HOGESCHOOL ROTTERDAM / CMI

Web Development

INFWEB01-D AND INFWEB21-D

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

Module description

Module name:	Web Development
Module code:	INFWEB01-D AND INFWEB21-D
Study points and hours of effort for full-time students:	This module gives 4 ects, in correspondence with 112 hours: • 3 x 7 hours frontal lecture • the rest is self-study
Examination:	Written examination and practical assessment
Course structure:	Lectures
Prerequisite knowledge:	Object oriented programming
Learning tools:	• Online materials of used technologies (see section learning materials) • Lessons outline: `https://github.com/hogeschool/` then navigate to development WEB repository
Connected to competences:	• Realisation • Analysis
Learning objectives:	The course has the following learning goals: • (U_M) students understand the model of a given MVC application in order to interact with a permanent storage structure; • (U_C) students understand a given controller of an MVC application in order to handle interactions with the model; • (U_V) students understand the view of a given MVC application in order to add interaction elements and improve safeness. • (PR_M) students can work with 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 .Net Core constructs, and idiomatic TypeScript and React constructs (augmented with a few essential *npm* libraries such as Immutablejs and Fetch).
Required tools	a
Course owners:	M. Abbadi and A. Omar
Date:	14 november 2019

1 General description

This is the course descriptor for the *Development Web* course.
Web Development 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.

1.1 Relationship with other teaching units

This course builds upon the development courses of the first year.
Knowledge acquired the course Web Development is also useful for some of the projects. A word of warning though: projects and development courses are largely independent, so some things that a student learns during the development courses are not used in the projects, some things that a student learns during the development courses are indeed used in the projects, but some things done in the projects are learned within the context of the project and not within the development courses.

2 Course program

The course is made up of seven lectures. During the lectures both theoretical concepts and applied examples will be covered. Units do not necessarily match with the week schedule. For example, the content of unit 1 can be covered in more than one consecutive lesson.

Units	Topics
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
2	Modeling-queries-and-managing-data • Impedance mismatch • Mapping data from relational/physical model to domain models • A generic model to safely query relational models • The costs of accessing data • Improving queries safeness through typing (LINQ)
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
4	Rendering-instances-of-the-model • The V in the MVC • Serving static pages with template engines • The challenge of interacting with the model
5	Towards-a-new-rendering-architecture • Client side/server independent programming • Managing state on the client (JavaScript and the DOM)
6	Single-page-application • Putting in relation model and view in the client • The concepts of containers and components in React • The callback model
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

2.1 Learning materials

- Materials used in the lessons: `https://github.com/hogeschool/` then navigate to development 5 repository
- Entity framework core online documentation: `https://docs.microsoft.com/en-us/ef/core/index`
- React online documentation: `https://facebook.github.io/react/docs/hello-world.html`
- Typescript online documentation: `https://www.typescriptlang.org/docs/home.html`

3 Assessment

Theory

The theory check consists of 20 multiple choice questions. The questions will be selected from the list of questions available on Github in the course repository. Each multiple choice question is associated to 0.5 points.

Practicum

The practicum check 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.

Each exercise is associated to a certain amount of points. The total amount of points is 10. 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.

Scoring

The exam results of two grades (each from 0 to 10). You need to pass both to get the credit points of the course. In order to attend the practicum exam, you need first to pass the theory part. For the written exam you need to score at least 5,5 to pass. For the practicum exam you need to score at least 5,5 to pass.

Matrix

The exam covers all learning goals.

Exam part	Written Part	Practicum Part 1	Practicum Part 2	Practicum Part 3
U_M	V			
U_C	V			
U_V	V			
PR_M		V		
PR_C			V	
PR_V				V

3.1 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.