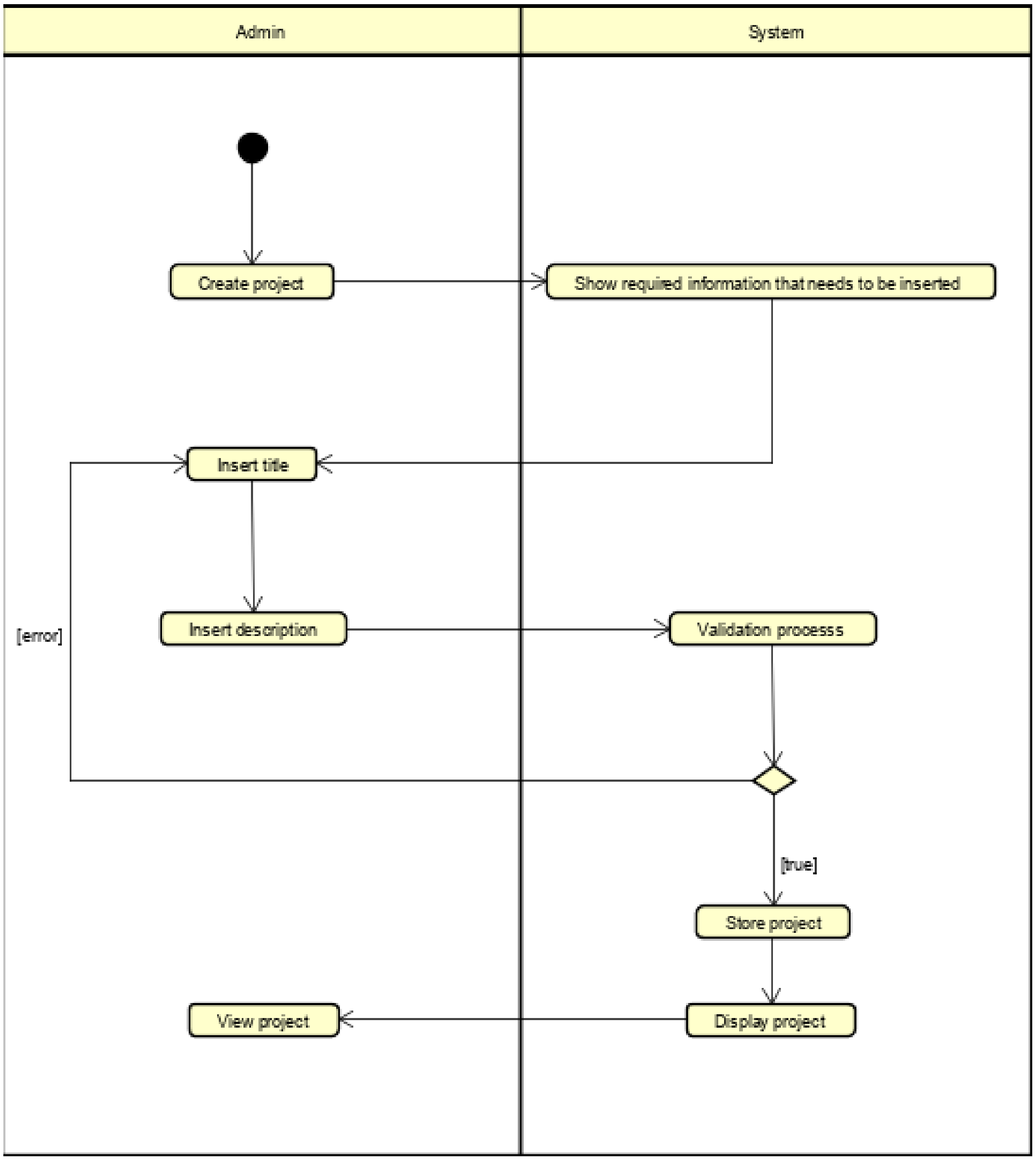
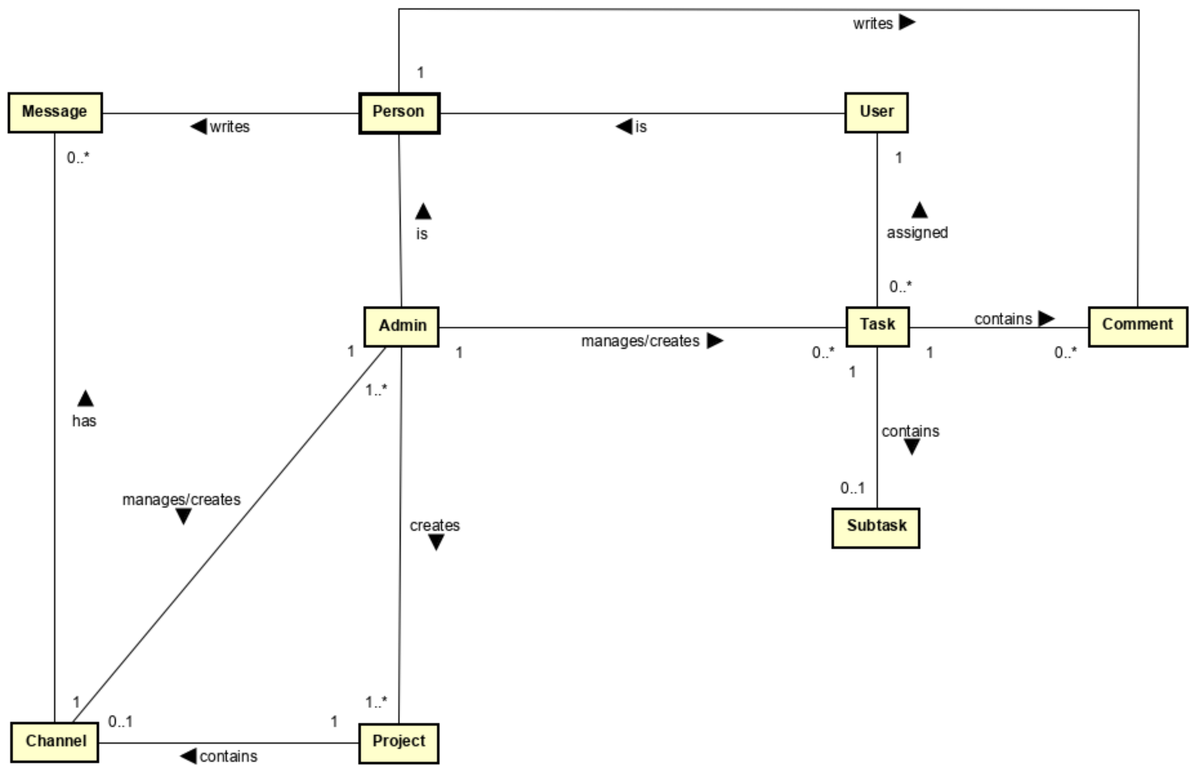


Figure 3 - shows all steps that are performed when creating a new project. The left part is the user's perspective and the right part shows the systems behavior based on user's actions. As shown above, if a user wants to create a project – system will request a title and a description. Therefore, depending on user's input, the validation outcome will be affected.

Domain Model

Domain model is a representation of conceptual classes and relationships between them. As it's almost the final part of the analysis chapter, this diagram must serve as a basis for a class diagram development. As a result, *Figure 4* diagram must show every object's atomic value and connection with other objects.

Figure 4 - Domain Model (Appendix E)



At this point, all the diagrams designed so far can lead to next development process phase which is Design part. But before that, it's required to analyze the security aspects of this project which are described in the next section.

Security

Introduction

This chapter aims to emphasize security investigation topics that will further lead to design phase of the project development. Undoubtedly, software security is remarkably important in relation to protecting user’s sensitive data. Nowadays, there are lots of applications running on the internet and this significantly increases hackers curiosity. It should be noted that even huge companies are not safe as their existing software has its own vulnerabilities even if it is using latest security techniques and technologies. Namely, every day hackers perform cyber-attacks on different systems for information stealing purposes. Altogether, security matter has to be a must in software development and devoting a proficient research to this section can prevent a certain number of cyber-attacks.

Threat Hunting

Threat hunting is a modern methodology to discover and narrow threats . Also, it is a defensive strategy used for analysis matter in order to generate some protection methods in respect for users that consume software services. Firstly, a threat model is created in order to specify which possible threats are going to be prevented.

Threat Model

To begin with, threat modeling is a process of finding vulnerabilities that can cause dangerous impacts to the software. Moreover, it also helps development team to perform critical judgements about what is needed to concern about.

STRIDE

STRIDE threat model is the clustering process of possible threats in order to find out what are the main risks that the system is going to face.

Spoofing identity is when attacker is using another user’s identity. For instance, an attacker is acting like another user or as an administrator.

Tempering is when attacker changes some information in the system. For example, an attacker changes someone’s credentials or something else in the database.

Repudiation is when attacker is clearing existing footprints in order not to be busted. Example, an attacker is deleting system logs.

Information disclosure is when attacker steals illegally some information for profit purposes. For example, an attacker is stealing credit card information and sells it on the dark web.

Denial of service is when attacker denies access to system for other users. For instance, spamming a system with useless traffic.

Elevation of privilege is when attacker has more rights in the system than he is supposed to have. An example is when attacker gains some additional functionalities that are not included in the users policy.

EINOO – Where are we attacked, and by whom

Thinking about who is attacking is very useful in terms of analyzing future consequences of an attack.

External attackers - non legit system users.

Insiders - attackers that are registered as users and have the minimum access of software functionality.

Further is determination of where the attacker is going to hit.

Network attacks - where attacker is modifying network traffic. This type of attack is nearly impossible to see if attempted and the common solution is cryptography.

Offline attacks - when attacker has access to information that is stored permanently in the system and tries to steal/modify it. Usually, offline attacks are way harder to perform because of braking system's access control requirement.

What means are used in the attack: X.800

The classification below follows X.800 standard.

The X.800 standard describes two main types of attacks on the network. The first category is called *passive attack* and another is *active attack.* Both of them have some branching, for instance, passive attacks are divided in two categories. First, *eavesdropping* is when the attacker is listening to the network and sees the information flow. Second, *traffic analysis* is when attacker is looking at who sent the information and how much is sent. Active attacks are of three types. *Replay* is when old data is resent, *blocking* when data is blocked from arriving to the destination. Additionally, *modification* is the last type and happens when attacker changes data context.

Altogether, detecting passive attacks is harder to detect due to less interaction. Traffic encryption can prevent this type of attack but even so the attacker is not stopped from trying to analyze encrypted traffic. On the other hand, active attacks are easier to discover but harder to prevent. Namely, to prevent this type of attack is first to focus on detecting it using authentication codes, signatures and other techniques.

Objectives statement

Namely, security objectives can be described using CIA triad as it describes software's security policy. The CIA principle is appliable throughout the whole matter of security analysis. With this in mind, this principle follows three fundamental standards and if one of them is violated it can lead to serious consequences.

Confidentiality Is the power of masking data from people that are not authorized to see it. Keeping data private is essential for every present software and being ensured that those who are unauthorized do not have access to specific assets makes system superior in respect to hackers. For instance, users that provide sensitive data such as credit card, personal address etc. hope that this information is flowing through the internet in a secure way and also stored securely.

Integrity is the ability to keep data unchanged while it is processed by the system. The threats such as system bugs and errors can cause data modification and violation of this standard. One common solution is to use a version control to stop unexpected errors/bugs and data altering. Additionally, this also involves setting user access controls for a better data management. Importantly, in case of a random system crash, all data must be recoverable, and system must provide stable backup solutions.

Availability is the aptitude of making regular software updates or other upgrades. Being sure that system works without warfare is crucial for every software security policy. Importantly, rotten communication between user and system along with a bad bandwidth connection can cause data loss or serious impacts. To prevent these threats, the proxy servers or backup copies along with other tools are used.

Incident likelihood + Incident consequence = Risk

To begin with, risk Is the likelihood change of an event that may have either positive or negative consequences. Obviously, some risks are harder to predict than others and in order to prevent a big amount of them, it's needed to understand the core concepts of it by discussing it's building blocks.

Incident likelihood is the sum of threat frequency and preventive measures or can be described as threat frequency related to how threat is going to be prevented. In contrast, it is a measurement of the probability that a threat will arise in specific preventive circumstances. On the other hand, Incident consequence is the sum of a threat effect and its corrective measures. In other words, it's the path of recovering after a threat did its effect. Therefore, a risk is the sum of incident likelihood and incident consequence. That being said, the definition above is confirmed and the next step is the risk assessment model.

Table 1 - Risk Assessment Table

Threat	Vulnerability	Consequences	Risk	Solution
System bugs	Functionality provided can be used internally	Unauthorized data access, data loss, CIA triad violation	High	High quality testing
SQL Injection	Unauthorized access to database	Data loss, data modification, data steal	High	Escaping inputs, mapping objects
Non expected user behavior	Exposed data	Data steal	Medium	Clear user guide
SYN flooding	Denial of service	Software access denial	High	Reverse proxy
Identity spoof	Attacker has access to	Unauthorized data access	Medium	Follow CIA triad standard

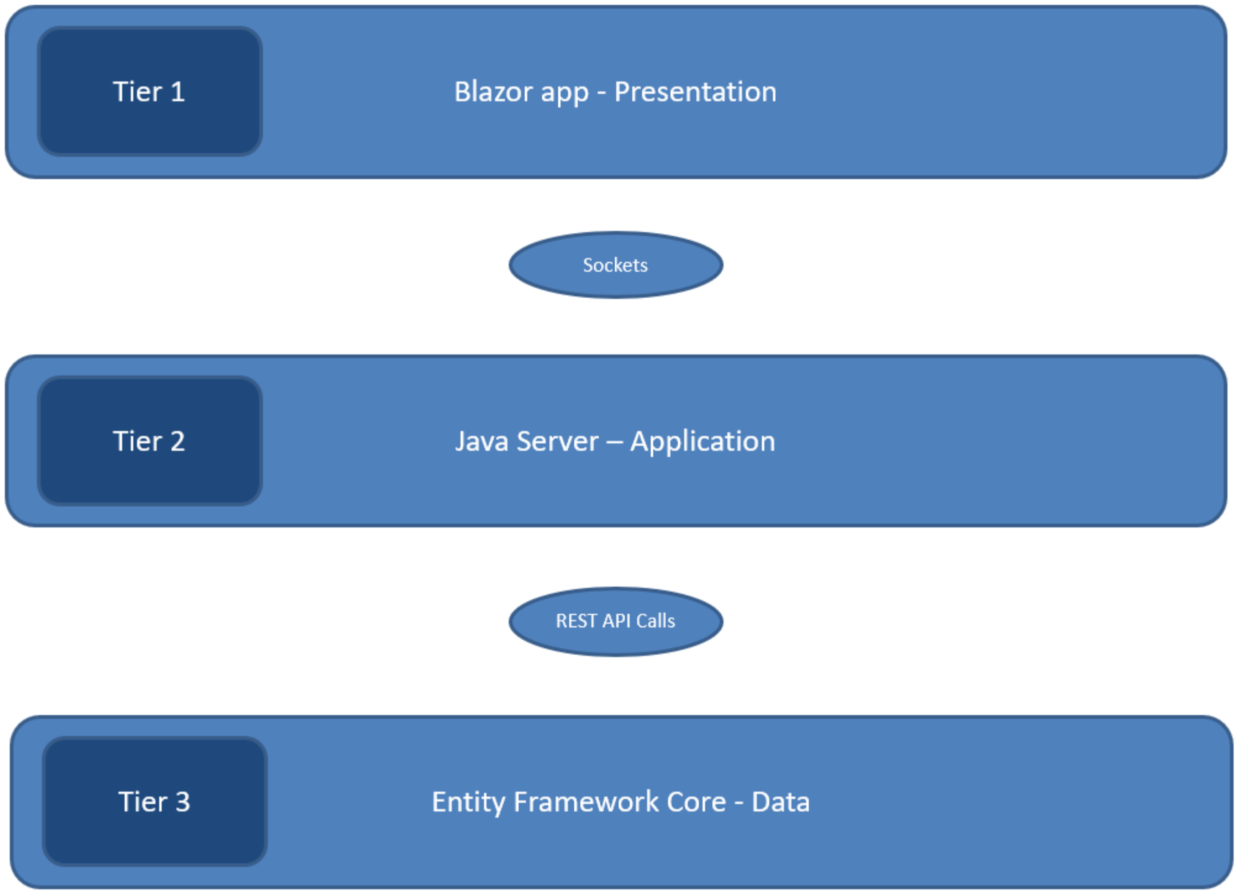
Design

Design Patterns

Architecture

Software developed is using 3-Tier Architecture pattern. Reason for choosing is the layered pattern structure that is easy for future extensive development, scalability, performance and physical independence. Thus, as shown in *Figure 5*, separating the entire application in three tiers makes development team divide themselves and focus on different parts of the application in order to speed up the development process.

Figure 5 - Architecture Diagram (Appendix B)


Tier 1
Blazor app - Presentation
Sockets
Tier 2
Java Server – Application
REST API Calls
Tier 3
Entity Framework Core - Data

The first, *presentation* tier is the front-end layer that includes the UI. In addition, it's on the top of the architecture design and it displays information to the user. This tier communicates with the second tier and sends requests that are processed further and the request received is showed on the user application screen.

The second, *application* tier is responsible for functional business logic of the application that perform application's central capabilities and acts as a server for client requests. To clarify, this tier processes the information send from the presentation tier in order to send a specific request next to data layer whether it's to receive a piece of data or fetch.

The third, *data* tier is responsible for receiving requests from business logic tier, storing/deleting information from database and sending data back to the second tier. For this layer was used .NET Entity Framework Core because of its ability of easy data manipulation.

DTO – Data transfer object

This section outlines a part of communication between tiers and how they are interconnected with each other in terms of information transfer. To begin with, a decision was made to create a data transfer object called Message and another class called Fields in order to be sent as a request through tiers. This object contains all the information needed for a request and also some additional information about request metadata. Furthermore, because of heterogeneity and language difference between tiers, this object is made in a way that all three tiers understand and can process it.

Figure 6 - C# Message Object Class Diagram (Appendix I)

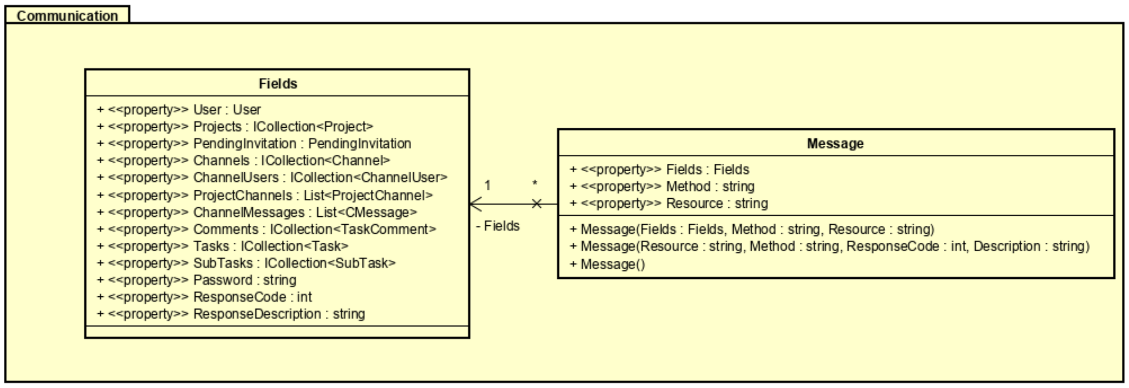
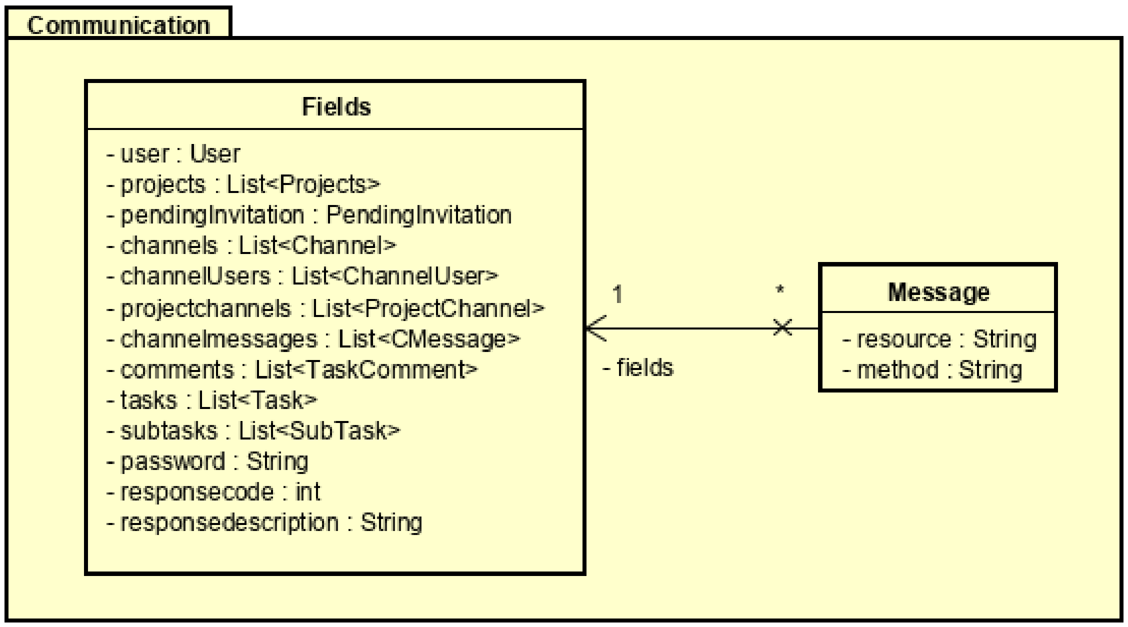

Communication
Fields
+ <<property>> User : User
+ <<property>> Projects : ICollection<Project>
+ <<property>> PendingInvitation : PendingInvitation
+ <<property>> Channels : ICollection<Channel>
+ <<property>> ChannelUsers : ICollection<ChannelUser>
+ <<property>> ProjectChannels : List<ProjectChannel>
+ <<property>> ChannelMessages : List<CMessage>
+ <<property>> Comments : ICollection<TaskComment>
+ <<property>> Tasks : ICollection<Task>
+ <<property>> SubTasks : ICollection<SubTask>
+ <<property>> Password : string
+ <<property>> ResponseCode : int
+ <<property>> ResponseDescription : string
1
*
- Fields
Message
+ <<property>> Fields : Fields
+ <<property>> Method : string
+ <<property>> Resource : string
+ Message(Fields : Fields, Method : string, Resource : string)
+ Message(Resource : string, Method : string, ResponseCode : int, Description : string)
+ Message()

Figure 7 - Java Message Object Class Diagram (Appendix I)


Communication
Fields
- user : User
- projects : List<Projects>
- pendingInvitation : PendingInvitation
- channels : List<Channel>
- channelUsers : List<ChannelUser>
- projectchannels : List<ProjectChannel>
- channelmessages : List<CMessage>
- comments : List<TaskComment>
- tasks : List<Task>
- subtasks : List<SubTask>
- password : String
- responsecode : int
- responsedescription : String
1
*
- fields
Message
- resource : String
- method : String

As shown in *Figure 6 and 7*, in order not to have conflicts with object deserialization both objects must share same object types and follow the same pattern. Message object has resource field for specifying the resource that needs to be processed/requested, method field for operation indication and fields variable with other project related fields that are required to be sent or received. Message object is serialized and sent as a JSON format.

System sequence diagrams

A sequence diagram is an interactivity representation of object groups in a system. Metaphorically speaking, each lifeline shows the process of object's birth giving. Additionally, it portrays how information flows through different parts of a system and depending on how information is processed and validated – the information returned to the user will be influenced.

Figure 8 - System sequence diagram (Appendix D)

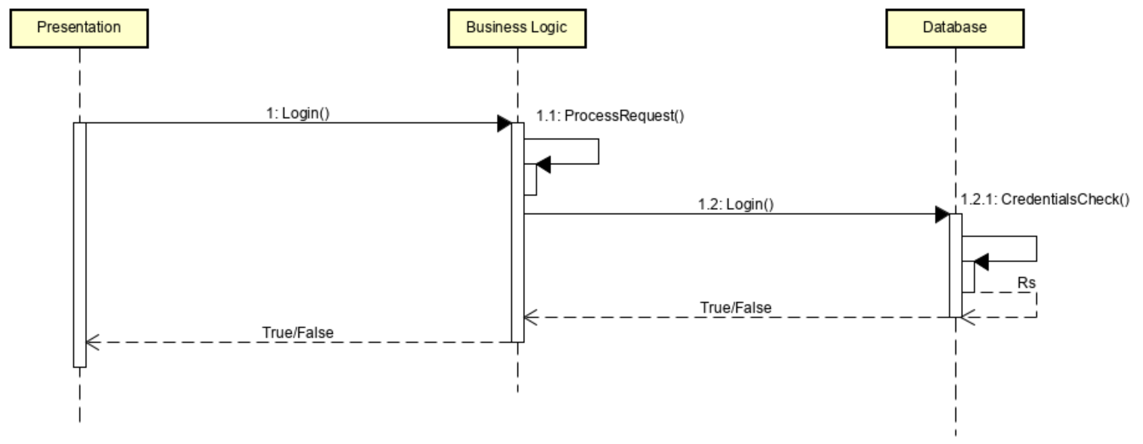


Figure 8 shows the login functionality and lifelines involved in the execution process. Throughout execution, each layer is responsible for different information validation and has its own logic in order to return a specific piece of information. It is observed that credentials are sent first to the business logic layer and depending on how information is processed a specific request will be sent to the database layer. As a result, if credentials are correct – the user will be logged into the system.

Figure 9 - System sequence diagram (Appendix D)

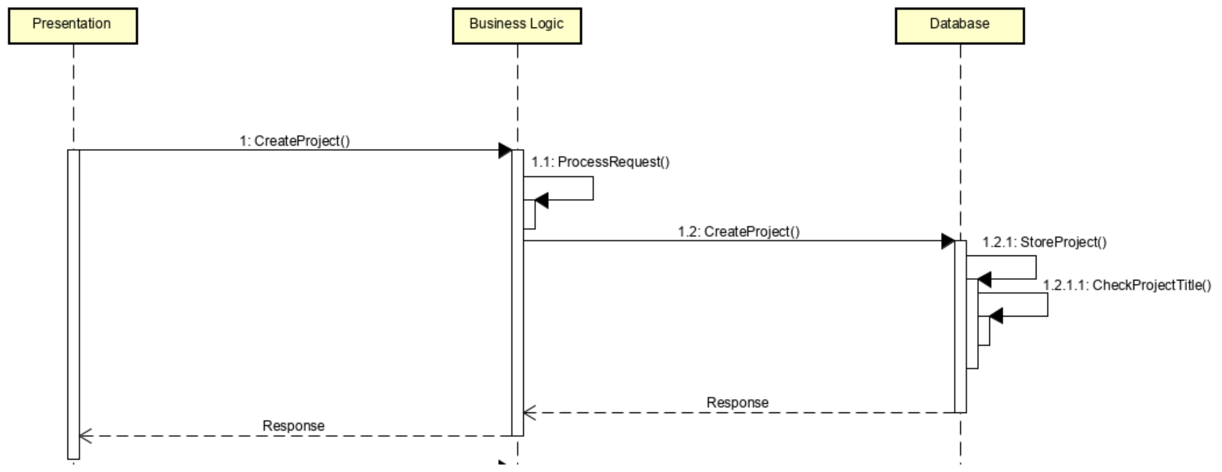


Figure 9 shows the create project method lifecycle. As mentioned above, the information is processed throughout layers and specific request is sent to the database. As a result, database checks if there is already a project named with the same title. If validation process goes successfully – user will create a project and will become its admin.

Class Diagram

This section underlines the class diagrams which shows a prototype of the application's static view.

Presentation tier

Figure 10 - Presentation Controllers Diagram (Appendix I)

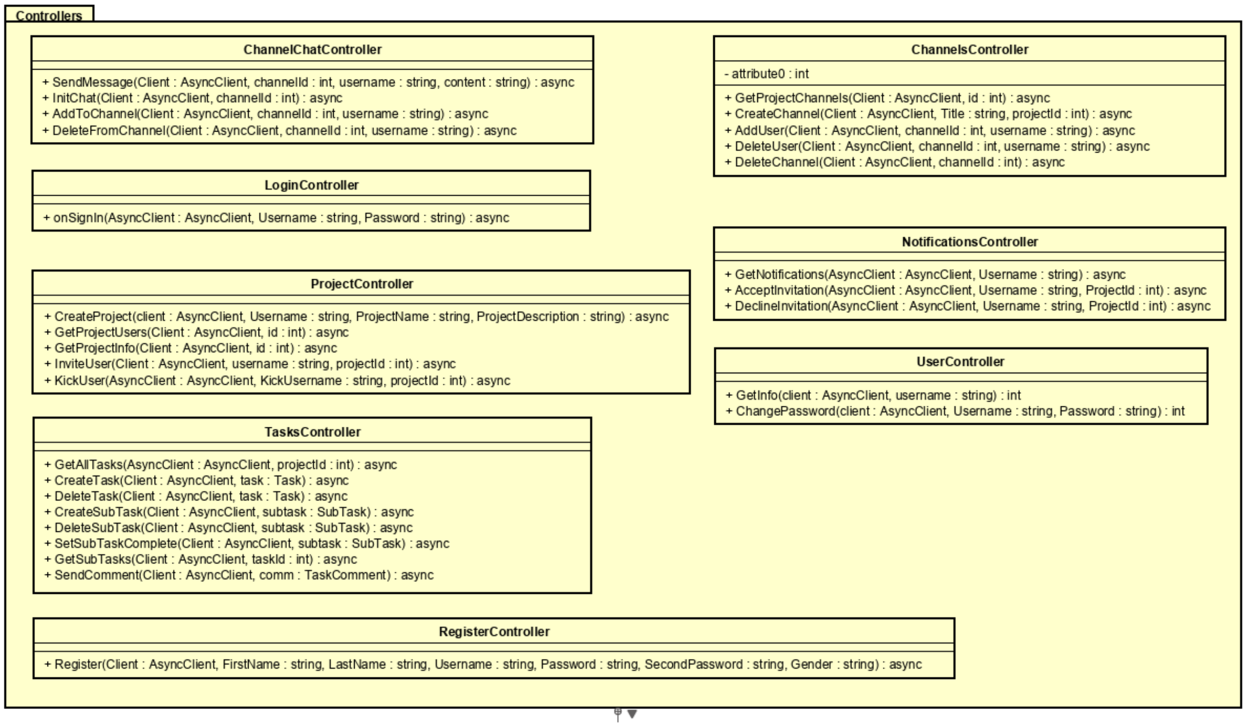

Controllers
ChannelChatController
+ SendMessage(Client : AsyncClient, channelId : int, username : string, content : string) : async
+ InitChat(Client : AsyncClient, channelId : int) : async
+ AddToChannel(Client : AsyncClient, channelId : int, username : string) : async
+ DeleteFromChannel(Client : AsyncClient, channelId : int, username : string) : async
ChannelsController
- attribute0 : int
+ GetProjectChannels(Client : AsyncClient, id : int) : async
+ CreateChannel(Client : AsyncClient, Title : string, projectId : int) : async
+ AddUser(Client : AsyncClient, channelId : int, username : string) : async
+ DeleteUser(Client : AsyncClient, channelId : int, username : string) : async
+ DeleteChannel(Client : AsyncClient, channelId : int) : async
LoginController
+ onSignIn(AsyncClient : AsyncClient, Username : string, Password : string) : async
NotificationsController
+ GetNotifications(AsyncClient : AsyncClient, Username : string) : async
+ AcceptInvitation(AsyncClient : AsyncClient, Username : string, ProjectId : int) : async
+ DeclineInvitation(AsyncClient : AsyncClient, Username : string, ProjectId : int) : async
ProjectController
+ CreateProject(client : AsyncClient, Username : string, ProjectName : string, ProjectDescription : string) : async
+ GetProjectUsers(Client : AsyncClient, id : int) : async
+ GetProjectInfo(Client : AsyncClient, id : int) : async
+ InviteUser(Client : AsyncClient, username : string, projectId : int) : async
+ KickUser(AsyncClient : AsyncClient, KickUsername : string, projectId : int) : async
UserController
+ GetInfo(client : AsyncClient, username : string) : int
+ ChangePassword(client : AsyncClient, Username : string, Password : string) : int
TasksController
+ GetAllTasks(AsyncClient : AsyncClient, projectId : int) : async
+ CreateTask(Client : AsyncClient, task : Task) : async
+ DeleteTask(Client : AsyncClient, task : Task) : async
+ CreateSubTask(Client : AsyncClient, subtask : SubTask) : async
+ DeleteSubTask(Client : AsyncClient, subtask : SubTask) : async
+ SetSubTaskComplete(Client : AsyncClient, subtask : SubTask) : async
+ GetSubTasks(Client : AsyncClient, taskId : int) : async
+ SendComment(Client : AsyncClient, comm : TaskComment) : async
RegisterController
+ Register(Client : AsyncClient, FirstName : string, LastName : string, Username : string, Password : string, SecondPassword : string, Gender : string) : async

Figure 10 shows all controller classes of the presentation tier. Each controller is responsible for every user's action performed – resulting a request creation and information sending to second tier as well as retrieving, deserializing and information transfer to the UI files. Important thing to mention is that these objects use dependency injection design pattern, therefore there are no tight relationships between them and objects that work on top of the software.

Business logic tier

Figure 11 - Business logic Class Diagram (Appendix I)

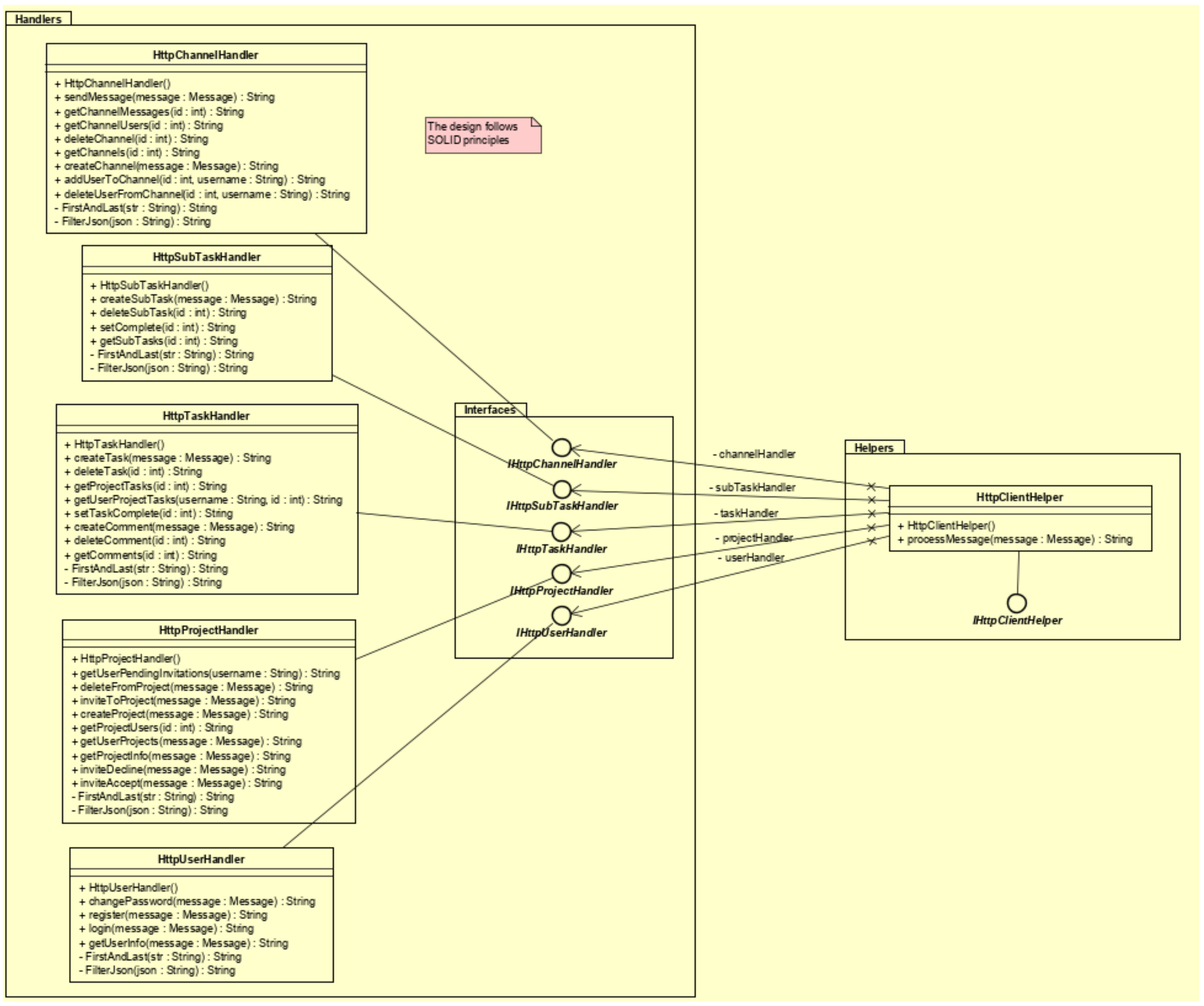
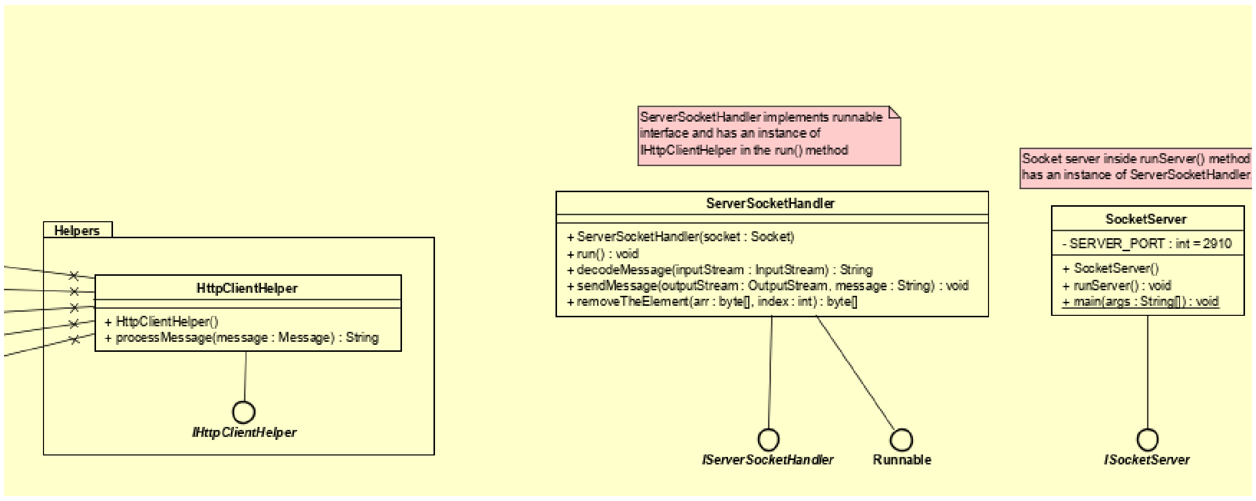

Handlers
HttpChannelHandler
+ HttpChannelHandler()
+ sendMessage(message : Message) : String
+ getChannelMessages(id : int) : String
+ getChannelUsers(id : int) : String
+ deleteChannel(id : int) : String
+ getChannels(id : int) : String
+ createChannel(message : Message) : String
+ addUserToChannel(id : int, username : String) : String
+ deleteUserFromChannel(id : int, username : String) : String
- FirstAndLast(str : String) : String
- FilterJson(json : String) : String
The design follows SOLID principles
HttpSubTaskHandler
+ HttpSubTaskHandler()
+ createSubTask(message : Message) : String
+ deleteSubTask(id : int) : String
+ setComplete(id : int) : String
+ getSubTasks(id : int) : String
- FirstAndLast(str : String) : String
- FilterJson(json : String) : String
HttpTaskHandler
+ HttpTaskHandler()
+ createTask(message : Message) : String
+ deleteTask(id : int) : String
+ getProjectTasks(id : int) : String
+ getUserProjectTasks(username : String, id : int) : String
+ setTaskComplete(id : int) : String
+ createComment(message : Message) : String
+ deleteComment(id : int) : String
+ getComments(id : int) : String
- FirstAndLast(str : String) : String
- FilterJson(json : String) : String
HttpProjectHandler
+ HttpProjectHandler()
+ getUserPendingInvitations(username : String) : String
+ deleteFromProject(message : Message) : String
+ inviteToProject(message : Message) : String
+ createProject(message : Message) : String
+ getProjectUsers(id : int) : String
+ getUserProjects(message : Message) : String
+ getProjectInfo(message : Message) : String
+ inviteDecline(message : Message) : String
+ inviteAccept(message : Message) : String
- FirstAndLast(str : String) : String
- FilterJson(json : String) : String
HttpUserHandler
+ HttpUserHandler()
+ changePassword(message : Message) : String
+ register(message : Message) : String
+ login(message : Message) : String
+ getUserInfo(message : Message) : String
- FirstAndLast(str : String) : String
- FilterJson(json : String) : String
Interfaces
IHttpChannelHandler
IHttpSubTaskHandler
IHttpTaskHandler
IHttpProjectHandler
IHttpUserHandler
- channelHandler
- subTaskHandler
- taskHandler
- projectHandler
- userHandler
Helpers
HttpClientHelper
+ HttpClientHelper()
+ processMessage(message : Message) : String
IHttpClientHelper

Figure 11 is showing HTTP handlers relationships, in order to have a consistent structure – the SOLID principles were followed. Each handler has its own responsibility for sending specific HTTPS requests to the Data tier. As shown, every handler implements a specific interface in order not to have strongly tied relationships between objects.

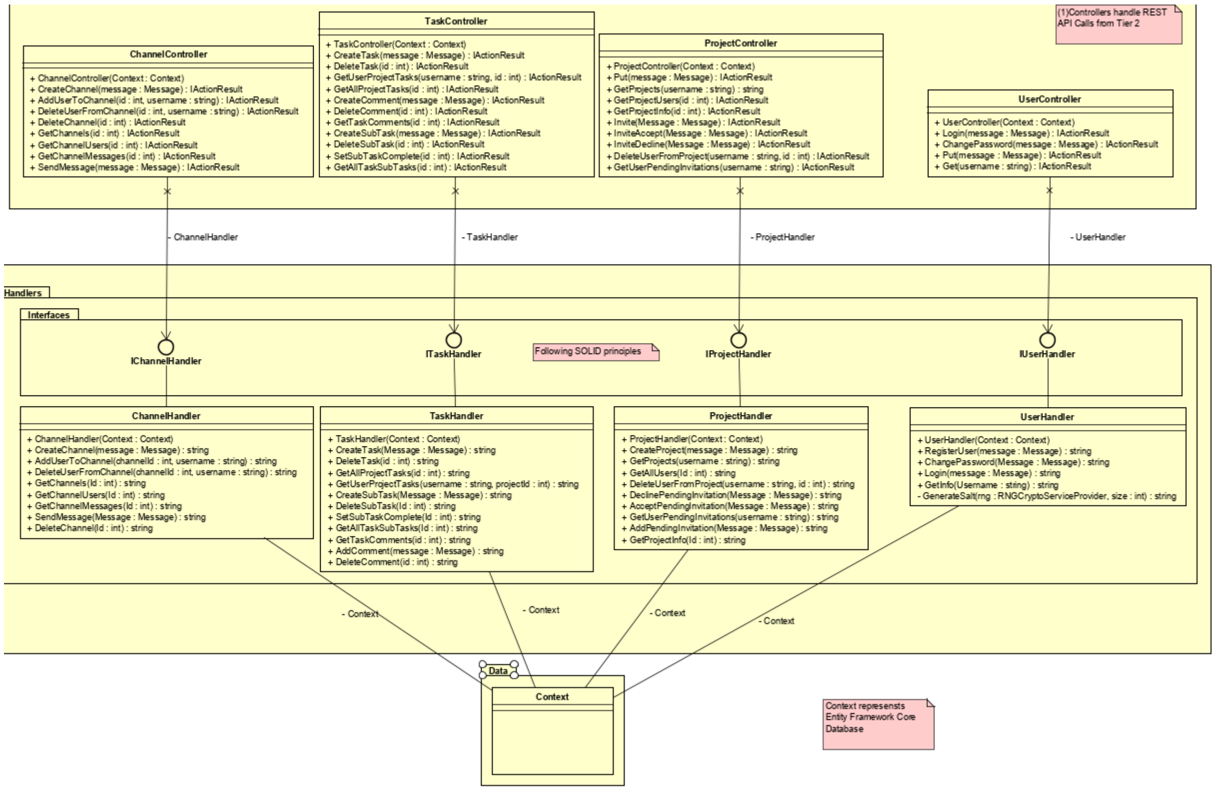
Figure 12 - Business Logic Continuation (Appendix I)


ServerSocketHandler implements runnable interface and has an instance of IHttpClientHelper in the run() method
Socket server inside runServer() method has an instance of ServerSocketHandler
Helpers
HttpClientHelper
+ HttpClientHelper()
+ processMessage(message : Message) : String
IHttpClientHelper
ServerSocketHandler
+ ServerSocketHandler(socket : Socket)
+ run() : void
+ decodeMessage(inputStream : InputStream) : String
+ sendMessage(outputStream : OutputStream, message : String) : void
+ removeTheElement(arr : byte[], index : int) : byte[]
IServerSocketHandler
Runnable
SocketServer
- SERVER_PORT : int = 2910
+ SocketServer()
+ runServer() : void
+ main(args : String[]) : void
ISocketServer

In *Figure 12* it's seen that business logic tier acts as a socket server in relation to the presentation tier. Its purpose is to receive information from the first tier and send it for further processing.

Data tier

Figure 13 - Data tier Class Diagram (Appendix I)


ChannelController
+ ChannelController(Context : Context)
+ CreateChannel(message : Message) : IActionResult
+ AddUserToChannel(id : int, username : string) : IActionResult
+ DeleteUserFromChannel(id : int, username : string) : IActionResult
+ DeleteChannel(id : int) : IActionResult
+ GetChannels(id : int) : IActionResult
+ GetChannelUsers(id : int) : IActionResult
+ GetChannelMessages(id : int) : IActionResult
+ SendMessage(message : Message) : IActionResult
TaskController
+ TaskController(Context : Context)
+ CreateTask(message : Message) : IActionResult
+ DeleteTask(id : int) : IActionResult
+ GetUserProjectTasks(username : string, id : int) : IActionResult
+ GetAllProjectTasks(id : int) : IActionResult
+ CreateComment(message : Message) : IActionResult
+ DeleteComment(id : int) : IActionResult
+ GetTaskComments(id : int) : IActionResult
+ CreateSubTask(message : Message) : IActionResult
+ DeleteSubTask(id : int) : IActionResult
+ SetSubTaskComplete(id : int) : IActionResult
+ GetAllTaskSubTasks(id : int) : IActionResult
ProjectController
+ ProjectController(Context : Context)
+ Put(message : Message) : IActionResult
+ GetProjects(username : string) : string
+ GetProjectUsers(id : int) : IActionResult
+ GetProjectInfo(id : int) : IActionResult
+ Invite(Message : Message) : IActionResult
+ InviteAccept(Message : Message) : IActionResult
+ InviteDecline(Message : Message) : IActionResult
+ DeleteUserFromProject(username : string, id : int) : IActionResult
+ GetUserPendingInvitations(username : string) : IActionResult
(1)Controllers handle REST API Calls from Tier 2
UserController
+ UserController(Context : Context)
+ Login(message : Message) : IActionResult
+ ChangePassword(message : Message) : IActionResult
+ Put(message : Message) : IActionResult
+ Get(username : string) : IActionResult
- ChannelHandler
- TaskHandler
- ProjectHandler
- UserHandler
Handlers
Interfaces
IChannelHandler
ITaskHandler
Following SOLID principles
IProjectHandler
IUserHandler
ChannelHandler
+ ChannelHandler(Context : Context)
+ CreateChannel(message : Message) : string
+ AddUserToChannel(channelId : int, username : string) : string
+ DeleteUserFromChannel(channelId : int, username : string) : string
+ GetChannels(Id : int) : string
+ GetChannelUsers(Id : int) : string
+ GetChannelMessages(Id : int) : string
+ SendMessage(Message : Message) : string
+ DeleteChannel(Id : int) : string
TaskHandler
+ TaskHandler(Context : Context)
+ CreateTask(Message : Message) : string
+ DeleteTask(id : int) : string
+ GetAllProjectTasks(id : int) : string
+ GetUserProjectTasks(username : string, projectId : int) : string
+ CreateSubTask(Message : Message) : string
+ DeleteSubTask(Id : int) : string
+ SetSubTaskComplete(Id : int) : string
+ GetAllTaskSubTasks(Id : int) : string
+ GetTaskComments(id : int) : string
+ AddComment(message : Message) : string
+ DeleteComment(id : int) : string
ProjectHandler
+ ProjectHandler(Context : Context)
+ CreateProject(message : Message) : string
+ GetProjects(username : string) : string
+ GetAllUsers(Id : int) : string
+ DeleteUserFromProject(username : string, id : int) : string
+ DeclinePendingInvitation(Message : Message) : string
+ AcceptPendingInvitation(Message : Message) : string
+ GetUserPendingInvitations(username : string) : string
+ AddPendingInvitation(Message : Message) : string
+ GetProjectInfo(Id : int) : string
UserHandler
+ UserHandler(Context : Context)
+ RegisterUser(message : Message) : string
+ ChangePassword(Message : Message) : string
+ Login(message : Message) : string
+ GetInfo(Username : string) : string
- GenerateSalt(rng : RNGCryptoServiceProvider, size : int) : string
- Context
- Context
- Context
- Context
Data
Context
Context represents Entity Framework Core Database

In *Figure 13* its shown that besides controllers, this tier also has some handler classes. In order to design in a consistent way – single responsibility rule was followed. Controller classes are responsible for exposing web services in a restful way so that object receives the information and sends it further to the handler object that is accountable for data storage or retrieval. Hence data tier is responsible for data management and storage, the context class is the object performing that and it's managed mostly by Entity Framework Core.

Security

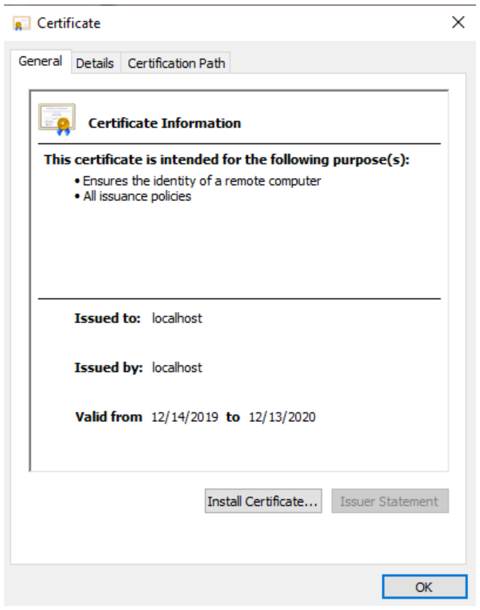
Introduction

The last section of design is devoted to security as it's primordial for a software deployment as mentioned in the analysis part. The outcome of security analysis was a number of models and methodologies in order to identify certain threats that might harm users as well as the software itself. The design part of security will cover all mechanisms that must be used in order to reduce overall risks.

HTTPS (Hypertext Transfer Protocol Secure)

Hypertext Transfer Protocol Secure is the secured interpretation of HTTP. The protocol provides secured communication between client and server using Transport Layer Security(TLS) or Secure Socket Layer(SSL) encryption. Using this protocol prevents a very common attack to perform that is called packet sniffing or formally named as man in the middle attack. Although, it won't prevent completely from packet sniffing but will encrypt the content so that the attacker will steal the cyphertext instead of original content. To continue with, SSL and TLS use asymmetric public key approach. For example, the data that is encrypted with a key can be decrypted only with another single key.

Figure 14 - HTTPS Certificate (Appendix J)



Certificate Information

This certificate is intended for the following purpose(s):
- Ensures the identity of a remote computer
- All issuance policies

Issued to: localhost

Issued by: localhost

Valid from 12/14/2019 to 12/13/2020

The certificate shown above is required for HTTPS implementation. This certificate contains the public key that user can use in order to securely send traffic through tiers.

Password hashing and salt

Hashing passwords with salts is a common password storage approach. To clarify, hash function output is a string that is hard to brute force, and a salt is a random generated string. Therefore, when storing a password, a salt is generated and the password is appended to it. Important to note, that salts and passwords are stored in the database separately. After that, a hash is generated with the appended salt to password. As a result, if attacker tries to steal a password, he will receive salted hashed password that is nearly impossible to brute force. This method can mitigate different types of attacks like rainbow table which is an attempt to discover the password from hash by using precomputed database hashes as well as dictionary attack which is using random strings to brute force the selected hash.

VPN – Virtual private network

When using software on a public network for example in a cafe restaurant, it's recommended to use VPN in order to make the connection between user and network secure. When using VPN, all user's traffic before reaching the internet goes to the VPN server, this being said, VPN works as a tunnel that hides all the data using encryption techniques. All in all, there are lots of VPN services on the market that are free and paid. Finally, paid VPNs used to provide faster internet speed than those that are free.

Implementation

Introduction

This chapter outlines how the software is implemented and also presents relevant code snippets that are relevant to context. Since the analysis part is finished and every single part of the system is designed, the implementation process used all the diagrams in order to continue further development. As software uses 3 Tier Architecture design pattern, implementation part will cover all layers of the system starting from Data layer.

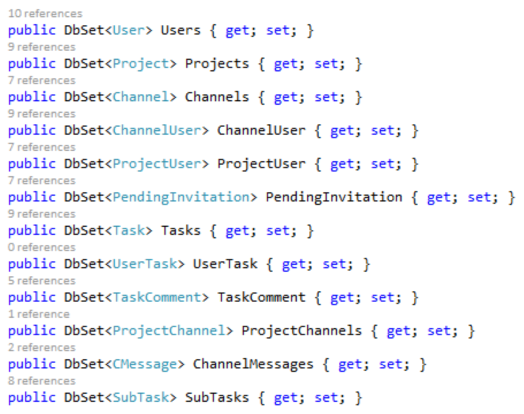
Data Tier

The main framework used for this tier is .NET Entity Framework Core. The data access is performed using a context object for querying data using MySQL database.

Context

Database context is performing query and saving operations to the database. In order to make the context know about what tables to create, the database sets are created as shown below in the *Figure 15*.

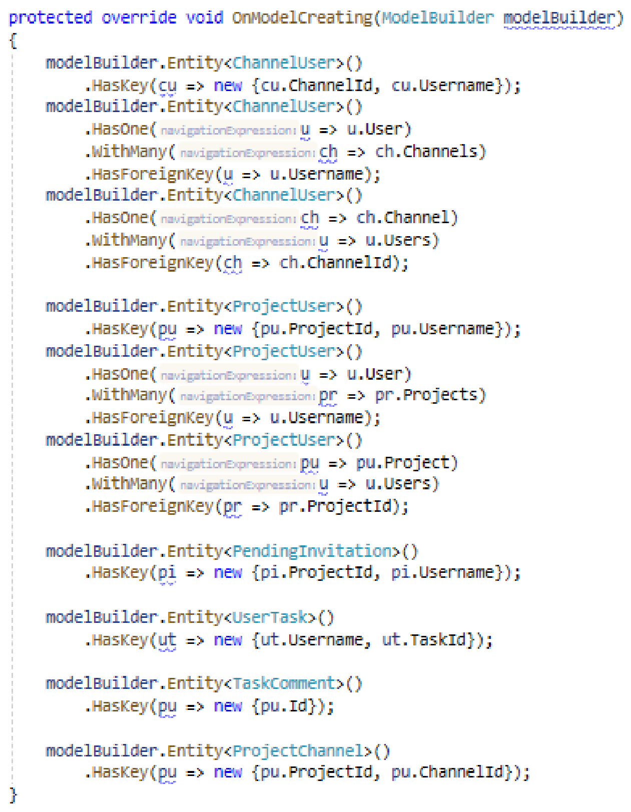
Figure 15 - Context DbSets (Appendix K)



```
10 references
public DbSet<User> Users { get; set; }
9 references
public DbSet<Project> Projects { get; set; }
7 references
public DbSet<Channel> Channels { get; set; }
9 references
public DbSet<ChannelUser> ChannelUser { get; set; }
7 references
public DbSet<ProjectUser> ProjectUser { get; set; }
7 references
public DbSet<PendingInvitation> PendingInvitation { get; set; }
9 references
public DbSet<Task> Tasks { get; set; }
0 references
public DbSet<UserTask> UserTask { get; set; }
5 references
public DbSet<TaskComment> TaskComment { get; set; }
1 reference
public DbSet<ProjectChannel> ProjectChannels { get; set; }
2 references
public DbSet<CMessage> ChannelMessages { get; set; }
8 references
public DbSet<SubTask> SubTasks { get; set; }
```

In order to develop consistent database relationships and configure domain classes was used FluentAPI. This approach is an alternative of using Data Annotations - *Figure 16*.

Figure 16 – FluentAPI (Appendix K)

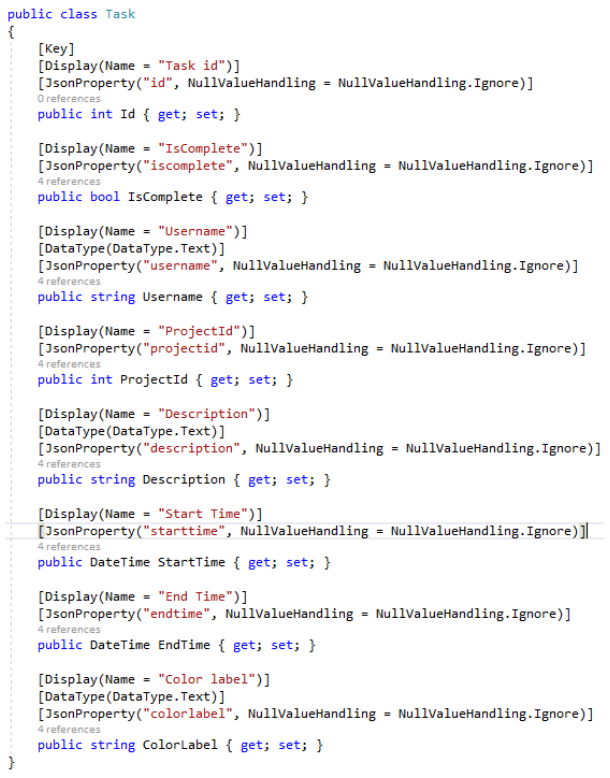


```
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    modelBuilder.Entity<ChannelUser>()
        .HasKey(cu => new {cu.ChannelId, cu.Username});
    modelBuilder.Entity<ChannelUser>()
        .HasOne( navigationExpression: u => u.User)
        .WithMany( navigationExpression: ch => ch.Channels)
        .HasForeignKey(u => u.Username);
    modelBuilder.Entity<ChannelUser>()
        .HasOne( navigationExpression: ch => ch.Channel)
        .WithMany( navigationExpression: u => u.Users)
        .HasForeignKey(ch => ch.ChannelId);

    modelBuilder.Entity<ProjectUser>()
        .HasKey(pu => new {pu.ProjectId, pu.Username});
    modelBuilder.Entity<ProjectUser>()
        .HasOne( navigationExpression: u => u.User)
        .WithMany( navigationExpression: pr => pr.Projects)
        .HasForeignKey(u => u.Username);
    modelBuilder.Entity<ProjectUser>()
        .HasOne( navigationExpression: pu => pu.Project)
        .WithMany( navigationExpression: u => u.Users)
        .HasForeignKey(pr => pr.ProjectId);

    modelBuilder.Entity<PendingInvitation>()
        .HasKey(pi => new {pi.ProjectId, pi.Username});

    modelBuilder.Entity<UserTask>()
        .HasKey(ut => new {ut.Username, ut.TaskId});

    modelBuilder.Entity<TaskComment>()
        .HasKey(pu => new {pu.Id});

    modelBuilder.Entity<ProjectChannel>()
        .HasKey(pu => new {pu.ProjectId, pu.ChannelId});
}
```

Entity Classes

Entity classes are the foundation of the database and based on them the migration process begins.

Figure 17 - Task Class (Appendix K)



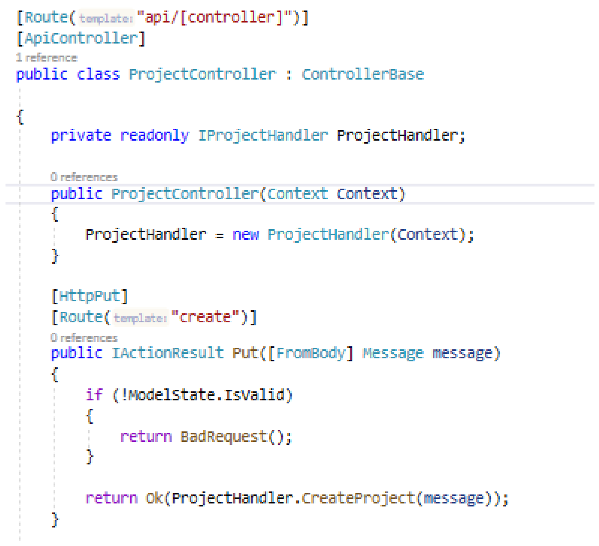
```
public class Task
{
    [Key]
    [Display(Name = "Task id")]
    [JsonProperty("id", NullValueHandling = NullValueHandling.Ignore)]
    0 references
    public int Id { get; set; }

    [Display(Name = "IsComplete")]
    [JsonProperty("iscomplete", NullValueHandling = NullValueHandling.Ignore)]
    4 references
    public bool IsComplete { get; set; }

    [Display(Name = "Username")]
    [DataType(DataType.Text)]
    [JsonProperty("username", NullValueHandling = NullValueHandling.Ignore)]
    4 references
    public string Username { get; set; }

    [Display(Name = "ProjectId")]
    [JsonProperty("projectid", NullValueHandling = NullValueHandling.Ignore)]
    4 references
    public int ProjectId { get; set; }

    [Display(Name = "Description")]
    [DataType(DataType.Text)]
    [JsonProperty("description", NullValueHandling = NullValueHandling.Ignore)]
    4 references
    public string Description { get; set; }

    [Display(Name = "Start Time")]
    [JsonProperty("starttime", NullValueHandling = NullValueHandling.Ignore)]
    4 references
    public DateTime StartTime { get; set; }

    [Display(Name = "End Time")]
    [JsonProperty("endtime", NullValueHandling = NullValueHandling.Ignore)]
    4 references
    public DateTime EndTime { get; set; }

    [Display(Name = "Color label")]
    [DataType(DataType.Text)]
    [JsonProperty("colorlabel", NullValueHandling = NullValueHandling.Ignore)]
    4 references
    public string ColorLabel { get; set; }
}
```

Figure 17 - shows the Task entity object with its data annotation configuration options.

Communication

The way data tier communicates with business logic is using REST approach. The reason to implement it this way, was that connection via REST API Calls provides a clear separation between business logic and data tiers and also it is independent in terms of platform type and languages.

Figure 18 - Data Project Controller (Appendix K)



```
[Route(template: "api/[controller]")]
[ApiController]
1 reference
public class ProjectController : ControllerBase

{
    private readonly IProjectHandler ProjectHandler;

    0 references
    public ProjectController(Context Context)
    {
        ProjectHandler = new ProjectHandler(Context);
    }

    [HttpPut]
    [Route(template: "create")]
    0 references
    public IActionResult Put([FromBody] Message message)
    {
        if (!ModelState.IsValid)
        {
            return BadRequest();
        }

        return Ok(ProjectHandler.CreateProject(message));
    }
```

Figure 18 - shows the Project API Controller implementation. To be clear, the controller holds a reference to the project handler object in order to transmit data further for processing. It's important to mention that the Context object is injected in the constructor using Dependency Injection. The put method will receive request under this route "api/Project/create" further getting the Message object from body. The result is sent back in a string JSON format.

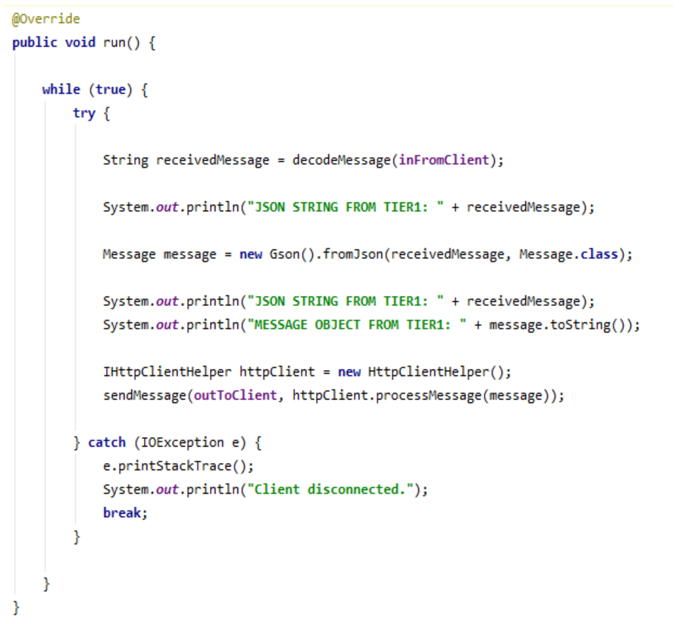
Business Logic Tier

The business logic is a Java program that serves as a SocketServer in relation to the Presentation tier. The data is sent through a socket and is transmitted for further processing.

SocketServer

The socket that resides on the server side is called a socket server. Its purpose is to receive data from the presentation tier and send it to other objects for processing objectives. The socket is running on localhost and receives information via input stream object.

Figure 19 - ServerSocket RunMethod (Appendix K)

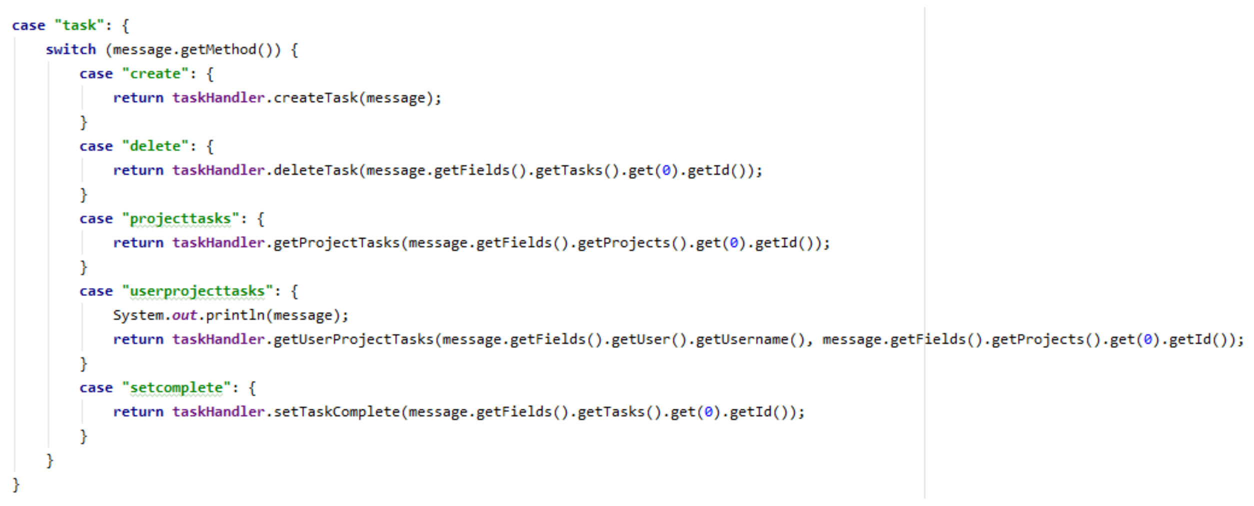


```java
@Override
public void run() {

    while (true) {
        try {

            String receivedMessage = decodeMessage(inFromClient);

            System.out.println("JSON STRING FROM TIER1: " + receivedMessage);

            Message message = new Gson().fromJson(receivedMessage, Message.class);

            System.out.println("JSON STRING FROM TIER1: " + receivedMessage);
            System.out.println("MESSAGE OBJECT FROM TIER1: " + message.toString());

            IHttpClientHelper httpClient = new HttpClientHelper();
            sendMessage(outToClient, httpClient.processMessage(message));

        } catch (IOException e) {
            e.printStackTrace();
            System.out.println("Client disconnected.");
            break;
        }

    }
}
```

Figure 19 - shows the top view process of receiving and sending the Message object next to http client object. The bits are decoded in the decode message method and stored into a string. Next, the string is converted into a Message DTO and sent further.

DTO Processing

Figure 20 - Message Processing Logic (Appendix K)

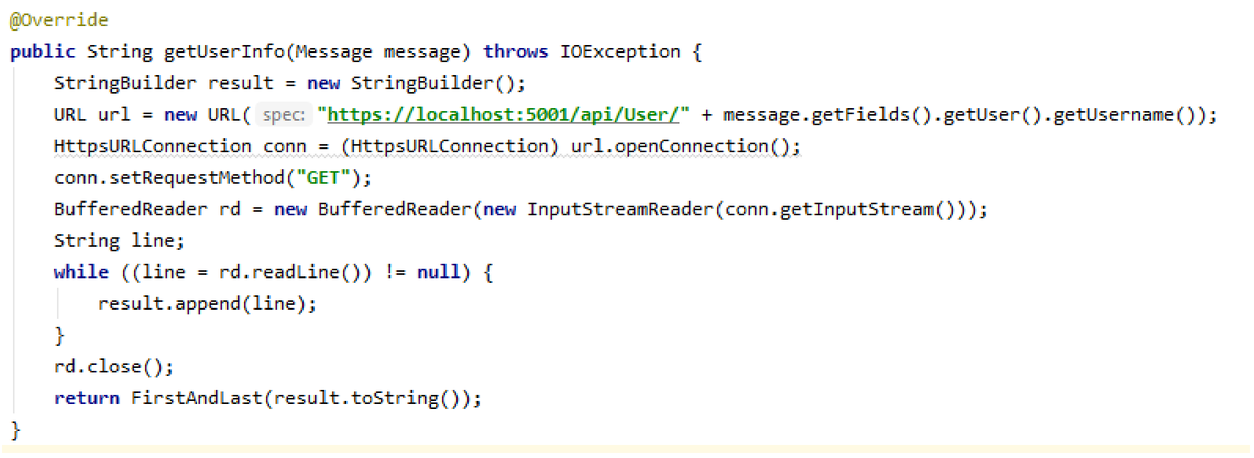


```
case "task": {
    switch (message.getMethod()) {
        case "create": {
            return taskHandler.createTask(message);
        }
        case "delete": {
            return taskHandler.deleteTask(message.getFields().getTasks().get(0).getId());
        }
        case "projecttasks": {
            return taskHandler.getProjectTasks(message.getFields().getProjects().get(0).getId());
        }
        case "userprojecttasks": {
            System.out.println(message);
            return taskHandler.getUserProjectTasks(message.getFields().getUser().getUsername(), message.getFields().getProjects().get(0).getId());
        }
        case "setcomplete": {
            return taskHandler.setTaskComplete(message.getFields().getTasks().get(0).getId());
        }
    }
}
```

Figure 20 – shows Message processing logic and how it is sent to different handlers. This case shows only message of resource type task.

HTTPS Methods

Figure 21 – Operation Performing (Appendix K)



```
@Override
public String getUserInfo(Message message) throws IOException {
    StringBuilder result = new StringBuilder();
    URL url = new URL( spec: "https://localhost:5001/api/User/" + message.getFields().getUser().getUsername());
    HttpsURLConnection conn = (HttpsURLConnection) url.openConnection();
    conn.setRequestMethod("GET");
    BufferedReader rd = new BufferedReader(new InputStreamReader(conn.getInputStream()));
    String line;
    while ((line = rd.readLine()) != null) {
        result.append(line);
    }
    rd.close();
    return FirstAndLast(result.toString());
}
```

Figure 21 - shows how a GET method is performed and also a part of communication between data and business logic tier.

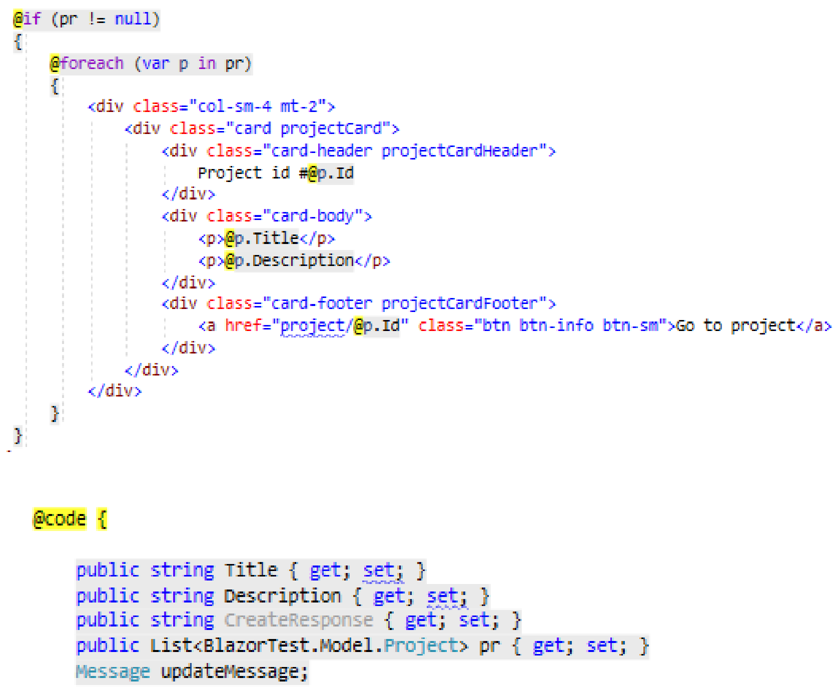
Presentation

The first, presentation tier is the front-end layer that includes the UI. The decision was made to use .NET Core 2 along with an open source framework named Blazor that function with C#, HTML and CSS. Blazor takes control over user interface using Razor Pages (.razor file) that consists of two mixed languages C# and HTML. Also, Blazor uses component-based approach that is nested able, reusable and shareable. Finnaly, for building a responsive web UI was used Bootstrap 4 and for styling - CSS.

Razor pages

Razor pages, in other words are Blazor components. As explained above, components are used for UI manipulation and it is a mix of C# and HTML languages. *Figure 22* - shows how two languages communicate in the same file as well as how project UI is displayed. In a nutshell C# is responsible mostly for displaying information itself whereas html structures it.

Figure 22 – Razor Page content (Appendix K)

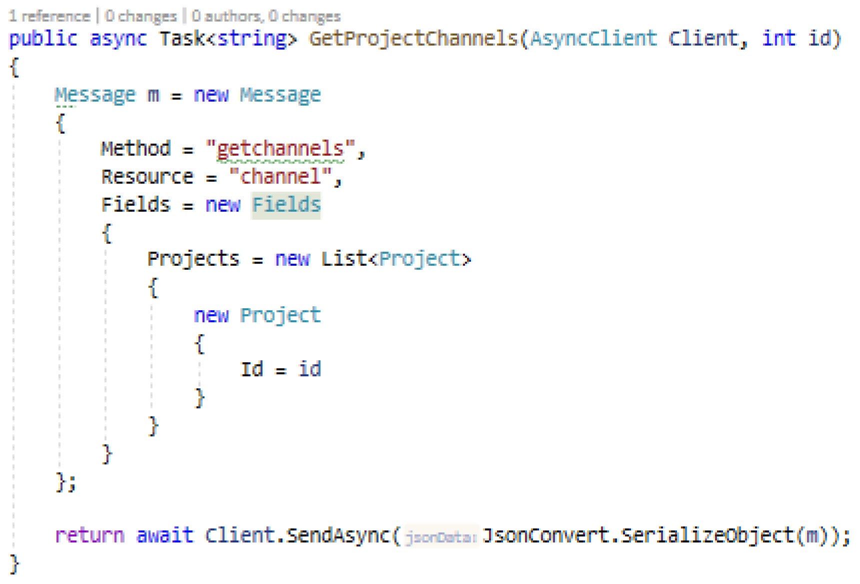


```
@if (pr != null)
{
    @foreach (var p in pr)
    {
        <div class="col-sm-4 mt-2">
            <div class="card projectCard">
                <div class="card-header projectCardHeader">
                    Project id #@p.Id
                </div>
                <div class="card-body">
                    <p>@p.Title</p>
                    <p>@p.Description</p>
                </div>
                <div class="card-footer projectCardFooter">
                    <a href="project/@p.Id" class="btn btn-info btn-sm">Go to project</a>
                </div>
            </div>
        </div>
    }
}


@code {

    public string Title { get; set; }
    public string Description { get; set; }
    public string CreateResponse { get; set; }
    public List<BlazorTest.Model.Project> pr { get; set; }
    Message updateMessage;
```

Controllers

Controllers are classes that have the responsibility to react on users' actions and perform some activity in order to change the UI or send some information to the second tier. Also, controllers are responsible for creating DTO messages in a way that business logic will be able to process.

Figure 23 - Presentation Controller Method (Appendix K)

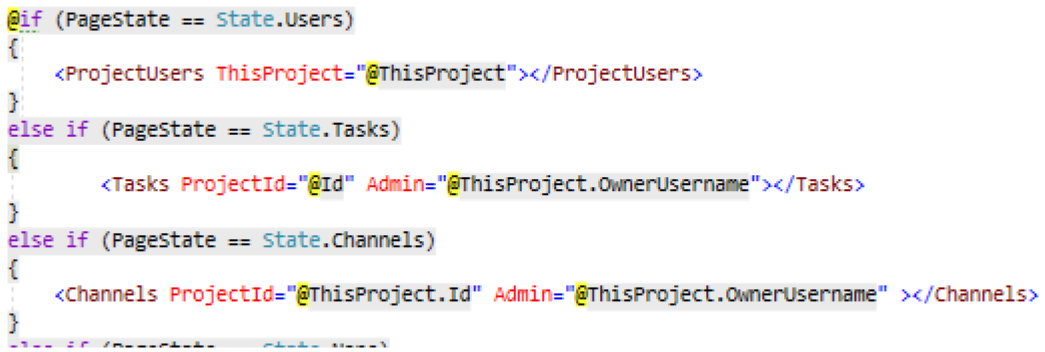


```
1 reference | 0 changes | 0 authors, 0 changes
public async Task<string> GetProjectChannels(AsyncClient Client, int id)
{
    Message m = new Message
    {
        Method = "getchannels",
        Resource = "channel",
        Fields = new Fields
        {
            Projects = new List<Project>
            {
                new Project
                {
                    Id = id
                }
            }
        }
    };

    return await Client.SendAsync(jsonData: JsonConvert.SerializeObject(m));
}
```

Figure 23 - shows how DTO is created and send through sockets, the message top information is saying that user wants to get all channel of a project, therefore the method is get channels and the resource is channel – note that project id is also specified.

Components

Figure 24 - Component Manipulation (Appendix K)



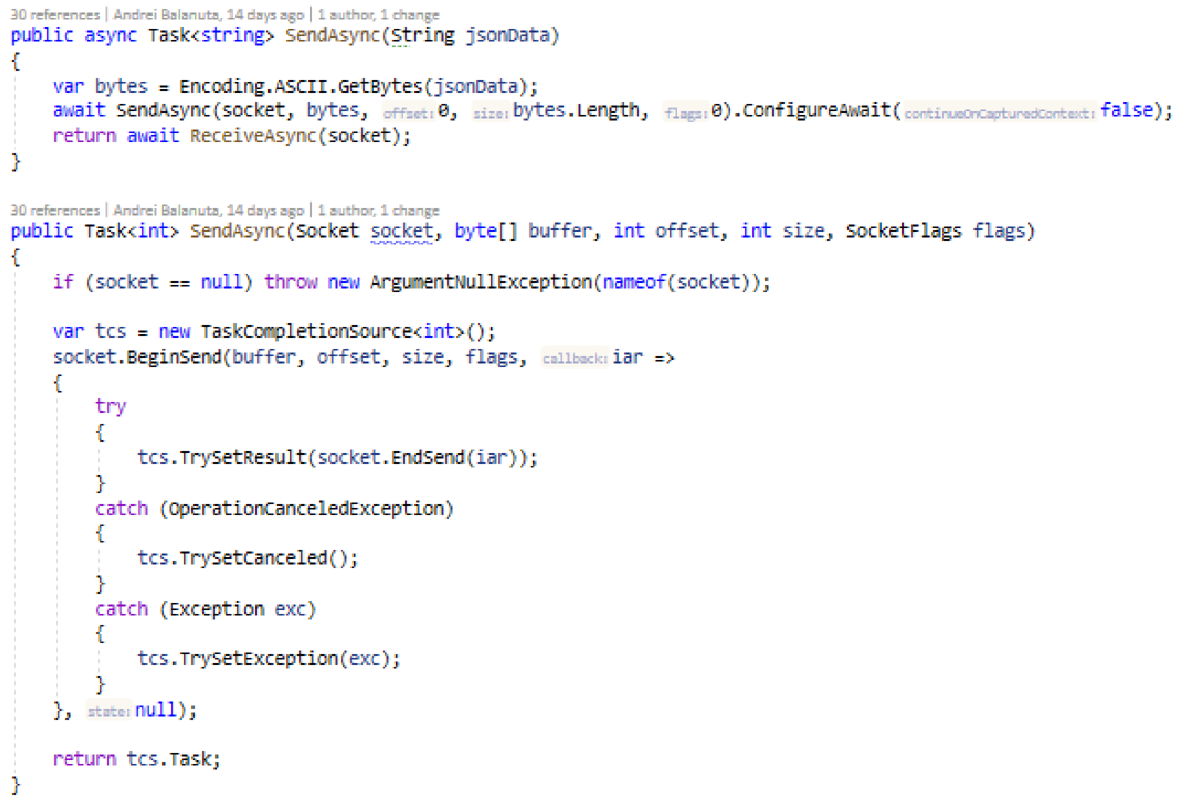
```
@if (PageState == State.Users)
{
    <ProjectUsers ThisProject="@ThisProject"></ProjectUsers>
}
else if (PageState == State.Tasks)
{
        <Tasks ProjectId="@Id" Admin="@ThisProject.OwnerUsername"></Tasks>
}
else if (PageState == State.Channels)
{
    <Channels ProjectId="@ThisProject.Id" Admin="@ThisProject.OwnerUsername" ></Channels>
}
```

Figure 24 - shows power of Blazor that is nesting components. This approach is making development approach easier in terms of navigation and flexibility.

Asynchronous Sockets

In order to make all threads independent, not only razor pages and controllers perform async methods but also sockets do.

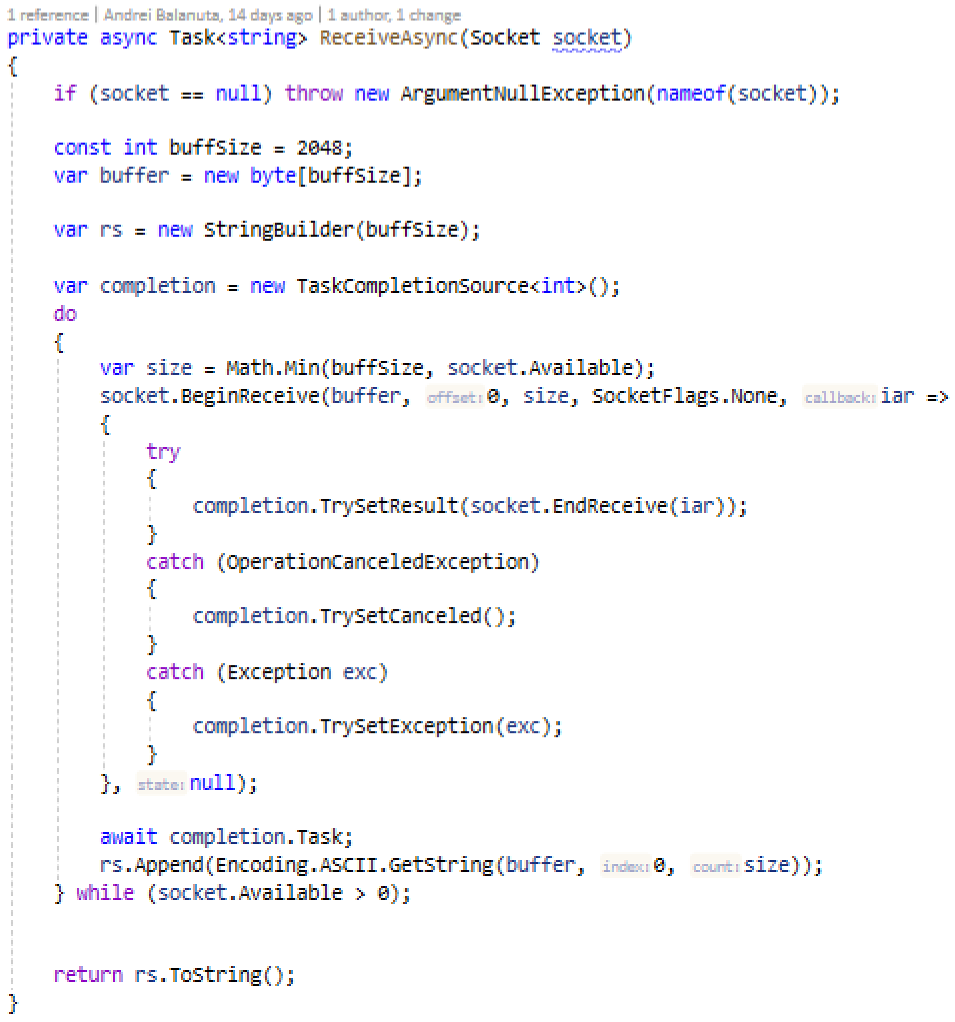
Figure 25 - Socket Send Async (Appendix K)



```
30 references | Andrei Balanuta, 14 days ago | 1 author, 1 change
public async Task<string> SendAsync(String jsonData)
{
    var bytes = Encoding.ASCII.GetBytes(jsonData);
    await SendAsync(socket, bytes, offset: 0, size: bytes.Length, flags: 0).ConfigureAwait(continueOnCapturedContext: false);
    return await ReceiveAsync(socket);
}

30 references | Andrei Balanuta, 14 days ago | 1 author, 1 change
public Task<int> SendAsync(Socket socket, byte[] buffer, int offset, int size, SocketFlags flags)
{
    if (socket == null) throw new ArgumentNullException(nameof(socket));

    var tcs = new TaskCompletionSource<int>();
    socket.BeginSend(buffer, offset, size, flags, callback: iar =>
    {
        try
        {
            tcs.TrySetResult(socket.EndSend(iar));
        }
        catch (OperationCanceledException)
        {
            tcs.TrySetCanceled();
        }
        catch (Exception exc)
        {
            tcs.TrySetException(exc);
        }
    }, state: null);

    return tcs.Task;
}
```

Figure 25 - shows how a socket sends a json formatted string asynchronously. To be clear, there are two methods send async with different parameters and return types. The first one is responsible for converting string data into bytes and calling the second send async method by setting the socket, bytes and some more configuration as parameters. The second send async method is responsible for begin sending and ending a pending sending. The begin/end send/receive pattern is a common part of asynchronous socket programming. Its important to mention that the same pattern follows the receive async method that is shown in the following *Figure 26*.

Figure 26 - Socket Receive Async (Appendix K)

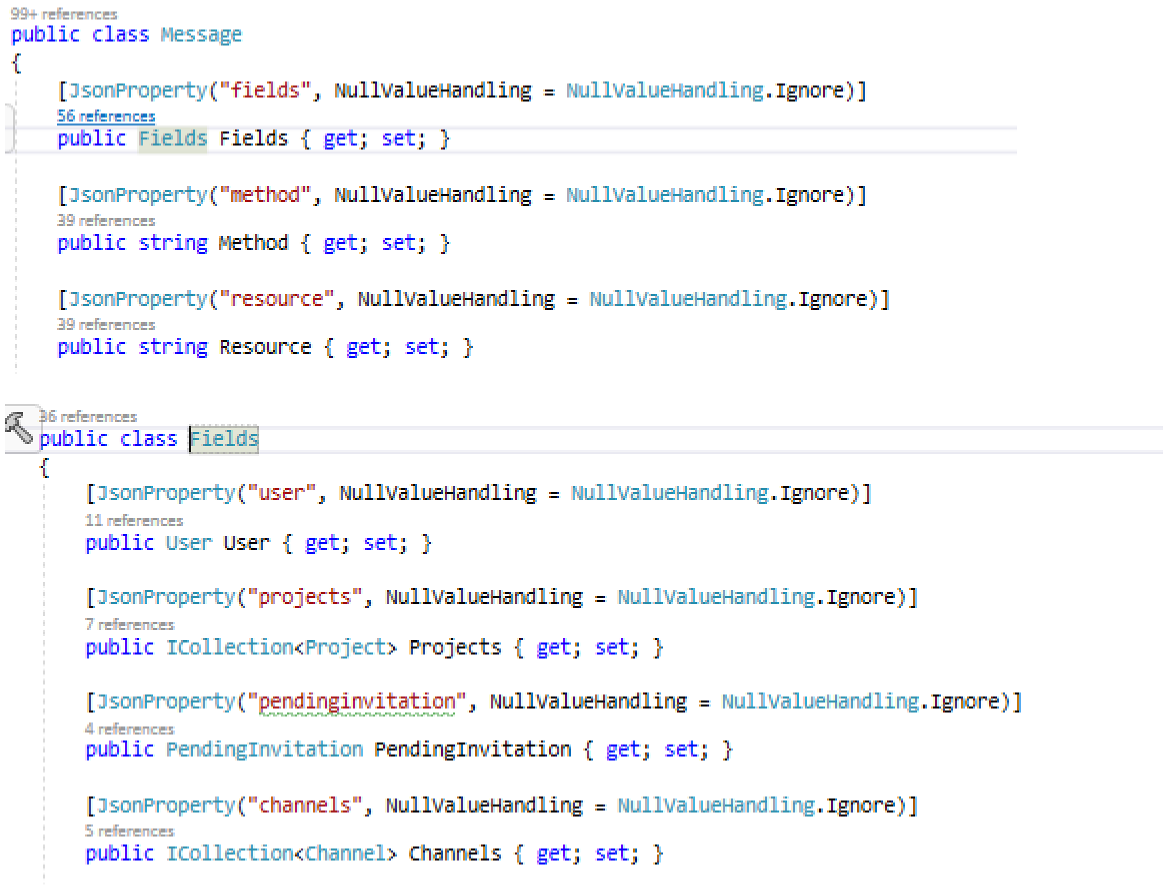


```
1 reference | Andrei Balanuta, 14 days ago | 1 author, 1 change
private async Task<string> ReceiveAsync(Socket socket)
{
    if (socket == null) throw new ArgumentNullException(nameof(socket));

    const int buffSize = 2048;
    var buffer = new byte[buffSize];

    var rs = new StringBuilder(buffSize);

    var completion = new TaskCompletionSource<int>();
    do
    {
        var size = Math.Min(buffSize, socket.Available);
        socket.BeginReceive(buffer, offset: 0, size, SocketFlags.None, callback: iar =>
        {
            try
            {
                completion.TrySetResult(socket.EndReceive(iar));
            }
            catch (OperationCanceledException)
            {
                completion.TrySetCanceled();
            }
            catch (Exception exc)
            {
                completion.TrySetException(exc);
            }
        }, state: null);

        await completion.Task;
        rs.Append(Encoding.ASCII.GetString(buffer, index: 0, count: size));
    } while (socket.Available > 0);


    return rs.ToString();
}
```

DTO and transfer process

Data transfer objects are made using specific fields in order to make each tier understand the request and to process it in the right way. Namely, the DTO is a Message object that holds specific variables as method, resource and fields. Through message sending process, message fields variable bears all the information that is required *Figure 27*.

Figure 27 - Message Object (Appendix K)

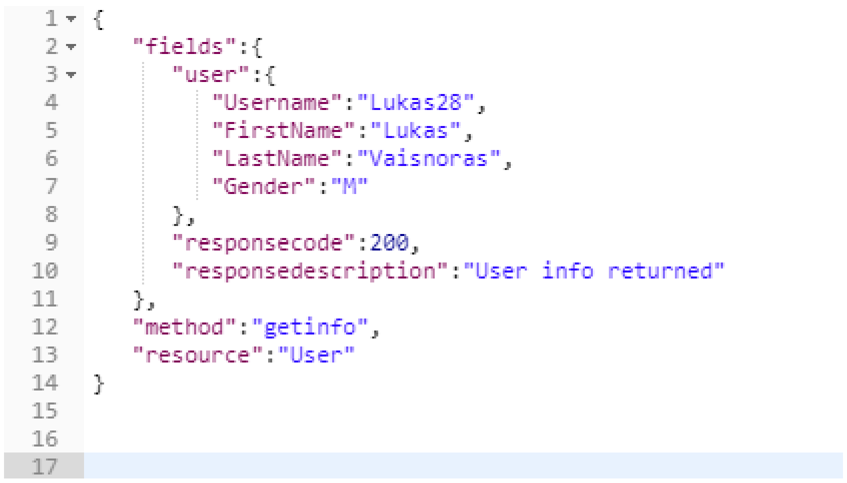


```
99+ references
public class Message
{
    [JsonProperty("fields", NullValueHandling = NullValueHandling.Ignore)]
    56 references
    public Fields Fields { get; set; }

    [JsonProperty("method", NullValueHandling = NullValueHandling.Ignore)]
    39 references
    public string Method { get; set; }

    [JsonProperty("resource", NullValueHandling = NullValueHandling.Ignore)]
    39 references
    public string Resource { get; set; }

36 references
public class Fields
{
    [JsonProperty("user", NullValueHandling = NullValueHandling.Ignore)]
    11 references
    public User User { get; set; }

    [JsonProperty("projects", NullValueHandling = NullValueHandling.Ignore)]
    7 references
    public ICollection<Project> Projects { get; set; }

    [JsonProperty("pendinginvitation", NullValueHandling = NullValueHandling.Ignore)]
    4 references
    public PendingInvitation PendingInvitation { get; set; }

    [JsonProperty("channels", NullValueHandling = NullValueHandling.Ignore)]
    5 references
    public ICollection<Channel> Channels { get; set; }
```

Note that Fields class presented in the figure has one third of the actual object. The full representation can be found in the class diagram.

In order for all tiers to understand it, the JsonProperty naming data annotation functionality was used. In addition, it also filters the object by ignoring null variables. All in all, the message object is serialized with JsonNewsoft in C# and with Gson in java.

Figure 28 - Json Message Json Serialization (Appendix K)



```
{
    "fields":{
      "user":{
        "Username":"Lukas28",
        "FirstName":"Lukas",
        "LastName":"Vaisnoras",
        "Gender":"M"
      },
      "responsecode":200,
      "responsedescription":"User info returned"
    },
    "method":"getinfo",
    "resource":"User"
}
```

Figure 28 - shows how Message object is serialized in a json format.

Testing

Black Box Testing

For testing purposes, the black box methodology will be used in order to test the software from user's perspective. This approach follows the principle that the source code is unknown and the application is tested from the browser. The following test is using project's user stories.

Table 2 - White Box Testing

Id	User Story	Passed/Not Passed
1	As a user I want my account to be created on the base of my personal information (First name, last name, date of birth, profile picture, username, gender, password).	Passed
2	As a user I want to be able to log in/log out of the system	Passed
3	As a user I want to be able to edit my personal information	Passed
4	As a user I want to be able to create a project and become it's administrator.	Passed
5	As an administrator I want to be able to invite people from my project by username.	Passed
6	As an administrator I want to be able to kick out members from my project.	Passed
7	As an administrator I want to be able to create/delete tasks.	Passed
8	As an administrator I want to be able to create/delete subtasks.	Passed
9	As an administrator I want to be able to set task priority by color labels.	Passed
10	As an administrator I want to assign project members to tasks.	Passed
11	As an administrator I want to set task's status as compete.	Passed
12	As a user I want to be able to leave a comment for a task.	Passed
13	As an administrator I want to create a channel for department's communication.	Passed
14	As an administrator I want to invite members to channel.	Passed
15	As an administrator I want to be able to kick out members from channel.	Passed
16	As a user I want to be able to send text messages, images, code snippets, docx, doc, pdf, xls, csv etc.	Partial

17	As a user I want to be able to chat privately with another user.	Not Passed
18	As a user I want to be able to see current progress of a task.	Passed
19	As a user I want to be notified about future tasks deadline, messages, project invitations.	Not Passed
20	As a user I want to have access to project tasks and message history.	Passed

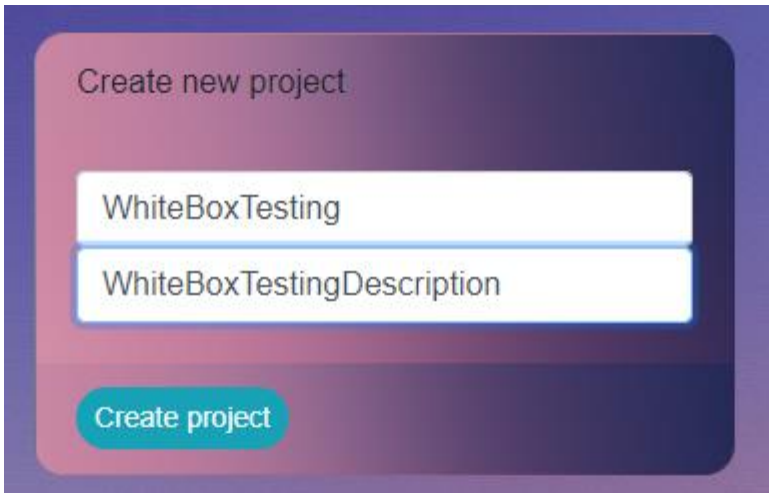
As a result of black box testing, most of the user stories are implemented and work without bugs.

White Box Testing

The white box testing is from developers perspective where the source code is available and testing happens from inside of the software. To begin the test, the feature create project will be used in order to view the full path through all three tiers and to see what actions are performed.

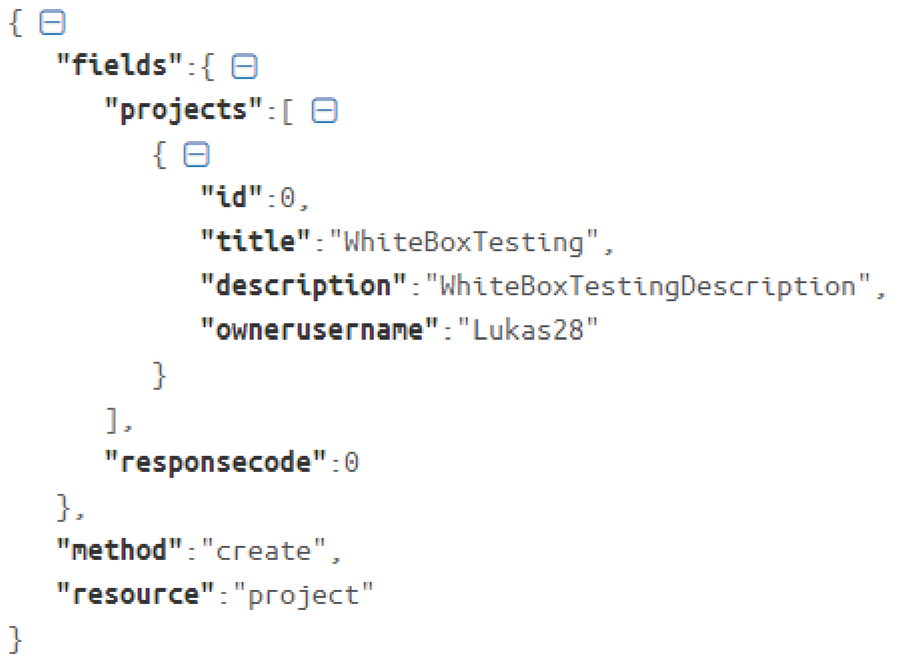
First, in order to create a project, it's required to jump into the projects page.

Figure 29 - Test1 (Appendix K)



Clicking the create project will send a serialized message in a JSON format to the business logic tier *Figure 29*.

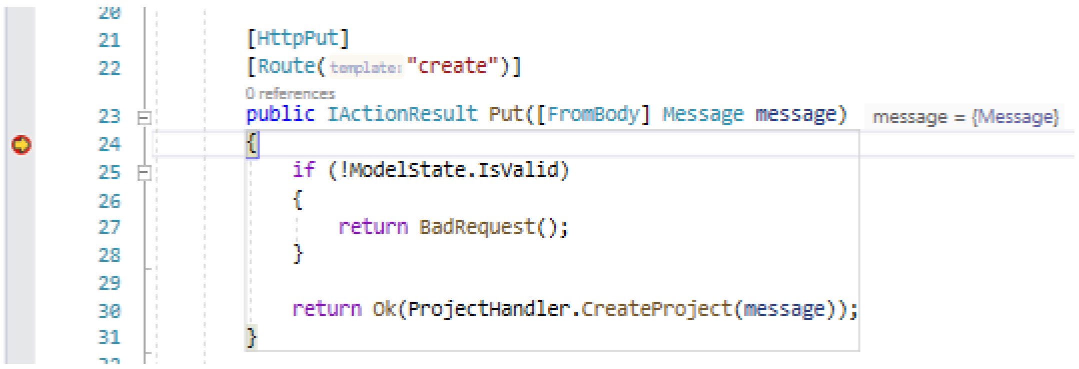
Figure 30 - Test2 (Appendix K)



```
{
    "fields":{
        "projects":[
            {
                "id":0,
                "title":"WhiteBoxTesting",
                "description":"WhiteBoxTestingDescription",
                "ownerusername":"Lukas28"
            }
        ],
        "responsecode":0
    },
    "method":"create",
    "resource":"project"
}
```

The contents of message received can be seen in the *Figure 30*, this message is processed and therefore is sent to the Data tier.

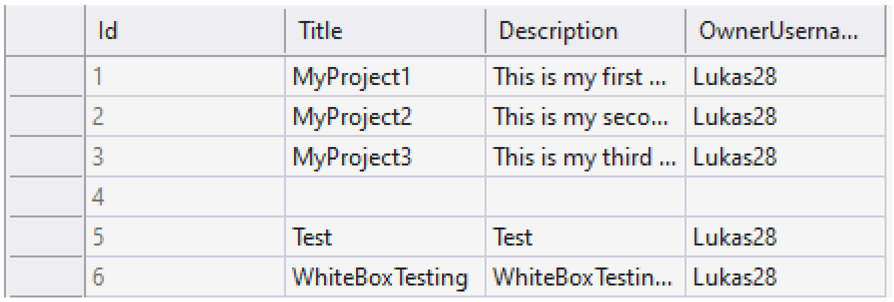
Figure 31 - Test3 (Appendix K)



```
[HttpPut]
[Route(template: "create")]
0 references
public IActionResult Put([FromBody] Message message)    message = {Message}
{
    if (!ModelState.IsValid)
    {
        return BadRequest();
    }

    return Ok(ProjectHandler.CreateProject(message));
}
```

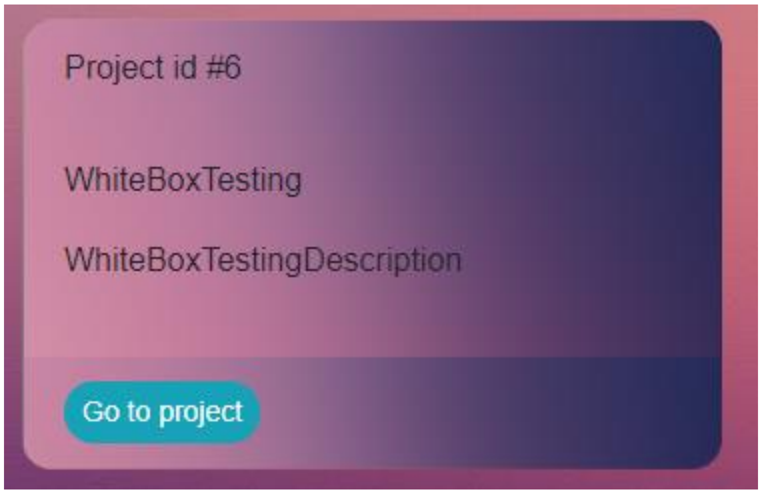
As follows in *Figure 31*, the rest api got the request and the break point was hit, therefore the message object is sent further for processing. Since the method is create, the project must be seen in the database.

Figure 32 - Test4 (Appendix K)



	Id	Title	Description	OwnerUserna...
	1	MyProject1	This is my first ...	Lukas28
	2	MyProject2	This is my seco...	Lukas28
	3	MyProject3	This is my third ...	Lukas28
	4			
	5	Test	Test	Lukas28
	6	WhiteBoxTesting	WhiteBoxTestin...	Lukas28

Figure 33 - Test5 (Appendix K)



Project id #6

WhiteBoxTesting

WhiteBoxTestingDescription

As a result of *Figure 32, 33*, the project is stored in the database and the user became the admin of it. Also, the project is displayed on users window screen.

Results and discussion

The purpose of this project was to create a project management software that follows the 3-tier architecture design. Additionally, the requirement was to use specific technologies that are mentioned in the project report. Therefore, the team managed to fulfill most of the requirements as it can be seen in the project report’s test section. The software works without any system conflicts and is ready for future development and updates. Without a doubt, the software is not perfect and requires more analysis, design and updates but generally speaking it’s usable and ready for deployment. Moreover, this software is stable in terms of the security aspect. In this project were used different techniques in order to hide information from unauthorized users and to prevent many other threats. Therefore, software runs securely and users can perform many actions without worrying about different threats that can damage or affect their sensitive data. Regarding test section, the groups point of view is that the best test is the user’s feedback. Once “Galaxis” Software Engineering team is going to consume this application, the group is ready for new challenges and critical feedbacks. All in all, the result is a software that went through a big process with many steps performed in order to achieve success.

Conclusions

To begin with, software development process consists of many steps and each step performed in this project gave a real shape to it. Also, taking in consideration that this is a student project, it cannot be compared to a real life one, therefore it's developed for teaching purposes. Moreover, the path from a single notion to a stable software is considerably long comparing other projects experience. Additionally, this project is a representation of what the group has learnt during the third semester. Thus, provided technologies were used in order to create a 3-tier architecture software.

To continue with, each step in the project is a small overview of the application. Starting with requirements it can be seen how they introduce every single feature as an atomic value. After setting the requirements, the analysis process begun and designing diagrams was a procedure of understanding how the user will act with the application in different situations. Overall, the analysis part went great because of the experience from previous projects.

Furthermore, after having all analysis diagrams, the design process begun to make the project proposal more structured and also gave a detailed understanding of how software should be implemented for developers. Taking in consideration that before implementation is required to spend a lot of time analyzing and designing the software organization – it is highly important to have all diagrams and resources before jumping into code.

Moreover, implementation part started with a simple skeleton where all three tiers could only send a single piece of string. From this point the software started to gain weight and more sense. Throughout the implementation process all diagrams were updated and redesigned due to agile development methodology. Therefore, analysis and design didn't stop when implementation phase has started. Additionally, during the implementation phase the group managed to use different design patterns and programming mechanisms that improved software optimization and processing speed. Also, at the end of the implementation process the security policy were adopted.

Additionally, testing process gave an understanding how different parts work and what needs to be improved for the application in order to make it fully work. In contrast, different testing methodologies were followed with the intent of grasping software behavior in different circumstances. Also, placing it in different tough situations made to improve its information conduction.

All in all, the group team feels joyful of the project's outcome. Implementing most of the requirements is a great achievement. Although, the application requires future updates and optimization – it's still a functional software and ready for future use. It is highly important that everyone using the software is directed from the user guide for avoiding different disputes.

Project Future

First, if talking about software, even if it is perfect – there will be always a reason for a small update. Software optimization is never ending in it's lifeline. Slit Project management tool is an instrument that Galaxis project manager can use in order to show his team members the correct path for achieving success. Therefore, coming with bad feedbacks might generate great ideas for software updates. As the time passes, it's usual that the team will evolve and will need more tools for software development objectives. Thus, the project group is up for new challenges.

Moreover, the security facet also can be updated with new techniques and features in order to prevent attackers from stealing or viewing information illegally. Thinking rational, 3 tier architecture is a big system that can break unexpectable and nothing lives forever. Thus, bugs can appear not because of analysis or design issues but also from unanticipated events.

Second, the backlog items that were not implemented in the project development process would be nice to implement in order to fulfill customer's requirements completely. The group didn't manage to store images and different file extensions that are very important for group's workflow. Also, chatting privately with another users is highly important in terms of interconnection between project members and also for admins. Therefore, private chatting issue can cause different conflicts between group members and efficiency decrease.

Third, a good idea would be implementing additional graphs in order to improve the efficiency of grasping and tracking every single step of the project development process. Also, implementing graphs can go further to forecasting different events. For example, the administrator would be able to see his economy status in terms of money and there are lots of forecasting algorithms that work on the base of data history. Although, it won't be a concrete forecast but if the software will be used in another field for example value chain team management, it would be extremely useful for them.

Third, another great idea would be to implement SSL socket connection between first and second tier in order to prevent man in the middle attacks. Even if the software has it's own level of security, there are always some gaps that attackers can use in order to steal users sensitive information.

In conclusion, the group knows that the software can have a big future and these are not the only ideas that should be implemented. Therefore, implementing these features will be a big breakthrough for the software in terms of more functionality and users consumption experience. All in all, there is a big hope that not only Galaxis will use this software but also many other teams around the world.

References

R.Brooks n.d. *Introduction (RIB) (1)*.

Authentication, M., Signatures, D., Applications, O. and Attacks, B., n.d. Chapter C ryptographic H ash. pp.313–328.

Chapter, A., Diffie, W. and Hellman, M., 1978. Public-Key Cryptosystems. pp.256–278.

Damg, I., 2007. An Introduction to some Basic Concepts in IT Security and Cryptography. pp.1–11.

Guidelines, V.I.A.E., 2018a. APPENDIX 3 Project Report. (August).

Guidelines, V.I.A.E., 2018b. SEMESTER & BACHELOR PROJECTS. (August).

Papernot, N., n.d. *Threat model*. *The limitations of Deep Learning in Adversarial Settings*, .

Rasmussen, J.K. and Vaisnoras, L., 2019. Project Description Support Document - VIA Software Technology Engineering. (September), pp.1–9.

(Authentication et al., n.d.; Rasmussen and Vaisnoras, 2019; Chapter, Diffie and Hellman, 1978;

Introduction (RIB) (1), n.d.; Damg, 2007; Guidelines, 2018b; Papernot, n.d.; Guidelines, 2018a)

Vision, C., 2009. astah * Basic Operation Guide. *Screen*.

Darril, 2015. Rainbow Table Attacks. [online] (September), p.1.
Available at: <https://blogs.getcertifiedgetahead.com/rainbow-table-attacks/>.
[Accessed 10 December 2019]

Anon n.d. Three-Tier Architecture. [online] p.1.
Available at: <https://www.techopedia.com/definition/24649/three-tier-architecture>.
[Accessed 17 December 2019]

Microsoft, 2019. .NET documentation. [online]
Available at: <https://docs.microsoft.com/en-us/dotnet/>.
[Accessed 16 December 2019]

Rouse, M., n.d. confidentiality, integrity, and availability (CIA triad). *Threat Management*. [online]
Available at: <https://whatis.techtarget.com/definition/Confidentiality-integrity-and-availability-CIA>.
[Accessed 16 December 2019]

CASSETTO, O., n.d. Threat Hunting: Tips and Tools. [online]
Available at: <https://www.exabeam.com/security-operations-center/threat-hunting/>.
[Accessed 16 December 2019]

Anon n.d. Using Threat Modeling in Cybersecurity to Hunt and Remediate. *WEB APPLICATION SECURITY*. [online]
Available at: <https://lab.wallarm.com/using-threat-modeling-in-cybersecurity-to-hunt-and-remediate-2fd41b3abd2e/>. [Accessed 10 December 2019]

University, A.R., 2008. Harvard Style of Referencing. *Anglia Ruskin University*, (July), pp.1–32.

Appendices

VIA University College

SLIT

Heterogeneous system
Process Report – Project management tool
Third semester – Software Engineering

Supervisors	Students
Jakob Knop Rasmussen	Andrei Balanuta (281045)
Jan Otto Munch Pedersen	Lukas Vaisnoras (280107)

Characters with spaces

21,094

Contents

Introduction

At the beginning of the semester our group was formed and signing the contract each of us took the responsibility to work on the semester project together. Setting the rules and limits together we knew that not respecting them would damage group relationship. After that, supervisors presented to us the third semester overview and gave us recommendations, semester plan and requirements. . In addition, teachers showed us the technologies as well as the way to combine them together in order to have a general idea on how the project software should be developed.

At the end of October, we already had the project proposal accepted, project description done, architecture diagram and the skeleton. At this point, all project members understood how technologies work together and the development went easier as the time passed. Having to say that, the system architecture we designed at the beginning lasted until end of the project. Using communication tools and different frameworks helped us to manage information and keep in touch when needed.

Furthermore, we had to choose a task management framework and those were waterfall, scrum and Kanban. As a result, we chose scrum because it helps to save time, encourages team members and it's easy to use since we already had some experience with it from previous semesters and we already knew how to use it wisely. Scrum was our main framework of keeping track of everything. It helped us to stay focused and gave us more motivation by setting deadlines and kept our group connected. In other words, scrum was the perfect framework for the group, not using it would make us a lot less productive.

Group Description

Andrei Balanuta and Lukas Vaisnoras.

Having different backgrounds and no experience working with each other before, we could not have any expectations at the beginning of the project. Andrei is from Moldova and before being enrolled at VIA he finished High School in his home country. Same as Andrei, Lukas also finished High School but in Lithuania. The first two semesters we both didn't change groups until the third one. This took us out of the comfort zone and made us explore different points of view (Group Contract can be found in Appendix A).

Group contract

Group contract was developed based on our principles and common requirements. We felt that it is the best way since we did not have group problems and big disagreements in the last semester using same rules. It covers main problems that can be caused during semester project. While working on the project with this contract we had no conflicts, and everyone was following rules which leaded us to great team work. In other words, the contract gave us some limits and responsibilities that made us more organized. (Group contract can be found in the Appendix A).

Cultural Background

Analyzing the Hofstede dimensions, we can see that there are big differences between Lithuanian and Romanian cultural backgrounds. Major divergences can be seen in power distance and masculinity, in both dimensions Romania is twice in the lead. However, we didn't feel that during the project process. In other words, when we got known each other more, all of us realized that we cannon notice differences in cultural background (Hofstede diagram is attached in the Appendix C).

Project Experience

Skills we got from the previous semesters helped to achieve this semester's goals. As always, experience in this semester grew exponentially as the time passed. We learnt to be patient and follow the rules that were set at the beginning of our journey. Moreover, the experience of this project has helped us to create an overview of the insight of a typical software we would develop in a real-life company using scrum and modern technologies. The deadlines we had to achieve got us more organized and taught us how to solve problems logically.

Project initiation

During the initiation stage, members went through the group formation, signing the group contract and presenting the project proposal. Generating project proposals was one of the hardest and brainstorming part of the project process because we didn't know what idea will fit perfectly with the technologies we used so far. Each of us came up with 2 ideas with the smallest possible quantity of requirements, and after a group meeting, we had to select what idea to present. The idea that we proposed was a Project management software and the biggest reason was that the requirements did not go over our expectations. The idea was something like Slack, that's why we named our software Slit. When we got the feedback from our supervisors and got the proposal accepted, we started to expand the requirements in a way that we would fit time deadline, and supervisor's expectations.

Project description

Firstly, when our project proposal got accepted, we started to write the Project Description, it was the step of creating the backstory how the problem has arisen and the big picture. Stating the definition of purpose helped us to determine the general final solution of the problem – developing collaborative software for a group of people that face project management issues.

Secondly, the problem statement led us to find more subproblems that helped us to develop project requirements in the future and to create the product backlog for scrum. Moreover, we also chose the methodology agile along with unified process and some tools - Jira Software for Scrum development, MeisterTask and Slack for collaboration and efficient information management process. Furthermore, we also have set the milestones by creating a time schedule by counting the working hours and dividing them into forecasted sprints and unified process phases. Although we knew our risks, project description phase made us write them down and evaluate potential risk impacts in order to prevent future process devastation.

As a result, project description was used as a guideline for further development. Defining all problems and future events was essential because it helped to target more on research which is fundamental for deeper understanding of the subject (Project description can be found in Appendix B).

Project Execution

To begin with, this part was the most interesting and at the same time the most stressful stage or software development as it represents gathering, examination and processing data. Project execution consists of group meetings, supervisor meetings, implementing software requirements using different frameworks and tools. This chapter will be described from SCRUM perspective.

Methods

Agile Scrum development methodology was chosen because it relies on a self-organizing and cross-functional team. Therefore, decisions were made using group meetings and group members were equal throughout the project development. This methodology led us not to have conflicts and to solve problems in a harmonious way. Using JIRA Software helped us a lot with the first two sprints, but after all we realized that it’s too complex for our and we decided to switch to excel to manage scrum information management, using GIT as a version control and Bitbucket as a remote repository, Slack made us work remotely and share resources and MeisterTask to track small tasks. If we would start the project today, we would use the same tools and frameworks as we did so far, because it fit perfectly together for us.

SCRUM

To begin with, the main methodology was Agile Scrum development along with Unified Process phases. In other words, scrum helped to implement Agile methodology and Unified Process to manage sprints. Reading sprints retrospective can help understand in detail all problems that we faced in this project.

(Scrum information can be found in Appendix D, F)

Group roles:

1. Andrei Balanuta – Product Owner
2. Lukas Vaisnoras – Scrum Master
3. All members – Development Team

Product Owner

The product owner was responsible for enlarging value to the team. The one that was liable for Product Backlog and that everyone in the team understood each User Story value. Moreover, he also was responsible for sorting User Stories in the product backlog as well as separating them for each future sprint. Having in depth understanding of each item's value, he was also responsible that every user story is fully implemented and deployed.

Scrum Master

Scrum master was one of the most influential and collaborative team members in the group. He kept all team members up to date with new information as well as tracking everything that happened inside the group. Throughout execution part, scrum master had to work around a lot because of some stressful sprints and risky moments. All in all, he resisted to big changes in agile development and got us to the right path towards achieving project goals.

Development team

Because we were two in the group, we decided that taking both of us this role will be a great idea. During execution we were responsible of solving sprint backlog tasks. Working together made us solve some problems faster and see beyond our blind spots. Taking two scrum roles at the same time made us less self-organized because in some cases we have been thinking from the wrong role perspective. To conclude, managing scrum in two people was hard but achievable.

Product Backlog

Product backlog was the first step before going into sprints. We have developed backlog all together and after all Product Owner filtered it and ordered it, so it looked achievable and easy to understand. Also, we decided to use Story Points instead of hours because we found this approach easier for estimating time. Even if we didn’t implement all the product backlog items, we provided enough functionality to software and usability (Product backlog can be found in Appendix E).

Sprint Planning

Before starting a sprint, sprint planning is required in order to think about strategies of how to achieve certain milestones. Each sprint planning was different and unique, starting from sprint 2 – retrospective has helped us to plan all sprints because of experience gained form previous sprints. Moreover, a few sprints met our expectations but not all of them. Sprint planning had an important role in the scrum process. For example, because of sprint planning we didn’t implement two backlog items by setting too many tasks in less time. To sum up, we learned to plan before starting and to analyses and design before implementing (Sprint planning logs can be found in Appendix F).

Sprint Backlog

Each sprint has its backlog and it’s a result of the sprint planning process. After a sprint is done, next user stories are selected in the next sprint backlog in order. Selected items are chosen in a way that we fit in the sprint deadline. Backlog item weight is calculated using story points and it is proportional to team’s potential velocity. In some cases, we both worked on a specific task and in other we divided different tasks for each other for efficiency increment (Each sprint backlog can be found in Appendix D).

Burndown Charts

Burndown chart is a graphical representation of how tasks were achieved in relation to the sprint deadline. A perfect chart would be a straight line that is pointing downwards. The burndown chart was updated after each finished item in order to see how much work we have left to do. Viewing the whole picture of the chart made us feel more stressed in some sprints and got us work harder in order to finish sprint backlog in time. At the end of the sprint, the finished chart helped us to write a more concise retrospective as we could se the whole efficiency picture. Below are examples of an efficient and inefficient burndowns experienced in this project (Burndown chards can be found in the Appendix D).

Figure 1 – Efficient burndown chart (Appendix G)

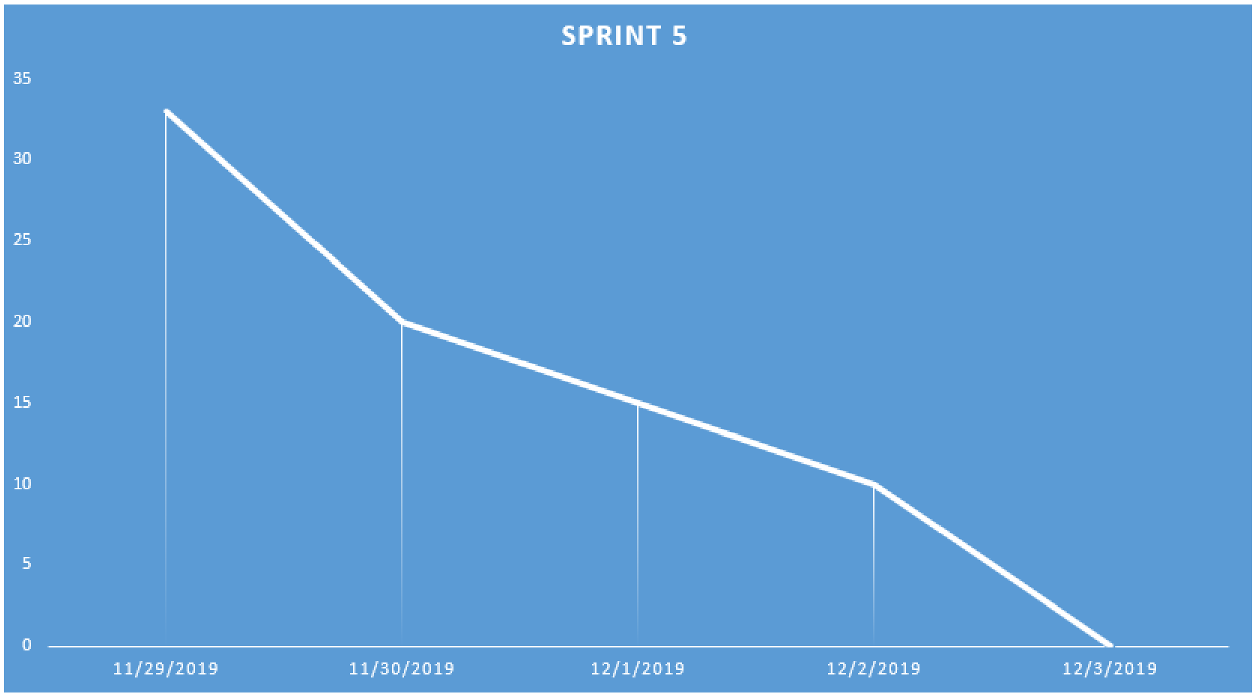
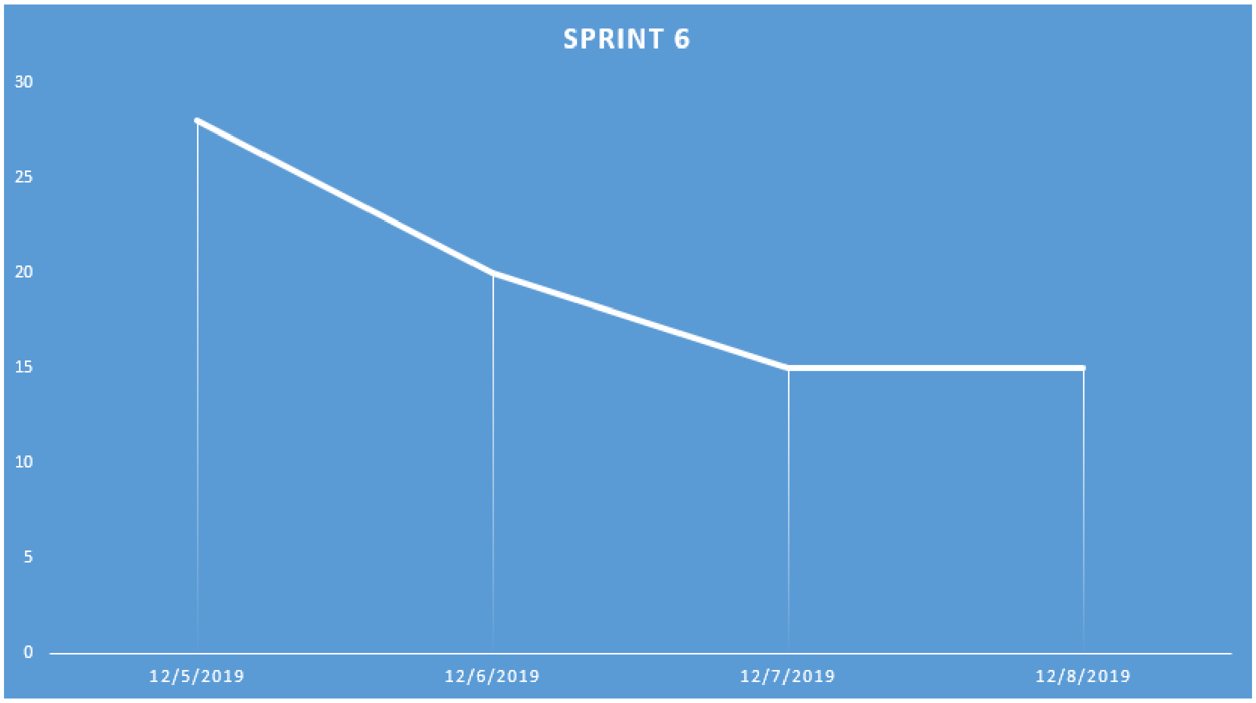


Figure 2 – Inefficient burndown chart (Appendix G)


SPRINT 6
30
25
20
15
10
5
0
12/5/2019
12/6/2019
12/7/2019
12/8/2019

Sprint Review and Retrospective

At the end of each sprint, we've been meeting and discussing about what we have achieved and what we can change in the next sprint for efficiency improvement. Sprint review has helped us to make a summary and to see what we have done so far in the sprint. Each team member came with a personal review and after that we created a common one. The retrospective is what was good or bad, and what can be changed in the next sprint in order to increment out approach capabilities. We have recorded all sprint reviews and retrospectives in the excel sheets (Appendix D).

Group Reflections

This project was a great challenge and we have developed a lot of skills working as a group. Scrum made us work together more closely and it taught us a lot about how to cooperate in different situations. We learned how to help each other work with resources that we had. Team management was very hard to handle but at the end of each sprint we have been doing in depth research on our mistakes and achievements, that led us to understand our weak points and focus on them in the future. Therefore, we experienced some moments that we could forecast exactly what will happen in the next week or sprint. We love how we evolved as a team and hope that we won't stop at this point. All in all, learning from each other's mistakes made us behave more accurate and patient. As a group we understood that working alone is way less efficient that with a group and asking for help is not a bad approach to solve a certain problem.

Project Results

So far this was not only the best programming experience, but also, we learned about time management, analysis, design and other important parts of a project development. We are happy to say that we are satisfied with the outcome of this project and now looking at the path that we have pursued so far, we are joyful that this project will be a part of our backgrounds.

During the project period we have been facing a lot of problems and risks. The biggest risk was the time management because of the huge amount of work we had to do. Therefore, to minimize this risk we focused more on sprint planning and tracked our scrum process continuously. Another risk was performance, due to the reason that our project was a combination of three courses, we had to attend classes and finish our assignments in order to prepare for software development. However, we started working on project before knowing how to develop it, but that's how the learning process works. The bad part is that we didn't implement all requirements that we have set at the beginning because we didn't know that at the end It will turn that complex for us. However, knowing that software is usable and most of the requirements are done makes us feel proud of our work.

Personal Reflections

Lukas Vaisnoras

The fact that it was already 3 rd. semester project, made me more confident because of the experience which I gained during first 2 projects. All of us knew sequence of the project, and we had supervisors as our helping hand. Everything was clear and perfectly presented during the semester. We had all access to the information that was very important and useful for us.

The project was harder to achieve, because of different and completely new things we needed to use, but also work went a lot smoother than previous ones because we already had experience working with SCRUM, that made our time well-planned, of course we still had minor problems working with it because we are still learning, but every mistake that we made was a lesson that all of us will remember.

Furthermore, I really enjoyed working with my new group, all members knew how to efficiently communicate with each other and knew how to plan time. We had no conflicts during the project which made our work more productive. We all got to know each other fast and that was the reason why we felt more comfortable and confident working together.

All in all, the project did not complete all requirements that we planned, but anyways I am satisfied of what we achieved, and I am happy with the outcome that we got. Without doubt I feel great about the stuff I learned during this time, and I am confident that I will use what I learned in the future projects.

Andrei Balanuta

To begin with, I knew that third semester will be much harder than the second, but I didn’t expect that we will spend that amount of time working on this project. The work we put into the project made us feel very confident using new technologies. Comparing all semesters, I can say that everyone grows exponentially as a software developer as the time passes.

Moreover, I am happy that we finished this project with success despite all challenges and stress we have faced so far. At the beginning I was looking for a strong group that would accept me as who I am and would be highly dedicated working on the project. I found Lukas and Deivydas and they took me in their group with no doubts. Deivydas had some high programming skills and years of experience and I learnt from him a lot of things whereas Lukas was just a simple student as me. As the time passed, Deivydas left the group because of some personal reasons. When he left, Lukas and I realized that we must do all the work together and we started to feel stressed about that. It turned out that we did a great job together and besides developing a good software we also developed a strong friendship.

Furthermore, I really liked the technologies that I have used and the 3-tier architecture design. The analysis part was the hardest one because we had to imagine how software would look and not knowing how to use technologies and how they work together I felt frustrated because everything went very slowly, but we managed to finish this part with success. I have learnt a lot of techniques and strategies during project execution and there was a moment when the software started to gain shape and I was surprised how fast information flows through tiers and how powerful this is. Using scrum helped us a lot and I will use this framework in my future projects because it pushes you to achieve your goals even in cases when you think that there is no solution for a problem.

To continue with, this semester I realized that I am passionate about software engineering and I will enjoy working in this field. Developing this software made me feel like I am working on a real-life project and I have put all my dedication into it. Even if it's not perfect, I love it because It's my creation and after handing in the project I am thinking about making it better and put it into my portfolio for finding an internship.

To sum up, experience I gained this semester is the best one and I hope that next semesters I will learn more new interesting technologies that will challenge me developing a new software. I am glad that I achieved this with Lukas, and I hope we will surprise our supervisors with this project.

Supervision

During the project we did not have any supervision meetings because we thought that everything went well. At the end of semester, we realized that it would be better to get some advices from teachers because they would show us the correct path to follow. We realize that it was a big mistake not making any supervision meetings and for the future we will work on this issue.

References

Guidelines, V.I.A.E., 2018. SEMESTER & BACHELOR PROJECTS. (August).

Guidelines, V.I.A.E., 2018. APPENDIX 2 Process Report. (August).

Guidelines, V.I.A.E., 2018. APPENDIX 4 Supervision.

Rasmussen, J.K. and Vaisnoras, L., 2019. Project Description Support Document - VIA Software Technology Engineering. (September), pp.1–9.

Anon n.d. No Title. [online] Available at: <https://www.hofstede-insights.com/>. [Accessed 1 December]

Schwaber, K. and Sutherland, J., 2017. The Scrum Guide: The Definitive The Rules of the Game. *Scrum.Org and ScrumInc*, [online] (November), p.19. Available at: <http://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-US.pdf>.[Accessed 19 December]

Appendices

Appendix A – Group Contract

Appendix B – Project Description

Appendix C – Hofstede Diagram

Appendix D – SCRUM

Appendix E – Product Backlog

Appendix F – Sprint Planning

Appendix G - Document Figures