Architecture and Design

Lezioni alla pari

April 19, 2020

Team Members

Ovidiu Andrioaia
David Cirdan
Luciano Mateias
Zhiyang Xia

Document Control

Change History

Revision	Change Date	Description of changes
V1.0	04/19/2020	Initial release

Document storage

This document is stored in the project's GIT repository at:
https://github.com/KilliKrate/Software-Documentation-G6/blob/master/docs/Architecture%20and%20Design/index.md

Document Owner

Group 6 is responsible for developing and maintaining this document.

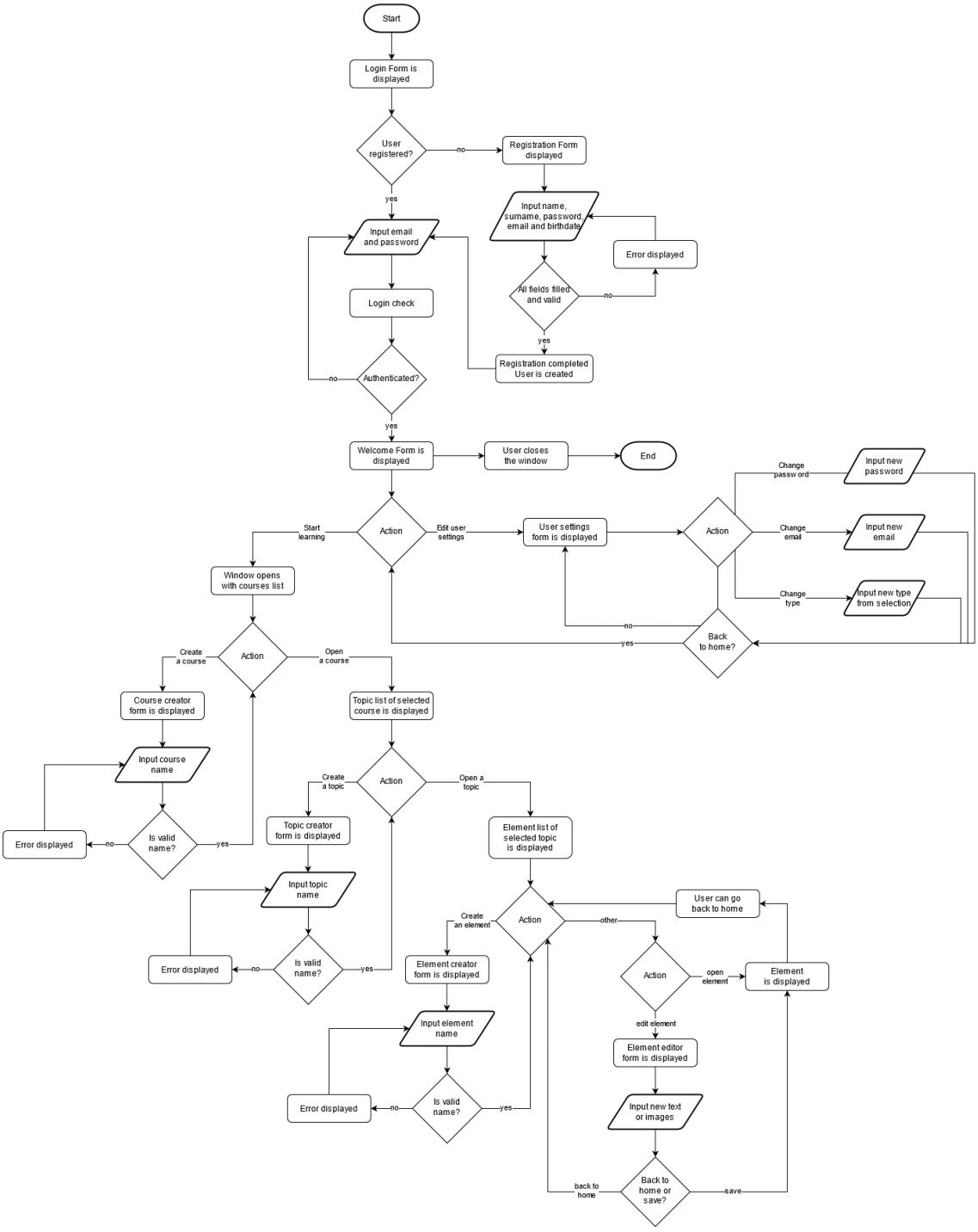
Table of contents

Introduction

The Lezioni alla Pari Architecture Document is designed to illustrate and identify the high level architecture used to design and implement the Lezioni alla Pari application. The document contains an overall view of the system hierarchy, logical views of the system components, and a process view of the system's communication.

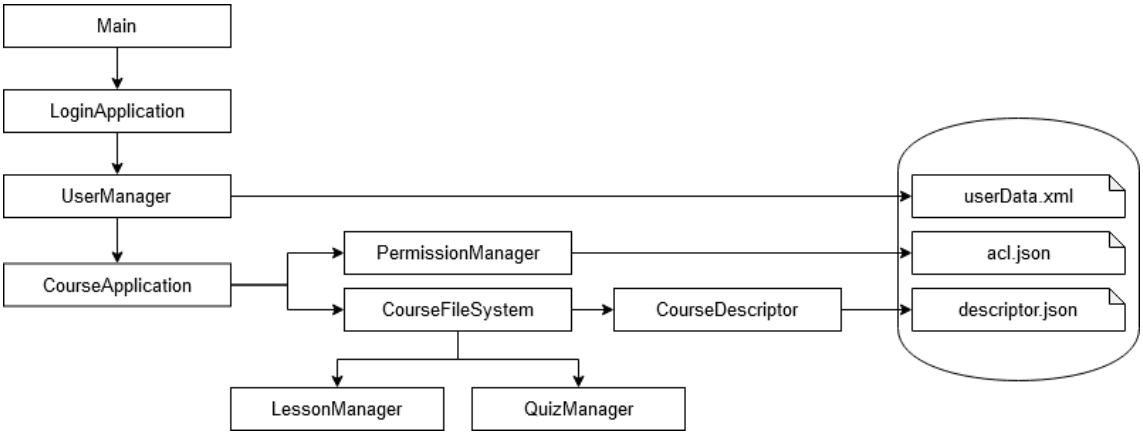
Back to Top

Flow Chart



Back to Top

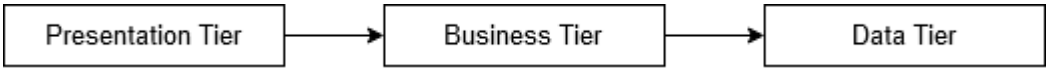
Logic Diagram



Back to Top

High Level Hierarchy

Hierarchy Diagram



Hierarchy Description

The architecture system for the Lezioni alla Pari application is a 3-tier application.

Back to Top

Components Classification

Presentation Tier

Purpose: To display forms, controls, images, videos to the user to create fluid and efficient user experience.

Specific Nature: The presentation tier will be in charge of displaying appropriate images, menus and videos to the user. This tier will also be in charge of handling left and right clicks. When a user clicks a menu on the GUI, the code corresponding to that event will be called. This tier will also be in charge of the spawning of appropriate threads. The need of spawning extra threads is due to the fact that the main thread of the app will be watching for event clicks, but we also need another thread constantly running to send asynchronous requests to the webserver.

Subcomponents: Quill WYSIWYG editor

- **Quill WYSIWYG editor** - Quill is a text editor that is used whenever a user, owner of a lesson or a quiz, wants to update it. With Quill we can easily upload images and videos that will make our lessons more approachable and intuitive. Styling text and entering formulas are especially useful features.

Business Tier

Purpose: Processes and responds to events, typically user actions. This tier is in charge of the heavy algorithm business logic found in complex solutions.

Specific Nature: The Business Tier is the core of our program, it will be in charge of responding to user requests and to interact with th

Associated Constructs: CourseApplication, CourseFileSystem

- **CourseApplication** - The CourseApplication class will be responsible of processing data and serving webpages to the presentation tier, it coordinates all the other classes. This class interacts with almost everything, from loading html lessons from the filesystem to creating and editing quizzes.
- **CourseFileSystem** - The CourseFileSystem class will be responsible of the interaction between the application and the filesystem by opening, reading and writing files in a prestuctured way.

Data Tier

Purpose: This tier is in charge of storing data in persistent storage.

Specific Nature: This tier will consist of XML and JSON files.These together will be our database management system. There will be 1 XML file named *user_data.xml* and 2 JSON files named *acl.json* and *descriptor.json*.

Associated Constructs: UserManager, PermissionManager, CourseDescriptor

- **UserManager** - UserManager will be used to get, add, update and remove a user in our platform. Data regarding users will be stored in the file *acl.json*, this construct will be in charge of interacting with that file.
  - Example of *user_data.xml*

```
<users usercounter="1">
    <user active="true" id="u-1">
        <name>John</name>
        <surname>Doe</surname>
        <password>johndoe420%</password>
        <email>johndoe@mail.com</email>
        <birthdate>10-10-1990</birthdate>
    </user>
</users>
```

- **PermissionManager** - PermissionManager will be used to store all the permissions of one or more user. Read and Write are the types of permissions that can be given to a user to prevent the access or the modify of a course and its topics and lessons/quizzes. Data regarding permissions will be stored in the file *acl.json*, this construct will be in charge of the interaction with that file.
  - Example of *acl.json*

```
{
    "courses": {
        "c-2": {
            "everyone": false,
```

```
            "u-1": "rw"
        },
        "c-4": {
            "everyone": false,
            "u-1": "r"
        }
    }
}
```

- **CourseDescriptor** - CourseDescriptor will be used to store the current state of the logical filesystem of courses, topics and lessons/quizzes. This construct is essential, it works like an *inode* store in a Linux filesystem. Data regarding this logical filesystem will be stored in the file *descriptor.json*, this construct will be in charge of the interaction with that file.
    - Example of *descriptor.json*

```
{
    "courses counter": 11,
    "topics counter": 20,
    "elements counter": 28,
    "courses": {
        "c-2": {
            "name": "Hello World, but this one is mine",
            "topics": {
                "t-2": {
                    "name": "You doing ok?",
                    "elements": {
                        "e-10": {
                            "name": "I think you are",
                            "type": "lesson",
                            "creation date": "2019-05-
24T23:30:29.271315",
                            "edit date": "2019-05-24T23:30:29.271315",
                            "delete date": null
                        },
                        "e-27": {
                            "name": "How to make potatoes",
                            "type": "quiz",
                            "creation date": "2019-09-
08T14:04:41.965836",
                            "edit date": "2019-09-08T14:04:41.965836",
                            "delete date": null
                        }
                    },
                    "creation date": "2019-05-24T23:30:24.424276",
                    "delete date": null
                },
                "t-19": {
                    "name": "New Topic!?!?!?!",
                    "elements": {
                        "e-22": {
```

```
                            "name": "Hey",
                            "type": "lesson",
                            "creation date": "2019-09-
04T18:12:59.636303",
                            "edit date": "2019-09-04T18:12:59.636303",
                            "delete date": null
                        },
                        "e-23": {
                            "name": "test",
                            "type": "lesson",
                            "creation date": "2019-09-
08T12:06:19.076754",
                            "edit date": "2019-09-08T12:06:19.076754",
                            "delete date": null
                        }
                    },
                    "creation date": "2019-09-04T18:01:46.662281",
                    "delete date": null
                }
            },
            "creation date": "2019-05-24T23:29:36.936242",
            "delete date": null
        },
    }
}
```


Back to Top

Process View

Process View Description

The Process View is essential in understanding how the separate components and subcomponents communicate with each other in a concurrent application. By better understanding the communication between components, it may be possible to optimize the data flow and storage of the application, as well as ensuring thread-safety.

Application View

This view is the main application view that is created at runtime of the program. The program creates the view: this is not a user created view. This view handles the basic program flow by controlling navigation between items, videos, quizzes, including the handling of user input to the graphical forms.

Presentation View

This view is user created, when the application enters the Course/Topic/Lesson management mode. This view is the main one, responsible for almost every action in the application. From this view an authorized user, based on its permissions, can create an item on the platform and give read and write access to other users.

Login View

This view is created when opening the application. This view manages the login, and the registration if needed, of users in order to give them the proper authorizations.

Registration View

This view is user created, when the users chooses to create a new account from the Login view. This view handles the basic operations of creating a new user on the platform.

Back to Top