HOGESCHOOL ROTTERDAM / CMI

Development 5 INFDEV04-5

2017-2018

Number of study points:	4 ects
Course owners:	M. Abbadi & A. Omar

1 Development 5 (INFDEV04-5)

1.1 Study points and contact time

The course awards students 4 ECTS, in correspondence with 112 hours of student work.
The course consists of seven frontal lectures and seven practicums. The rest is self study.

1.2 Introduction

This is the course descriptor for the *Development 5* course.
Development 5 covers a presentation of modern, distributed (over HTTP) applications which follow the MVC architectural pattern. Given the huge breadth of usable technologies in the field, we will focus the implementation on a single stack: ASP.Net Core, Postgresql, and React over TypeScript.
An important reminder: the course is not meant to be a *build a website in 16 hours* workshop. This would collide with the philosophy of the Informatica degree, which aims at giving the foundational tools to empower students as ongoing learners. For this reason, we will not dive deeply into the intricacies of a given technology, but only use the minimum needed to understand the underlying concept(s) and move on. As a student, it is highly desirable to realize that most likely each of you will use *different technologies* on the workplace, and will have to *keep learn new ones*, so focusing on a single language, stack, or library would do more harm than good.

1.2.1 Learning goals

The course has the following learning goals: - (PR_M) students can implement the model of a given MVC application in order to interact with a permanent storage structure; - (PR_C) students can work with a given controller of an MVC application in order to handle interactions with the model; - (PR_V) students can work with the view of a given MVC application in order to add interaction elements and improve safeness.
The course, and therefore also the learning goals, are limited to idiomatic C# and ASP.Net Core constructs, and idiomatic TypeScript and React constructs (augmented with a few essential *npm* libraries such as Immutablejs and Fetch).

1.2.2 Competences

Realization

1.2.3 Exam

The exam consists of a series of exercises where students, given partial code and the desired state transitions, are requested to fill in the missing code that matches the given state transitions.
The exercises are split as follows:

- Part 1, 25% of the exercises, requires students to complete a model implementation;
- Part 2, 25% of the exercises, requires students to complete a controller implementation;
- Part 3, 50% of the exercises, requires students to complete a view implementation.

For example the exam could be made of 8 exercises: 2 about the model, 2 about the controller and 4 about the view.

This reflects the relatively higher emphasis on interaction that modern distributed applications are showing. This leads development time to be split non-uniformly across the architectural elements.

1.2.3.1 Scoring

The exam results in a full grade (from 0 to 10). Each exercise awards one single point, if correctly completed. The grade is computed as the percentage of points obtained, divided by 10. Students who score at least 55% of the total points will get a passing grade. For example, if a student completes 5 exercises out of 8, this means obtaining 5 points out of 8, which means 62,5% and corresponds thus to a 6,25 (62,5/10).
[3:53] because it's impossible to have an exam made by 5 exercises (given the 25/25/50 percentages) #### Retake If the exam is not passed, then it will need to be retaken during the current schoolyear. The retake will be scheduled at the end of the following period.

1.2.3.2 Exam matrix

The exam covers all learning goals.

Exam part	Part 1	Part 2	Part 3
PR_M	V		
PR_C		V	
PR_V			V

1.3 Lecture plan

The course is made up of seven lectures and practicums. During the lectures both theoretical concepts and applied examples will be covered. During the practicum the students will be asked to practice independently (with the help of teachers when needed) the applied examples seen in the lectures. The students can experiment in the practicums. All the work done in the practicums can be seen as formative exercises in preparation of the exam.

1.3.1 Chapter 1 - Introduction to distributed applications

- The characteristics of distributed applications
- The Model-View-Controller (MVC) design pattern
- Object-relational mapper (ORM)
- The M in MVC

1.3.2 Chapter 2 - Modeling queries and managing data

- Impedence mismatch
- Mapping data from relational/physical model to domain models
- A genereric model to safely query relational models
- The costs of accessing data
- Improving queries safeness through typing (LINQ)

1.3.3 Chapter 3 - Controlling the data-flow in the application

- Taming the complexity of models in a distributed application
- The C in MVC to narrow the access to the model
- HTTP Protocol
- Architecturing distributed applications through the REST-model and testing

1.3.4 Chapter 4 - Rendering instances of the model

- The V in the MVC
- Serving static pages with template engines
- The challenge of interacting with the model

1.3.5 Chapter 5 - Towards a new rendering architecture

- Client side/server independent programming
- Managing state on the client (Javascript and the DOM)

1.3.6 Chapter 6 - Single page application

- Putting in relation model and view in the client
- The concepts of containers and components in React
- The callback model

1.3.7 Chapter 7 - Increasing safeness on client side

- Validating state access through types
- Typescript as superset of Javascript to guarantee correct usages of the model
- Implication of using types in structuring the application

1.4 Learning materials

- Materials used in the lessons: `https://github.com/hogeschool/Development-5/tree/2017-2018/Lectures`
- Entity framework core online documentation: `https://docs.microsoft.com/en-us/ef/core/index`
- Asp.net core online documentation: `https://docs.microsoft.com/en-us/aspnet/core/`
- React online documentation: `https://facebook.github.io/react/docs/hello-world.html`
- Typescript online documentation: `https://www.typescriptlang.org/docs/handbook/react-&-webpack.html`