SOFTWARE DESIGN

LABORATORY INTRODUCTION

FOREWORD

ABOUT MYSELF

- Graduated UTCN three years ago *(time flies)*,
- Lead developer and solution architect at a corporation in Cluj,
- Trying to be active in open source and on Stack Overflow,
- Genuinely got into teaching just to help spread know-how.

PRESENT YOURSELVES

LAB. GOALS

- Gain useful **skills** and **experience** for your later career.
- Become familiar with typical patterns and styles.
- Work with commonly used technologies.
- Improve your code-writing style.
- Get used to **think** first, code later.

NOT LAB. GOALS

- Learn *by heart* tens of design patterns that you will barely use.
- Learn about a "silver bullet" and:
  - Get fixated on a *single technology* and extrapolate concepts from it
  - Get fixated on a *single pattern or style* and use it all over the place
- Write a ton documentation that **nobody** will ever read.

MY PERSONAL GOALS

- Help you *clarify concepts* and fill in gaps (even from other disciplines).
- Provide reasonable advice regarding to your designs and code.
- Promote *healthy discussions* about technology and the surrounding industry.

RULES

ATTENDANCE

- Laboratory attendance is **mandatory**: you can have maximum three absences during the semester.
- You **cannot** switch groups, each assistant has his own rules.
- You can switch semi-groups if you swap with someone else:
  - Write me an email with the other person in CC beforehand.

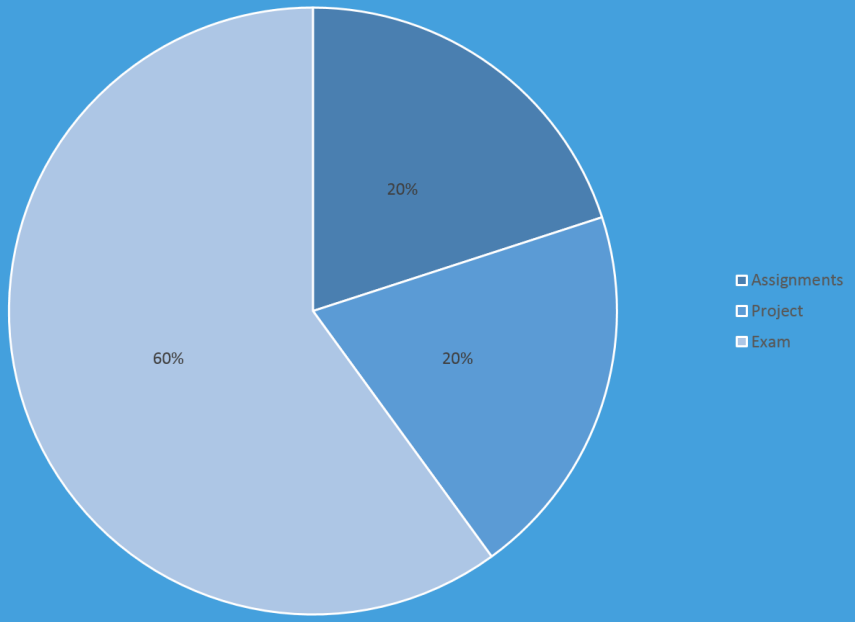
NO CHEATING

GRADING

Two types of laboratory work:

- [20%] Assignments
- [20%] Project

In total, the lab counts towards **40%** of your final grade.



ASSIGNMENTS

Three assignments in total.

- Each will be graded from 1 to 10.
- Each must get a grade **above or equal to 5**.
- The final grade for the assignments part is the average of the three.

You can make improvements on your assignment later for a better grade.

Each assignment will have two parts:

- Documentation for your design,
- Implementation of the design.

PROJECT

You must build a final project until the end of the semester.

During the semester, you will have to hand in three deliverables (which will help you build the documentation and design the system).

Grading:

- [30%] Deliverables,
- [30%] Final documentation,
- [40%] Implementation.

BONUS POINTS

For the assignment topic there will be two bonus features. Also, for each assignment there will be several bonus requirements which you can implement for bonus points.

You can use your bonus points to increase your assignment grade.

Additionally, if you:

- Implement both bonus features in all three assignments,
- Get grades >= 9 for all three assignments,
- And accumulate at least 9 bonus points.

You are excused from implementing the final project, as the assignment becomes your project; you get a 10 for both the implementation and the documentation part (the deliverables still need to be handed-in).

DEADLINES

Each assignment and project deliverable has a deadline on the schedule.

You may hand it in with *at most* **two weeks** delay, for which you will lose **1 point** per week.

DELIVERY

You cannot hand in more than one assignment (including improvements) and one deliverable per week.

This implies that you may hand in both an assignment and a deliverable in the same week.

RESOURCES

CONTACT

You can always contact me via email: Serban.Petrescu@outlook.com.

Don't hesitate to email me for any questions; I will try my best to answer within a day.

I encourage you to use Stack Overflow proactively:

- If you cannot find an already existing question for your topic, create one yourself.

LINKS

- Patterns of Enterprise Application Architecture
- Gang of Four Design Patterns
- Uncle Bob's Books (unofficial link):
  - Clean Code
  - Clean Architecture
- GitHub resources:
  - Our lab's resources
  - Last year's resources
  - Buzea's examples

TOOLING

I strongly suggest that use **Java** for building your project and assignments.

Optionally, for the U.I., you can use **JavaScript**.

We will use **GitHub** for sharing the code. Please create an account if you don't already have one.

My suggestion is to work on your personal machines if possible.

TOOLING - CONT.

Recommended tools (please use the **latest** versions available):

- [git] GitHub desktop,
- [java] at least JDK 8,
- [java] IntelliJ Community (or Ultimate with a student license),
- [js] Visual Studio Code,
- [db] MySQL.

ASSIGNMENTS

ASSIGNMENT TOPIC

We will choose together a **single** topic for the whole semi-group.

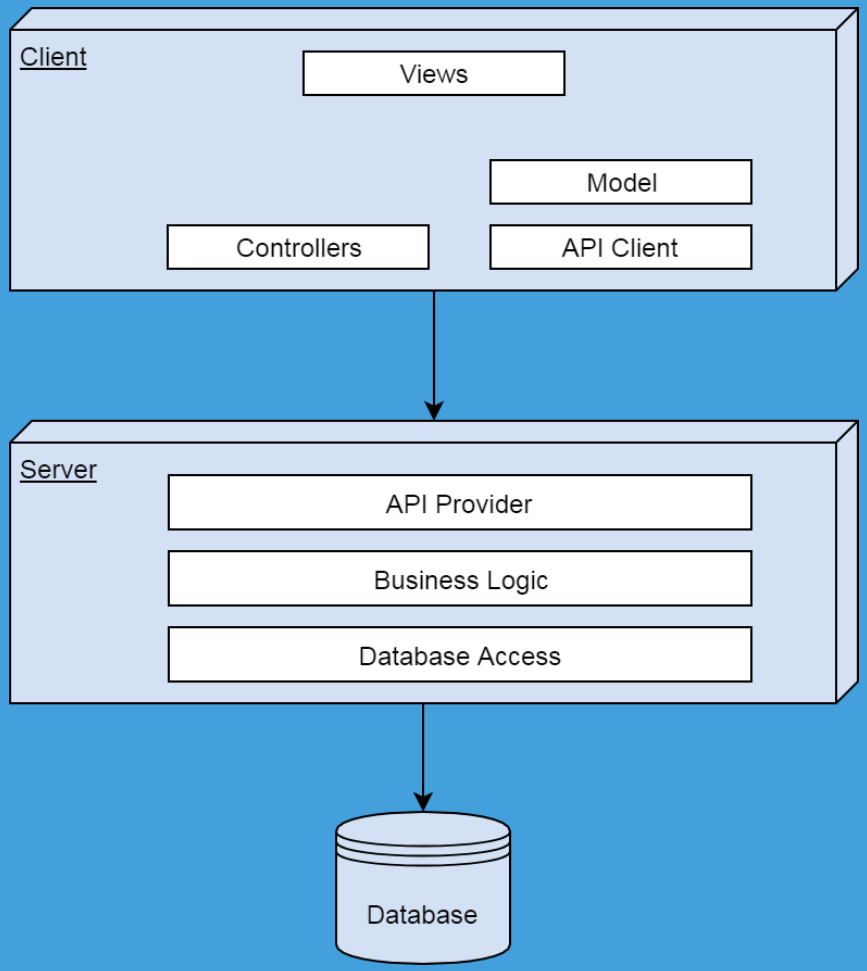
The assignments will then be centered on this topic.

ASSIGNMENT ARCHITECTURE

We will build a application for a given business scenario.

In each assignment, we will incrementally add to this system:

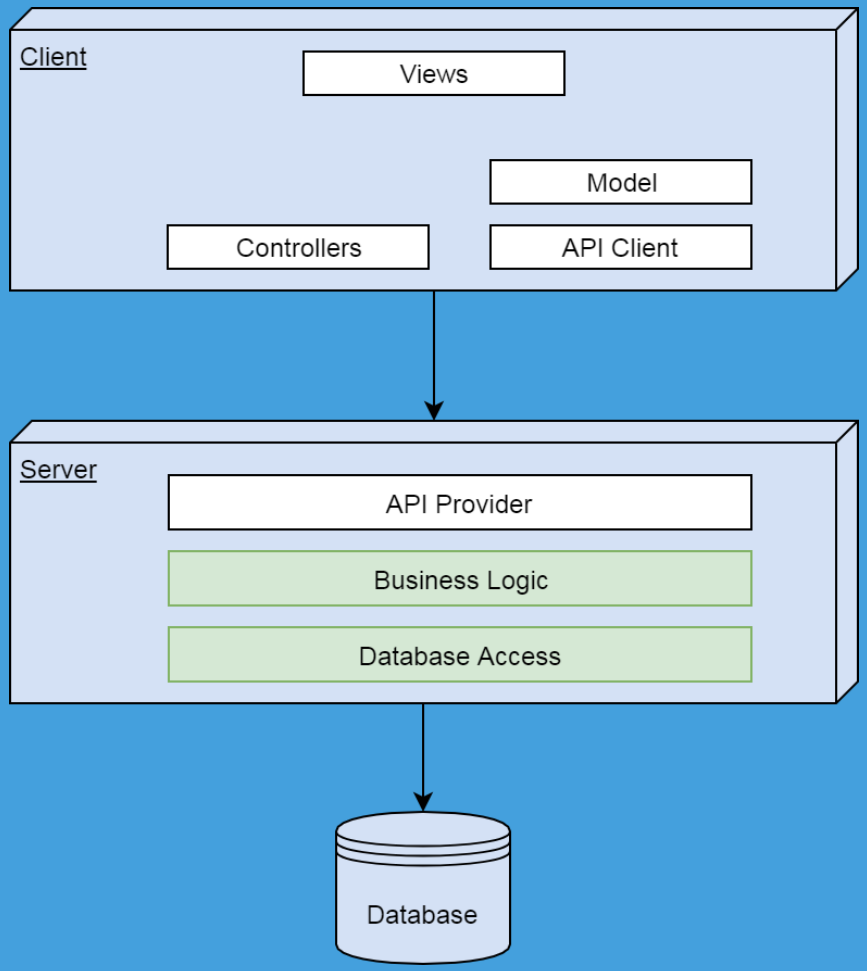
1. The server side (layered architecture),
2. The client side (model-view-controller),
3. The connection between the two (client-server).



ASSIGNMENT 1

- **Layered** architecture,
- Abstract factory,
- One data source pattern
- One domain logic pattern
- No U.I. - just a **console** app.
- Unit tests.

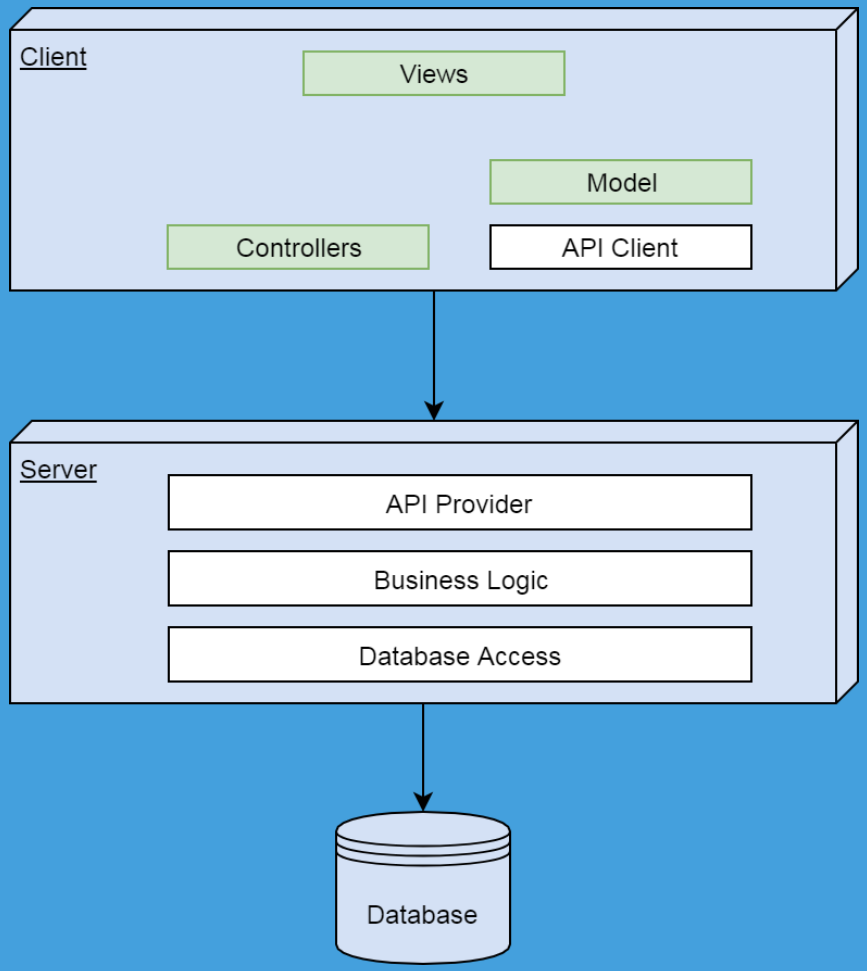
- *Integration tests.*



ASSIGNMENT 2

- MVC architecture,
- Command pattern,
- Builder, composite or decorator patterns,
- Just the U.I - without any backend.

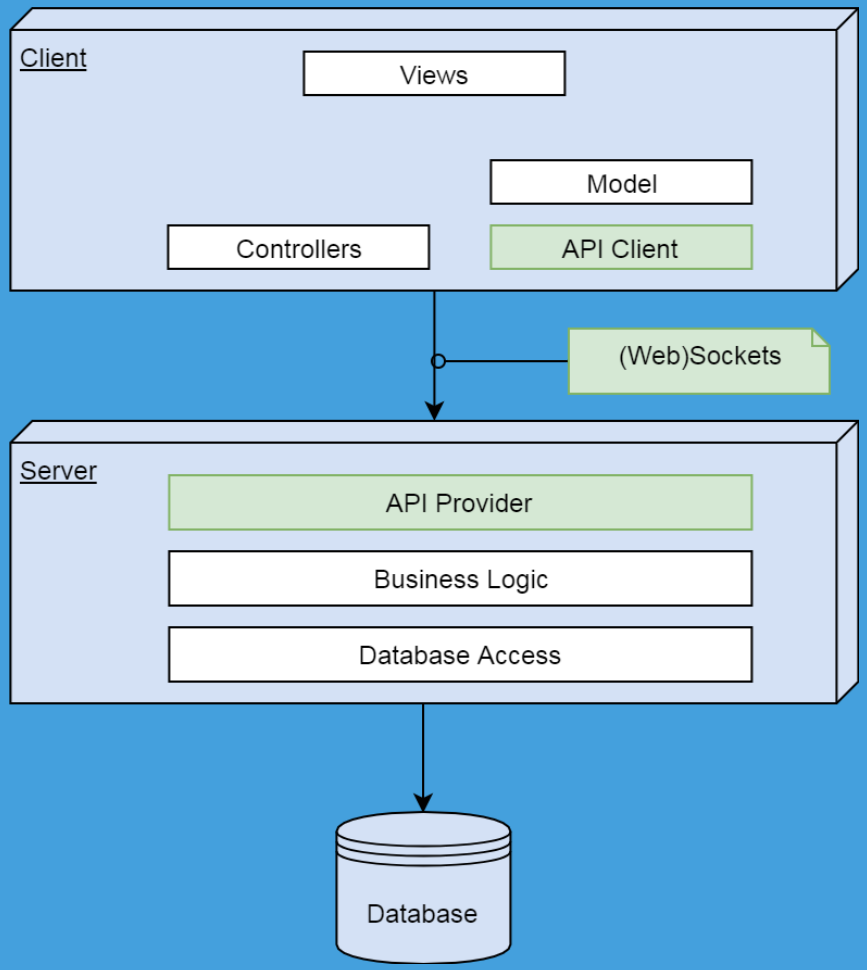
- *U.I. tests (browser-based).*



ASSIGNMENT 3

- Client-server architecture,
- Observer pattern,
- Data transfer objects,
- Command pattern (again),
- Will connect the U.I. with the backend.

- *End-to-end tests.*



PROJECT

PROJECT TOPIC

Each student will have to come up with his own project idea.

You can find some examples here: GitHub Starter.

My recommendation: find something that you are interested in.

Constraints:

- Must have users.
- At least three entities (apart from users).
- At least three use cases (apart from login / logout).

PROJECT ARCHITECTURE

Must be a **web application** (either Spring Boot or NodeJs).

User interface can either be rendered server side (e.g. Thymeleaf) or client side (e.g. Angular).

Desktop app is also acceptable, but with a penalty of 2 points.

Data is stored in a database and data access is done via **ORM** (JPA).

PROJECT DELIVERABLES

1. Vision, Use Case Model, Supplementary Specification, Glossary
2. Domain Model, Architectural Design, Component and Deployment
3. Design Model, Data Model

All these documents must use the following RUP templates: GitHub Buzea.

ROLES IN THE INDUSTRY

For this project, you will do tasks which pertain to several different roles inside a typical IT company.

Basically, you will be the:

- Product owner / business analyst,
- Architect / lead developer,
- Developer / tester,
- User.

My advice: try to fulfill the tasks as best as you can to have a (very basic) idea of what each role entails.

TODAY'S TASKS

TODAY'S TASKS

- Create a GitHub account.
- Access the test GitHub classroom assignment and do a commit.
- Think about a project topic.
- Answer this quiz.
- Discuss a about UML, OOP, etc.