

NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCE DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

BSC THESIS

Developing an Electronic Classroom Platform with Ruby on Rails

Ioannis A. Efthymiou

Supervisors: Alex Delis, Professor NKUA Panagiotis Liakos, PhD Candidate

ATHENS

MARCH 2016



ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Αναπτύσσοντας μια Πλατφόρμα Ηλεκτρονικής Τάξης με Ruby on Rails

Ιωάννης Α. Ευθυμίου

Επιβλέποντες: Αλέξης Δελής, Καθηγητής ΕΚΠΑ Παναγιώτης Λιακός, Υποψήφιος Διδάκτορας

AOHNA

ΜΑΡΤΙΟΣ 2016

BSC THESIS

Developing an Electronic Classroom Platform with Ruby on Rails

Ioannis A. Efthymiou A.M.: 1115200600296

SUPERVISORS: Alex Delis, Professor NKUA Panagiotis Liakos, PhD Candidate

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Αναπτύσσοντας μια Πλατφόρμα Ηλεκτρονικής Τάξης με Ruby on Rails

Ιωάννης Α. Ευθυμίου Α.Μ.: 1115200600296

ΕΠΙΒΛΕΠΟΝΤΕΣ: Αλέξης Δελής, Καθηγητής ΕΚΠΑ Παναγιώτης Λιακός, Υποψήφιος Διδάκτορας

ABSTRACT

Objective of my thesis is to develop an e-class platform, which is a web application aiming to provide assistance to both faculty and students in managing the courses throughout each academic year. Maximizing functionality and user experience were high-valued design goals. As a result, the application feels simple, yet elegant to the end user.

SUBJECT AREA: Web Development

KEYWORDS: ruby on rails, web application, e-class, ajax, javascript

ΠΕΡΙΛΗΨΗ

Σκοπός της πτυχιακής μου εργασίας είναι η ανάπτυξη μιας πλατφόρμας ηλεκτρονικής τάξης (e-class), το οποίο είναι μια εφαρμογή διαδικτύου. Ο στόχος της εφαρμογής είναι βοηθήσει τους φοιτητές και το διδακτικό προσωπικό κατά την διάρκεια της ακαδημαϊκής χρονιάς. Η καλύτερη δυνατή λειτουργικότητα και η εμπειρία χρήσης ήταν από τους σημαντικότερους στόχους κατά την σχεδίαση. Ως αποτέλεσμα, η εφαρμογή είναι απλή, αλλά εξαιρετικά λειτουργική για τον χρήστη.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Ανάπτυξη Εφαρμογής Διαδικτύου

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: ruby on rails, εφαρμόγή διαδικτύου, ηλεκρονική τάξη, ajax, javascript

Dedicated to my Lily, my friends and family.

ACKNOWLEDGEMENTS

I would like to thank my Lily, for everything.

My brother Makis, for always being next to me.

My mother Lalila, for raising me to be the man I am.

My father Antonis, for inspiring me to do what I love.

My brother Theseas, for reminding me what it is to be young and set high goals.

The rest of my family, for sharing my life and providing a loving family.

All of my friends, each of them unique and special in my life.

All the teachers and professors, from kindergarden through university, for their role in shaping me to what I am.

Last but not least PhD candidate Panagiotis Liakos, for his amazing assistance in this thesis, and Professor Alexis Delis for inspiring me to follow this path.

TABLE OF CONTENTS

PR	ROLOGUE	12
1.	INTRODUCTION	13
1.1	Challenges in Design	13
1.2	Challenges in Development	13
1.3	Tackling the Challenges	14
1.4	Purpose of this Thesis	15
2.	THE ROLES AND FUNCTIONALITY OF E-CLASS	16
2.1	The Administrators	16
2.2	Professors	17
2.3	Undergraduate Students	23
2.4	Shared Pages	28
3.		29
3.1	The Models	29
	3.1.1 User	29
	3.1.2 Course	31
	3.1.3 StudentAttendsCourse	32
	3.1.4 Announcement	33
	3.1.5 Assignment	33
	3.1.6 Document	34
3.2	The Controllers	34
	3.2.1 The Course Controller	34
	3.2.2 The Assignment Controller	36
3.3	The Views and Helpers	36
	3.3.1 The Views	37
	3.3.2 The Helpers	37
3.4	The Gems	37
3.5	How it all ties together - A Workflow	38
4.		42

ABBREVIATIONS AND ACRONYMS	•••	 ••	•••	 •	•••	•	 •	-	•••	•	•	 •	43
REFERENCES		 											44

LIST OF FIGURES

Fig.	1:	Admininstrator - User management	16
Fig.	2:	Admininstrator - Create new course	16
Fig.	3:	Admininstrator - Course index	17
Fig.	4:	Admininstrator - Edit course	17
Fig.	5:	Professor - Personalized courses	18
Fig.	6:	Professor - Course index	18
Fig.	7:	Professor - Edit course description	19
Fig.	8:	Professor - Create course announcement	19
Fig.	9:	Professor - Edit course announcement title	20
Fig.	10:	Professor - Edit course announcement message	20
Fig.	11:	Professor - Create course assignment	21
Fig.	12:	Professor - Edit course assignment - Description only	21
Fig.	13:	Professor - Edit course assignment	22
Fig.	14:	Professor - add and delete documents	22
Fig.	15:	Professor - View homeworks	23
Fig.	16:	Professor - students attending course	23
Fig.	17:	Student - My courses	24
Fig.	18:	Student - Courses	24
Fig.	19:	Student - Enroll from individual course	25
Fig.	20:	Student - Course assignments overview	25
Fig.	21:	Student - View assignment	26
Fig.	22:	Student - Assignment not expired, homework not submitted	26
Fig.	23:	Student - Assignment not expired, homework submitted	27
Fig.	24:	Student - Document download	27

Fig. 25:	Log-in page	28
Fig. 26:	Sign-up page	28
Fig. 27:	Administrator changing the role of a user.	38

PROLOGUE

The following was written in Athens, Greece, in March 2016. It documents the development of a web application using RoR. It is distributed in hope of assisting others interested in getting started with developing web applications using RoR. However, there are *no guarantees* that the following will always be applicable, due to the everchanging nature of web technologies. Consulting with the official documentation is always recommended.

1. INTRODUCTION

Ever since the World Wide Web was made commercially available in the 1990s, it has experienced an unprecedented growth in popularity, as well as itself as an industry. Today, the majority of information we receive is online. Whether it is the news, communicating with other people, or just using the social media, we spend a lot of time online.

It is a logical next step, to take advantage of this power and use it to further assist our education. An electronic class platform (e-class) is a useful tool that makes keeping track of the courses you attend easier.

As an administrator, you can have an overview of the platform, create new courses and assign faculty members to be in charge of them.

Professors have a place where they can inform students about each of their courses, provide updates for those who are attending, give out assignments and receive homework by the students.

Students have the opportunity to find out about all the courses available, keep track of courses they attend and effortlessly enroll and withdraw from them.

1.1 Challenges in Design

The design and development of such a platform offers some very interesting challenges. Design challenges include, but are not limited to, the following.

- **Consistency:** Each page has to be personalised according to the user viewing it, while maintaining a consistent layout.
- **Intelligibility:** A website should be intuitive to use. The users, regardless of their experience or roles in the platform, should be able to easily navigate through the site and accomplish their goals.
- **Usefulness:** The application has to provide attractive alternatives to the status quo of course management. For example, posting an assignment online is more efficient than printing it out and handing it to students.

1.2 Challenges in Development

Once the hurdles of designing the application are overcome, the problems of developing it arise. Again, here is an example of such challenges, but not a full list of them.

Proper Modeling: Creating a proper modelization for the design, that conforms with the Model-View-Controller(MVC) pattern is vital for web applications, and ensures proper modularity.

- **Coding Efficiency:** A web application like an e-class offers web pages with similar, but different functionality. Depending on the user role, a page may offer different options to the user. For example, a professor should have different options when viewing a course he is in charge of, than when he is viewing one where he is not. This needs to be achieved with the minumum amount of reduntant code, since duplicate code often is a source of errors and also adds an unnecessary level of complexity to maintaining and debugging the application.
- **Security:** Security is always a concern for online applications. Whether the case is a malicious user, or just a simple hiccup on the connection, both incoming and outgoing data needs to be validated. Furthermore, users must be limited to what their role gives them access to.

1.3 Tackling the Challenges

RoR is a framework built in Ruby for developing web appications, is a perfect candidate for the task at hand. RoR is based on two pillar stones, as stated on the official RoR website.[1]

- Don't Repeat Yourself: DRY is a principle of software development which states that "Every piece of knowledge must have a single, unambiguous, authoritative representation within a system.". By not writing the same information over and over again, the code is more maintainable, more extensible, and less buggy.
- Convention Over Configuration(CoC): Rails has opinions about the best way to do
 many things in a web application, and defaults to this set of conventions, rather than
 require that you specify every minutiae through endless configuration files.

RoR is built based on addressing the coding efficiency problem discussed earlier. Additionally, RoR is by default a MVC framework, which makes proper modeling easier. Moreover, it is a widely used, open source software, with a large community. This means that there is already a ton of information out there, and someone, sometime, has already asked the same question. In case that is not true, the rails community will provide a solution soon enough. Which brings us to the next point. Like all programming languages, Ruby — and consequently RoR — has its own libraries, called "Gems".

Gems are extensively used, and quite often hold the solutions to problems that occur. In case a particular feature is requested, it is advisable to first check whether a gem that addresses it already exists, before attempting to implement a solution onelself. It is very likely that others have already made the effort to address the problem, and the solution awaits in the form of a gem. Gems can be found in Github[2], RubyGems[3] and The Ruby Toolbox.[4]

1.4 Purpose of this Thesis

In this thesis, we document the creation of an e-class platform by using RoR, how gems can be used to assist in development, and also demostrate the use of other technologies, to compliment what RoR offers. In the next part, it showcases the end result, and the functionality the application offers. Following, it goes in to detail about the code, the gems and how to properly use them.

2. THE ROLES AND FUNCTIONALITY OF E-CLASS

2.1 The Administrators

Administrators have a crucial role in the application, although their functionality is fairly limited. Administrators have the ability to:

Manage users: Have an overview of all users, and manage their roles. Since only administrators can access the user management page, a master account exists, whose role cannot be changed, to ensure there is always at least one administrator.

ECLASS	н	ome Help Manage Users Manage Courses Log out
Manage Users		
ID Name	Email	User Role
1 Sipes Daisha	admin@di.uoa.gr	Master Account
2 Langworth Myrtice	sdi0600296@di.uoa.gr	Undergrad +
3 Jakubowski Michale	professor@di.uoa.gr	Professor •
4 Murray Evert	sdi0600297@di.uoa.gr	Undergrad +
5 Little Hiram	professor2@di.uoa.gr	Professor •
6 Rolfson Hunter	sdi0500151@di.uoa.gr	Undergrad +
7 McKenzie Jean	admin2@di.uoa.gr	Admin •
8 Murphy Okey	sdi0200152@di.uoa.gr	Undergrad +
9 Donnelly Litzy	admin3@di.uoa.gr	Admin +
10 Hansen Leone	professor4@di.uoa.gr	Professor +
	Previous 1 2 3 4 5 Next	

Fig. 1: Admininstrator - User management

Create Courses: Create new courses. Administrators must set the course's name and code, and assign a professor in charge of it. A drop-down menu with all professors is provided for convenience and error avoidance.

ECLASS		Home Help	p Manage Users	Manage Courses	Log out
New Co	urse				
Code					
Name					
Professor	Bashirian Ethelyn				•
	Submit			Back	
ECiass - 0 2015 - 2016					About Contact

Fig. 2: Admininstrator - Create new course

Edit and Delete Courses: Changes need to be made from time to time. A course might need to have some attribute altered, or even be deleted. For that reason,

administators have the option to edit or delete each course.

ECLA	ASS	ŀ	iome Help Manage Users	Manage Courses Log out
Ma	nage Courses			
Code	Name	Lecturer	Students Attending	
C01	Introduction to Programming	Jakubowski Michale	10 Ed	t Destroy
C02	Introduction to Programming 2	Jakubowski Michale	17 Ed	t Destroy
C03	Introduction to Programming 3	Little Hiram	9 Ed	t Destroy
C04	Comptus temporibus quae	Orn Tristin	13 Ed	t Destroy
C05	Praesentium calco	Shanahan Hassie	19 Ed	t Destroy
C06	Est vitium	Roberts Whitney	16 Ed	t Destroy
C07	Trucido caelum ara vallum	Conroy Nikki	12 Ed	Destroy
C08	Summa theca cavus xiphias	Bashirian Ethelyn	12 Ed	t Destroy
C09	Defigo venustas patria acquiro	Shanahan Hassie	16 Ed	t Destroy
C10	Corroboro certus	Little Hiram	13 Ed	t Destroy
		- Previous 1 2 3 Next -		
		New Course		
6				

Fig. 3: Administrator - Course index

Ē	ECLASS	Home Help Manage Users Manage Courses Log out	
I	Edit Cour	Se	
	Code	C01	
	Name	Introduction to Programming	
EClass - 0.2015 - 2016	Professor	Jakobowski Michałe Bauhlara Cłosłym, Comyn Naki Echmann Forencia Green Krata Hagenes Dents Hagenes Dents Hagenes Dents Michara Lacos ante Anterna Lacos ante Michara Ante Michara Lacos ante Michara Ante Michar	About Contact
		Little Harkm Om Trottin Redly Variace by Solamer Regun Scharter Regun Scharter Regun	

Fig. 4: Admininstrator - Edit course

2.2 Professors

Professors have a variety of tools available in order to manage their courses. These include:

Edit Course Description: Access to an index page containing only the courses they manage. Moreover, when viewing the main course index, a visual cue is in place to indicate their courses.

	ECLASS		Home Help	All Courses My Courses	Edit Profile Log ou	t
	My Co	Durses				
	Code	Name			Students Attending	
	C01	Introduction to Programming			9	
	C02	Introduction to Programming 2			12	
	C04	Curatio cernuus			10	
	C06	Cruciamentum abeo			7	
EClass - 0 2015 - 2016						About 0

Fig. 5: Professor - Personalized courses

	ECLAS	s		Home Help	All Courses My	r Courses	Edit Profile Log out
	View	/ Courses					
	Code	Name	Lecturer		Students	Attending	
	C01	Introduction to Programming	Auer Reagan			9	Manage
	C02	Introduction to Programming 2	Auer Reagan			12	Manage
	C03	Introduction to Programming 3	Schuster Emmanuelle			6	
	C04	Curatio cernuus	Auer Reagan			10	Manage
	C05	Vado ipsa comes	Lubowitz Melyssa			6	
	C06	Cruciamentum abeo	Auer Reagan			7	Manage
	C07	Corroboro crepusculum hic cruentus	Bayer Cristal			5	
	C08	Nihil tristis	Haag Marco			8	
	C09	Amoveo consequatur sopor anser	Steuber Abner			7	
	C10	Vultuosus iste thorax	Green Alize			11	
			Previous 🚺 2 3 N	ext →			
Class - 0 2015 - 2016							

Fig. 6: Professor - Course index

Edit Course Description: The ability to edit the description of the courses they manage, to better reflect the objective of each course. They can accomplish that by clicking on the description while viewing the course.

Developing an Electronic Classroom Platform with Ruby on Rails

	ECLASS	Home Help	All Courses My Courses Edit Profile Log out	
	Kent To Courses			
	C01 - Introduction to Pro	gramming		
	Auer Reagan			
			Show Attending Students	
	Description	Announcements	Assignments	
	Description			
	Fugial nam quae rerum cupiditale totam quia quos. El el 10 el alguane eanum. Tempore sed quaen escum descruz parta Cupiditate libro logital est minima. Laborrum qui rerum. Ibés Esse in est qui autem aut dotor. Veniam dotore rerum vell g	laudantium quam dotores soluta. Qui nai excepturi doto tea rihii sa gui et utam et impedit. sa guis sed nihii. Rerum nesciuni dotorem commodi labo Sulomit	em recutandae josam delenili repetat. Id voluptate nesciunt re molestiae	
		Cancel		
© 2015 - 2016				About 0

Fig. 7: Professor - Edit course description

Create New Announcements: Adding a new announcement for the students can be done seamlessly, directly from the course page.

ECLASS	Home Help	All Courses My Courses Edit Profile Log out
CO1 Intraduction to D	regregation	
	rogramming	
, doi noogan		Charac Minudian Charlente
		Snow Attending Sudemis
Description	Announcements	Assignments
Announcements		
Tal		
The		
Message		
	Submit	
	Cancel	
Dignissimos atque adipisci quis quae velt.		03/09/2016
Ad consectetur fugiat eum est recusandae.		03/09/2016
Voluplatem quae quam quas ipsa ad itaque neque.		03/09/2016
Aut reiciendis doloribus ducimus in dicta.		03/08/2016
Quam similique est laudantium		03/09/2016

Fig. 8: Professor - Create course announcement

Edit Course Announcements: Announcements can be edited, just like descriptions. All it takes is for the course professor to click on the announcement title or message, and they can edit it in place.

Developing an Electronic Classroom Platform with Ruby on Rails

E	CLASS	Home Help	All Courses My Courses Edit Pro	file Log out
< C A	Back To Courses 201 - Introduction to Programming uer Reagan			
			Show Attending Students	
	Description Announce	ments	Assignments	
	Announcements			
	New Anno	uncement		
	Dignissimos alque adipisci quis quae veit. Submit		Cancel	03/09/2016
	Ipsum ad placeat mollitia sapiente quas sed et. Dolorum ullam consectetur. Officia ma sed minima officia ab. Est qua impedit. Rerum eaque officia quia commodi officia acc. ut. A sunt vetit. Non optio et aut numquam tenetur quo.	ime non id moliita dolorem ve samus. Sit numquam nulla ea	elit. Vitae ea labore magnam sint qui. Enim a aut dolorem ipsa. Inventore minima deler	accusamus iti ut ut sunt
	Ad consectetur fugiat eum est recusandae.			03/09/2016
	Dolores nihil dignissimos repudiandae el. Voluptate eius sunt velit labore. Voluptatem r sequi autem at ullam asperiores. Optio at distinctio accusamus fugiat. Sed porro nobis corrupti reiciendis veniam nisi. Velit quasi fugiat. Sint non eum earum consequuntur de	epellendus eos. Quis et fugit a voluptatem accusamus moles ieniti. Vel autem quo repreher	amet id hic odit ducimus. Unde ducimus vo stiae. Ad nesciunt assumenda iste vitae. An iderit. Omnis aut eos.	luptatem sumenda et
	Malantana and a same and a same and			00.00.0040
	voupearem quae quam quas ipsa au saque hèque.			03/09/2016
	Aut reiciendis doloribus ducimus in dicta.			03/09/2016

Fig. 9: Professor - Edit course announcement title

ECLASS	Home Help	All Courses My Courses Edit Profile Log out
< Back To Courses C01 - Introduction to Pr Auer Reagan	rogramming	
		Show Attending Students
Description	Announcements	Assignments
Announcements	New Announcement	
Dignissimos atque adipisci quis quae velit.		03/09/2016
Ad consectetur fugiat eum est recusandae.		03/09/2016
Dolores n/bil dozistemos repudiandae et Volustate Unde ducinas volustatem sesui autem al utem aspe assemmenta de vitua. Assembla et conzel reisen Vett auxal logist. Sint non sum exum occreaturutur Gutomit Cancel	kus sunt vell labore. Voluptetem repetiendus eos. Quis et luga rerores. Quito al distincto accusamus lugas. Sed parco nobis y de avenan nai. delentil. Vel autem quo reprehendent. Omnia aut eos.	a engl M his odil duolmus. okuplatem accusamus molestisa. Ad nesclant
Voluptatem quae quam quas ipsa ad itaque neque.		03/09/2016
Aut reiciendis doloribus ducimus in dicta.		03/09/2016
Quam similique est laudantium.		03/09/2016

Fig. 10: Professor - Edit course announcement message

Create Course Assignment: Professors have the ability to create assignments for their students, and the choice to add accompanying documents, available to download, for assistance. Moreover, they set a deadline. Once the deadline is over, the students can no longer upload their homework.

	ECLASS	Home	Help All Courses	My Courses Edit Profile	Log out
	Content Course Content	ogramming			
	Title				
	Description				
	Due Late 2016 : March : 9 : Document Name Document Name Document Name	Browse No file selected. Browse No file selected. Browse No file selected.		Remove Document Remove Document Remove Document	
		Create Assignment Back			
EClass - 0 2015 - 2016					

Fig. 11: Professor - Create course assignment

Edit Course Assignment: Assignments can be edited in two ways. The description can be edited in place in the assignment page, and documents can be added or deleted. There is a dedicated edit page in order to change the title and deadline of the assignment.

	ECLASS	Home Help All Courses My Courses Edit Profile Log out	
	KBack To Introduction to Programming Introduction to Programming - Quaer quibusdam voluptatem sequi invento Due Date: 03 - 07 - 2016 (Expired)	at aut et minus Cello Cello	
		No Submitted Homeworks	
	Description		
	Dono envi al consendator el Mandillis. Intentos entina en. Datre soluzionen el repetito Doctinas veluentaria dirguisettos. Entendi directos procesanos el fortes reconcidentaria Magri diguisettos e so por secondo desencant impedit discutaria. Un intentos acutileos la veluentaria de veluenta intentores molestas espísitado harum. Eum inventos acutileos harum aquerest. Oficia sur la fortere soluta porterendo sisterum. Adipital recurandas está quas qui. Adi en nu maximo.	due opio ex. El trum aut anizin en ode eco. You necessitativos entros excentario o el de consequente modure. Cum volutates debite sos deloremase mbil reguerm que sos. Morá quiegaum similaços desenut montrum yo. Republid qui aut	
	Submit Cancel		
	Documents		
	First Document Second Document	View Download Remove View Download Remove	
	Add new Doc	ment	
EClass - 0 2015 - 2016			About Contact

Fig. 12: Professor - Edit course assignment - Description only

ECLASS	Home Help All Courses My Courses Edit Profile	Log out
< Back To Cours C01 - Ini Edit Assignm	。 troduction to Programming ent	
Title	Quaerat aut et minus quibusdam voluptatem sequi inventore.	
Description	Ouos error ut consectetur et blanditis. Inventore veniam eum. Dotore voluptatem et repellendus. Ducimus voluptatibus digniseimos. Tenetur distinctio accusamus nihi facere reprehenderit culpa ex. El rerum aut ad odt eos. Vel necesstatibus comis excepturi. Magni dignissimos ea que orieciendi desenum impedi taliquari. Ut voluptatibus dolor voluptas inventore molestias explicabo harum. Eum inventore architecto ut qui consequatur incidunt. Cum voluptates debites codocremça nihi harum quaerat. Officia une facere soluta perferendis laborum. Adpisci recusandas eint quas qui. Aut ea numquam quo sed. Mod quiaquam similipue desentin notarun. It Repellat qui ut maximo.	animi
Due Date	[2016 ::] [March ::] [7 ::]	
	Edit Assignment	
	Back	
FCiass - 0.2015 - 2016		Abuit Contert
		Putoti Contact

Fig. 13: Professor - Edit course assignment

E	CLASS	Home Help	All Courses My Courses	Edit Profile Log out	
< اr D	Back To Introduction to Programming Introduction to Programmi Dosum corporis nulla qua Uue Date: 05 - 05 - 2016 (1 month, 26 -	ning - Sequi nesciunt le illum dolorum. days, 7 minutes, and 18 seconds)	perspiciatis	Edit Delete Homeworks	
	Description				
	Neque consequentur eum ul molestiae explicabo. Ne quis corporti avino. Corporte afficia consequentur quis molestias, inventorr Dolorem minima non totam. Eum diprisamos taborum Apportores allas setti eti 10 add. Elgendi explicabo sequi delenti. Excepturi placeat at	sclunt facilis qui voluptatem exercitationem et. Deten est aperiam aut tenetur. Ut deserunt possimus pro- e earum est quaerat. Enim perspiciatis in molestiae o qui eos. Sunt magni quo voluptates recusandae.	iti alque et debitis. Maxime est ident quaerat. onsequatur cupiditate. Fugiat t	placeat repellendus ienetur aliquam.	
		NUMBER NO INCOMPOSE	Add Locument	Canton	
6.000 - 9,013 - 2013					About Contact

Fig. 14: Professor - add and delete documents

View and Download Homeworks: Professors have access to a page where they can download homeworks submitted by students. As shown in 12 and 14, the link is only enabled if at least a homework has been submitted.

	ECLASS		Home Help All Courses		
	< Back To Sequi n Sequi ne illum dolc Homework Su	esciunt perspiciatis ipsum corports nulla q SCIUNT perspiciatis ips rrum. bmissions	use illum dolorum. sum corporis nulla quae		
	Student ID	Student email	Student Name		
	2	sdi0600296@di.uoa.gr	Terry Franco	Download	
EClass - @ 2015 - 2016					About Contact

Fig. 15: Professor - View homeworks

View Attending Students: Lastly, professors can have an overview of all users attending each of their courses, their names and emails.

	ECLASS		Home Help All Courses My Courses Edit Profile I	Log out
	< Back To In	troduction to Programming		
	Introd	uction to Programming - Stu	dents	
	ID	Name	Email	
	2	Terry Franco	sdi0600296@di.uoa.gr	
	4	Kilback Claire	sdi0600297@di.uoa.gr	
	7	Wilms Dayana	sdi0200151@di.uoa.gr	
	12	Dooley Ted	sdi0700154@di.uoa.gr	
	16	West Maida	sdi0600156@di.uoa.gr	
	20	Gusikowski Stacy	sdi0200157@di.uoa.gr	
	22	Langosh Antonina	sdi0500158@di.uoa.gr	
	30	Daugherty Emory	sdi0500160@di.uoa.gr	
	36	Effertz Wiley	sdi0100163@di.uoa.gr	
Class - 0 2015 - 2016				About Conta

Fig. 16: Professor - students attending course

2.3 Undergraduate Students

Students differ from the other two categories, in the sense that they are mostly recepients of information instead of providers. However, they do enjoy unique features, like:

An Index of Courses they Attend: Besides the index of all available courses, shared by all roles, students also have a personalised page which includes only the courses they are enrolled.

	Code	Name	Lecturer		
	C01	Introduction to Programming	Auer Reagan	Withdraw	
	C02	Introduction to Programming 2	Auer Reagan	Withdraw	
	C03	Introduction to Programming 3	Schuster Emmanuelle	Withdraw	
	C09	Amoveo consequatur sopor anser	Steuber Abner	Withdraw	
	C10	Vultuosus iste thorax	Green Alize	Withdraw	
	C11	Eligendi utrimque nisi stella	Hintz Cathrine	Withdraw	
	C12	Tardus cubo sumptus	Kohler Lemuel	Withdraw	
	C13	Thymum tergo	Schuster Emmanuelle	Withdraw	
	C16	Damnatio eaque	Boyer Estelle	Withdraw	
	C17	Vulgo tamdiu ducimus	Hintz Cathrine	Withdraw	
			- Previous 1 2 Next -		
6 (51)15 (10)16					About 0
					About C

Fig. 17: Student - My courses

Quick Enrolling and Withdrawing: Students have the ability to easily enroll to and withdraw from courses, from every page associated with them. They can accomplish that in the course index page, the personalised course page 17, as well as each individual course page. 20

	ECLAS	s	Home	Help All Courses My Courses	Edit Profile Log out
	All C	ourses			
	Code	Name	Lecturer	Students Attending	
	C01	Introduction to Programming	Auer Reagan	ş	Withdraw
	C02	Introduction to Programming 2	Auer Reagan	15	Withdraw
	C03	Introduction to Programming 3	Schuster Emmanuelle	6	Withdraw
	C04	Curatio cernuus	Auer Reagan	10	Enoril
	C05	Vado ipsa comes	Lubowitz Melyssa	6	Enoril
	C06	Cruciamentum abeo	Auer Reagan	1	Enoril
	C07	Corroboro crepusculum hic cruentus	Bayer Cristal	5	Enoril
	C08	Nhil tristis	Haag Marco	٤	Enoril
	C09	Amoveo consequatur sopor anser	Steuber Abner	1	Withdraw
	C10	Vultuosus iste thorax	Green Alize	11	Withdraw
			- Previous (1) 2 3 Next -		
EClass - © 2015 - 2016					

Fig. 18: Student - Courses

Developing an Electronic Classroom Platform with Ruby on Rails

	ECLASS	Home Help	All Courses My Courses Edit Profile Log out	
	CO4 - Curatio cernuus Auer Reagan			
			Enori	
	Description	Announcements	Assignments	
	Assignments			
	Quod aspernatur voluptatem aut et quidem.	Due Date: 04/19/2016	5 (1 month, 10 days, 15 hours, 26 minutes, and 17 seconds)	
	Enim eligendi cumque sit placeat quaerat itaque.	Due Date: 0	3/22/2016 (13 days, 15 hours, 26 minutes, and 17 seconds)	
	Est dolorem quia provident eveniet.	Due Date: 0	3/23/2016 (14 days, 15 hours, 26 minutes, and 17 seconds)	
C1863 - 0 2013 - 2018				About

Fig. 19: Student - Enroll from individual course

Visual Assignment Deadline Overview: Students can check their assignment deadlines at a glance, both from the assignment index in the course page, and the assignment page itself.

E	CLASS	Home Help	All Courses My Courses Edit Profile Log out	
(((^{(Back To Courses} C01 - Introduction to Programm Wer Reagan	iing		
	Description	Annuncomente	Withdraw	
	Assignments	Announcements	Розијшнено	
	Quaerat aut et minus quibusdam voluptatem sequi inventore.		Due Date: 03/07/2016 (Expired)	
	Sequi nesciunt perspiciatis ipsum corporis nulla quae illum dolorum.		Due Date: 01/01/2016 (Expired)	
	Temporibus tempora numquam modi quis id consequantur sapiente.		Due Date: 03/07/2016 (Expired)	
	Et non odit repudiandae.	Due Date	e: 03/29/2016 (20 days, 15 hours, 26 minutes, and 4 seconds)	
	Et quibusdam esse inventore vero quia excepturi modi.	Due Date	e: 04/01/2016 (23 days, 15 hours, 26 minutes, and 4 seconds)	
ECiass - © 2015 - 2016				About Contact

Fig. 20: Student - Course assignments overview

E	ECLASS	Home Help All Courses My Courses Edit Profile Log out	
، ۱ ۲	Clack To Introduction to Programming Introduction to Programming - Sequi ne perspiciatis ipsum corporis nulla quae i Due Date: 01 - 01 - 2016 (Expired)	esciunt Ilum dolorum.	
	Description		
	Neque consequentur eum et molessiae explicado. Nescium facilis qui volqutatem exercitat corporis vero. Corporis officia consequator quis molessias. Inventore est apenam aut tenetur. Ut deserun Dolerem minima non totam. Eura digrassimos laborum exam est queerat. Enim perspicial Aperrores alas sia: et ello sort. Eligend explicado sequi delenti. Escepturi placeat at qui eco. Sunt magri quo volquipales	ionem et. Delenti atque et debits. Maxime est placeat repellendus quis t possimus provident quaenat. Is in molestae consequatur cupiditate. Fugiat tenetur alquam. recuandue.	
	Your homework		
	The assignment has expired, and you have submitted: todo.txt	Download	
EClans - 0 2015 - 2016			About Contac

Fig. 21: Student - View assignment

Homework Submission and Download: The interface is different depending on whether the homework is submitted or not. A student is able to upload a different file, overwriting the original, as well as download the one he has uploaded. Once the deadline for the assignment is over, the student no longer has the ability to upload files 24, but is still able to download a file he has submitted 21.

E	ECLASS	Home Help All Courses My Courses Edit Profile Log out	
 	K Back To Introduction to Programming Introduction to Programming - Et non c Due Date: 03 - 29 - 2016 (20 days, 15 hours, 25 minutes, and 3	dit repudiandae. 5 seconds)	
	Description Josom at impedit aut welt debits voluptatem architecto. Eos ut dólores, impedit quía quibus cocancad elus et. Quasi langar reiciendis et quo alías eos. Et sunt aliquam. Nihi dólores quasi inventore tem Alas esumer eras ciutas. Similga en alía quam. Com nomitina tutas. Defeniti rerum autem reiciendis. Aut et holls e dólores, in eiux eos non architecto animi.	dam accusardum debitis rerum et expedita. Quos officia facilis qui pora.	
	Submit your Assingment Choose File No file chosen Submit		
EClass - © 2015 - 2016			About Contact

Fig. 22: Student - Assignment not expired, homework not submitted

Developing an Electronic Classroom Platform with Ruby on Rails

E	ECLASS	Home Help All Courses My Cour	rses Edit Profile Log out	
 	Back To Introduction to Programming Introduction to Programming Derspiciatis ipsum corpo Due Date: 05 - 05 - 2016 (1 month, 26 c	ning - Sequi nesciunt ris nulla quae illum dolorum. Iays, 15 hours, 24 minutes, and 31 seconds)		
	Description Neque consequentur eum ut molestae explicabo. Nes corporis vero. Corporis officia consequatur quis molestas. Inventore Dolorem minima non totam. Eum diguestinos laborum Asperiores allas sint est IIIo sed. Eligend explicabo sequi delenti. Excepturi placeat at o	ciunt facilis qui voluptatem exercitationem et. Deleniti atque et debitis. Maxime e est apeniam aut tenetur. Ur deserunt possimus provident quaerat. earum est quaerat. Enim perspiciatis in molestiae consequatur cupiditate. Fugi qui eos. Sunt magri quo voluptates recusandae.	est placeat repellendus quis at tenetur aliquam.	
	Submit your Assingment Choose File No file chosen Submit	You've already submitted this on 03/09/2018, 00:46. Submitting it again will overwrite the previous submission.	Download	
EClass - © 2015 - 2016				About Contact

Fig. 23: Student - Assignment not expired, homework submitted

View and Download Assisting Files: Students are also able to view and download files associated with assignments through the assignment page.

E	CLASS	Home Help All Courses My Courses	Edit Profile Log out	
i C	Back To Introduction to Programming ntroduction to Programming - Quaera quibusdam voluptatem sequi inventor due Date: 03 - 07 - 2016 (Expired)	at aut et minus e.		
	Description			
	Quos error ut consectetur et blandtils, inventore veniam eum. Dotore voluptatem et reg Ducimus voluptatibut dignisismons, Teinetta distinctio accusamus mihi facer reprehend exceptut. Margi dispismons es quo reichtio dissenut mitogendi allavam. Ut voluptatibuto didur voluptas inventore molestias explicabo hanm. Eum inventore arci nihi harum quaenti. Oficia aure facer soluta perferendis laborum. Adiplicci recusandae sint quas qui, Aut en Repelat qui aut maxime.	ettendus. Int culpa ex. Et rerum aut animi ad odt eos. Vel neces ittecto ut qui consequatur incidurt. Cum voluptates de numquam quo sed. Modi quisquam similique deserur	ssitatibus omnis abitis eos doloremque nt nostrum ut.	
	Your homework			
	The assignment has expired, and you have not submitted your work.			
	Documents			
	First Document Second Document		Download Download	
EClass - © 2015 - 2016				About Contact

Fig. 24: Student - Document download

2.4 Shared Pages

Pages demonstrated above are similar, but offer different features according to the role of the user visiting them. Obviously, in some cases this is not needed, so pages like the log-in screen or the sign-up page are shared.

	ECLASS		Home Help Sign in Sign u	p
	Log in			
	Email			
	Password			
	Remember me?			
	Log in	Forgot your password?	Sign up	
EClass - © 2015 - 2016				About Contact

Fig. 25: Log-in page

	ECLASS	Home Help Sign in Sign up	
	Sign Up		
	Name		
	Surname		
	Email		
	Password (6 characters minimum)		
	Password Confirmation		
	Sign up	Log in	
EClass - © 2015 - 2016			About Contact

Fig. 26: Sign-up page

3. DEVELOPING THE APPLICATION

Following the presentation of the application and its features, we now analyse the process of developing it. By explaining how it came to be, a closer look is taken at RoR and what it has to offer. RoR is by design a MVC framework, therefore the next logical step is to examine the models, views and controllers, as well as the techniques, tools and gems used to tie it all together.

3.1 The Models

RoR uses Active Record(AR) as an Object Relational Mapping(ORM) framework.[5] This, in association with CoC, do most of the configuration automatically. Manual configuration only needs to be done when the standard convention cannot be followed. A closer look at the models of the application shall accentuate this point. Models that do not offer functionality not already discussed will be omitted.

3.1.1 User

```
class User < ActiveRecord::Base
rolify
devise :database_authenticatable, :registerable,
    :recoverable, :rememberable, :trackable, :validatable
validates :name, :surname, presence: true
after_create :assign_default_role
has_many :homeworks, inverse_of: :user
has_many :courses_teaching, :class_name => :Course, inverse_of: :lecturer,
    ..., :foreign_key => "lecturer_id"
has_many :student_attends_courses, inverse_of: :student, :foreign_key => "user_id"
has_many :courses_attending, :class_name => :Course, through:
    ..., :student_attends_courses, :foreign_key => "user_id"
def assign_default_role
add_role(:undergrad)
end
```

```
def is_master_acc?
    self.id == User.first.id
end
```

end

Listing 1: The User Model

This is the file for the User model. Dissecting it line by line provides a better understanding.

class User < ActiveRecord::Base

This line contains the name of the model and shows the inheritance of the AR base.

• rolify

devise :database_authenticatable, :registerable,

:recoverable, :rememberable, :trackable, :validatable

There are functions provided by gems, rolify [8] and devise[9], and will be discussed in detail later on.

• validates :name, :surname, presence: true

With this simple function, it is ensured that every User record will contain a name and a surname. E-mail and password are not present, since they are handled by devise[9].

• after_create :assign_default_role

after_create sets callback functions to be executed on the object, immediately after they are created. In this case, a role is assigned to the user as soon as they sign up.

has_many :homeworks, inverse_of: :user

By using has_many, model relationships are defined, specifically a one-to-many relationship. inverse_of: is optional, but quite powerful. It offers bi-directional access to models, omitting the need for an SQL query to do so.

has_many :courses_teaching, :class_name => :Course, inverse_of: :lecturer,
 ... :foreign_key => "lecturer_id"

This line of code is similar to the previous one. However, it offers 2 additional parameters. Since the first argument of has_many is not the name of a model, the model is provided by using : class_name. This enables us to use an alias, which accomplishes 2 goals. Firstly, it allows us to have more than one relationships between two models. In this case, the course has to belong to a single user, its lecturer, but also have many users, the students who attend it. Additionally, it increases code readability, since course.lecturer is more intuitive than course.user. The last argument, : foreign_key, specifies the column in the corresponding table, in this case the courses table, which includes the key to the user model.

has_many :student_attends_courses, inverse_of: :student, :foreign_key =>
 "user id"

has_many :courses_attending, :class_name => :Course, through:

→ :student_attends_courses, :foreign_key => "user_id"

Further highlighting the benefit of setting an alias, a second relationship between users and courses is set. This time it is a many-to-many relationship. Usually, this is accomplished by using has_and_belongs_to_many. Since it is not the only relationship between the two tables, a third table needs to be introduced, by using the through : argument. In order to increase the readability of the code, an alias is again used. By adding a slight complexity, merely an extra line, two different kinds of relationships can be set between the models, with an added bonus of "natural" code readability.

```
    def assign_default_role
add_role(:undergrad)
end
```

```
def is_master_acc?
  self.id == User.first.id
end
```

Two methods are defined for the User model. assign_default_role is the method called after a new instance of the model is created. All users are initially set to be students, to prevent unexpected behaviour in case a role is not assigned. The other function, is_master_acc? is used to verify that an administrator account will always exist, as discussed earlier. 1

3.1.2 Course

```
class Course < ActiveRecord::Base
belongs_to :lecturer, :class_name => :User, inverse_of: :courses_teaching
```

has_many :announcements, inverse_of: :course, :dependent => :destroy has_many :assignments, inverse_of: :course, :dependent => :destroy

validates :code, :name, :lecturer, presence: true end

Listing 2: The Course Model

The course model. Examing it fills up the blanks left from the user model 1 and gives us an overall understanding of relationships between models.

• belongs_to :lecturer, :class_name => :User, inverse_of: :courses_teaching

belongs_to compliments the has_many used by the user model. belongs_to

is ambiguous, in the sense it is not clear whether it is a one-to-many or one-toone relationship. When inverse_of: is set, the relationship is indicated by its value. A singural word means it is one-to-one, and plural is one-to-many. However, inverse_of: is optional, so when it is not user, the relationship is determined by the corresponding model — has_many means one-to-many, whereas has_one is used for one-to-one. Regardless of the kind of the relationship, belongs_to means that this model will hold the id of the object it belongs to. By conversion, this column is named model_name_id, but since an alias is used, :class_name is set, and :foreign_key is set to the corresponding model.

has_many :student_attends_courses, inverse_of: :course, :foreign_key =>
 "course id"

```
has_many :students, :class_name => :User, through: :student_attends_courses,
```

→ :foreign_key => "course_id"

These expressions mirror the ones in the user model 1.

 has_many :announcements, inverse_of: :course, :dependent => :destroy has_many :assignments, inverse_of: :course, :dependent => :destroy

Through the above associations, it is clear that the course model has one-to-many associations with the assignment and announcement models. It is safe to assume that both of their models include belongs_to :course, inverse_of: in them and a course_id column in their tables.

The parameter dependent => : destroy makes sure than once a course is deleted, all announcements and assignments associated with it will be deleted as well.

• validates :code, :name, :lecturer, presence: true

The validation has one noticable difference to the one found in User 1. It validates the presence of : lecturer, although the attribute is named : lecturer_id. This is actually crucial, because not only it validates that the field is not empty, but also that the id belongs to an existing record.

3.1.3 StudentAttendsCourse

Listing 3: The StudentAttendsCourse Model

The missing link to the many-to-many relationship between courses and the students who attend them. This model belongs to both a student and a course, and through it the two other models are connected. : class_name, : inverse_of and : foreign_key are set according to the values mentioned before. 1 2

3.1.4 Announcement

```
class Announcement < ActiveRecord::Base
belongs_to :course, inverse_of: :announcements
validates :title, :message, :course, presence: true
default_scope { order('updated_at DESC') }
end
```

Listing 4: The Announcement Model

The announcement model. The last function in the model, default_scope[10]. This method takes a block of code as an argument and adds a scope for all operations in the model. In this example, it is used in conjuction with order ('updated_at_DESC'), which orders announcements in descending order, according to their last update, thus ensuring that the most recently updated announcements always appear first.

3.1.5 Assignment

class Assignment < ActiveRecord::Base</pre>

belongs_to :course, inverse_of: :assignments
has_many :documents, inverse_of: :assignment, :dependent => :destroy
has_many :homeworks, inverse_of: :assignment, :dependent => :destroy
accepts_nested_attributes_for :documents
validates :title, :description, :due_date, :course, presence: true
validates_associated :documents
end

Listing 5: The Assignment Model

The assignment model contains a set of methods related to models that belong to it.

accepts_nested_attributes_for :documents

This function enables the management of the documents model through its parent assignment. This is extremely useful because it allows the addition of documents to the assignment directly when creating it, as well as updating documents through the assignment.

validates_associated :documents

This method acts like validates does, but for the associated object. It completes the functionality of accepts_nested_attributes_for, by validating the attributes of the associated record.

3.1.6 Document

class Document < ActiveRecord::Base

belongs_to :assignment, inverse_of: :documents mount_uploader :doc, DocumentUploader validates :name, :doc, :assignment, presence: **true**

end

In the document uploader, the mount_uploader appears. This method is provided by the carrierwave gem.[11]

3.2 The Controllers

Controllers are responsible for transferring data between the end user and the application. The routing from each request to the appropriate controller is done in a single file, "routes.rb". Controllers in RoR prepare the information and make it available to for the views, and ara capable on responding differently to different kinds of requests. Below, selected bits and pieces from controllers are presented. Due to the CoC nature of RoR, a significant part of the functionality is identical among controllers, so it makes little sense to showcase them all.

3.2.1 The Course Controller

• class CoursesController < ApplicationController

All controllers in RoR inherit the ApplicationController[6], which in turn inherits ActionController[7]. This allows for a centalised class to configure application security.

• before_action :set_course, only: [:show, :edit, :update, :destroy, :description,

```
\rightarrow :attending_students]
```

Like in the models, **before_action** sets methods to be called before the actual method is called. In this example, the method **set_course** is executed exclusively — unless called directly of course — before the actions between the brackets.

```
def set_course
  params[:id] = params[:course_id] if params[:id].nil?
  @course = Course.find(params[:id])
end
```

This conforms with the DRY nature of RoR, since instead of having the code above, or a call of the method, separately in all the methods in the bracket, the methods requiring it are neatly congregated in the before_action.

def create
 @course = Course.new(course_params)

 respond to do |format|

```
if @course.save
  format.html { redirect_to @course, notice: 'Course was successfully created.'
  }
  format.json { render :show, status: :created, location: @course }
  else
   get_professors
  format.html { render :new }
  format.json { render json: @course.errors, status: :unprocessable_entity }
  end
  end
  end
```

Examining the method to create a new course, leads to interesting findings. Right away, another method, called course_params is passed as an argument in the Course.new method.

```
def course_params
```

```
params.require(:course).permit(:code, :name, :lecturer_id, :description)
end
```

The course_params sanitizes the parameters passed by the request, and only allows the proper ones to go through. The next step is to attempt to save the newly created course. Depending on the outcome, one of two might happen. Either the course is successfully saved, and the controller responds with the corresponding format of the request, or it fails and the appropriate action is taken. In this specific case, the function get_professors is called, and then a proper response, depending on the format is given.

```
def get_professors
```

```
professors = User.with_role :professor
@professors = []
professors.each do |a|
@professors << ["#{a.surname} #{a.name}", a.id]
end
@professors.sort! { |a, b| a[0] <=> b[0]}
end
```

The get_professors method is used to create an array which contains information about all available professors. That array is used in the drop-down menu of the course form, as displayed earlier. 4

```
    def create
        @course = Course.new(course_params)
        respond_to do |format|
        if @course.save
            format.html { redirect_to @course, notice: 'Course was successfully created.'
        }
        format.json { render :show, status: :created, location: @course }
        else
            get_professors
            format.html { render :new }
            format.json { render json: @course.errors, status: :unprocessable_entity }
        end
        en
```

This method is a good example of the aliases discussed in the models section 1. The is_professor? and is_undergrad? methods are provided by rolify[8]. Depending on the type of user making the request, the controller is able to get either the courses the professor is teacher, or the ones the undergradute student attends. Then, the paginate method, provided by the will_paginate gem[12], takes care of the pagination in the view page, as showcased before 5 17. render method renders the desired view. By conversion, Rails renders the view that shares the name with the controller. In case this is not the desired effect, it has to be instructed to do otherwise.

3.2.2 The Assignment Controller

```
def assignment_params
```

params.require(:assignment).permit(:title, :description, :due_date, :course_id,

```
→ documents_attributes: [:name, :doc, :_destroy])
```

end

The assignment_params method is interesting, since it has already been established that assignment may include documents when they are created 11.

The documents_attributes array and its contents is what permits the attributes of the documents to pass the sanitazation process, and be created.

3.3 The Views and Helpers

According to the MVC paradigm, views are used to present the information to the user. However, in order to not break the paradigm, no logic should be implemented in the views. To avoid convoluted code in views and controllers, or a version of the view for every possible outcome, RoR uses helpers and partials. Helpers may contain logic and are called from the view, maintaining the MVC pattern and avoiding that obstacle. Views mainly consist of the following parts, glued together by the helpers.

3.3.1 The Views

- **HTML Pages:** Html pages are the main component of the views. However, they hardly ever are plain html pages. RoR by default uses Embedded Ruby(ERB) pages. Other options include HTML Abstraction Markup Language(HAML) [13] and Slim [14]. This project uses ERB files, with the exception of the assignment form 11, which utilizes the Cocoon gem [15]. The gem example was in HAML, so the form was created in it as well for variety's sake.
- **Partial Pages:** As the name implies, those are not whole pages, but html snippets, fitting like pieces of a puzzle where they are required. Besides not being able to stand alone, partials are the same as the HTML pages described above. Their name is required to start with an underscore(_).

```
<span style="color: <%= color %>">
<%= time_left %>
</span>
```

JavaScript Files: Not to be confused with JavaScript files used holding the usual JavaScript functions — these are found in the assets/javascripts folder, and often written in CoffeeScript [20]. The JavaScript files associated with views are mostly rendered as a response, made by the controller, to an Asynchronous JavaScript and XML(AJAX) request. JavaScript files can also contain ERB.

3.3.2 The Helpers

Helpers are ruby methods, used to implement logic for the views. Helpers might return a result to the view, render a partial, maybe even do nothing. The following snippet is used to determine whether it is the course professor viewing the assignment, so the edit and delete buttons should be displayed, and render that partial if that is true.

```
def edit_and_delete_button(course, id)
  render partial: "edit_and_delete_assignment" if is_course_professor?(course, id)
end
```

3.4 The Gems

Gems, Ruby's open-source, community-backed libraries, are extensively used in Rails. It is highly likely that a problem that occurs or a feature that needs to be implemented

already exist in a gem. Following is a sample of the gems used in this application.

Bootstrap-sass: Boostrap-sass provides a port of Bootstrap 3 for RoR. [16]

Devise: Devise offers out-of-the-box user authentication and management. [9]

Rolify: Rolify assists with role management and scoping. [8]

- **CarrierWave:** CarrierWave is a classier solution for file uploading. [11]
- Cocoon: Cocoon makes nested form handling seamless. [15]
- Will Paginate: Will Paginate offers pagination with a single method. [12]
- **Best in Place:** Best in Place offers a highly customizable option for editing records in place by using JSON. [17]
- **jQuery Turbolinks:** jQuery Turbolinks is used to address an issue to jQuery functions, caused by turbolinks. [18]
- **Faker:** Faker generates genuine-looking data. It is ideal for seeding the database tables, which is perfect for developing an application. [19]

3.5 How it all ties together - A Workflow

In the previous sections, each component of the application was analysed in detail. To fully understand how it all ties together, an example is in order. The example consists of a step-by-step examination of an administrator changes the role of a user. In the image below, the administrator is changing the role of the user from undergrad to professor.

	Log out
Manage Users	
ID Name Email Use	Iser Role
1 Sipes Daisha admin@di.uoa.gr Master Account	
2 Langworth Myrtice sd0600296@d.uxa.gr Undergrad •	
3 Jakubowski Michale professor/@di.uoa.gr Professor •	
4 Murray Evert sd0000297@di.uoa.gr Undergrad •	
5 Little Hiram professor2@di.uoa.gr Admin soor •	
6 Rolfson Hunter sdi0500151@di.uoa.gr Professor	
7 McKenzie Jean admin2@d.uoa.gr Admin →	
8 Murphy Okey sd0200152@di.uoa.gr Undergrad +	
9 Donnelly Lizy admin3@di uoa.gr Admin •	
10 Hansen Leone professor4@di.uoa.gr Professor+	
- Previous 🚯 2, 3, 4, 5 Next	

Fig. 27: Administrator changing the role of a user.

The HTML behind the "Professor" button

```
<form action="/users/4/give_role?rol=professor" accept-charset="UTF-8"

data-remote="true" method="post">

<input name="commit" value="Professor" class="btn btn-block btn-warning"

type="submit">

</form>
```

According to the routes file,

```
resources :users, only: [:index] do
post "give_role"
end
```

that path is linked to the users controller, in the give_role method. The controller contains two before_action methods.

```
before_action :set_user, only: [:give_role]
before_action :set_hashes
```

The set_hashes, which are instance variables, containing the three roles as well as the colors associated with each.

```
def set_hashes
  @colours = {
    "admin" => "danger",
    "professor" => "warning",
    "undergrad" => "info"
  }
  @drop_roles = [
    "admin",
    "professor",
    "undergrad"
]
end
```

The set_user method retrieves the user whose roles is about to change, and it removes the current role, unless it is the master account.

```
def set_user
@user = User.find(params[:user_id])
@user.roles = [] unless @user.is_master_acc?
@role = params[:rol]
end
```

It checks that by calling the is_master_acc? method from the user model.

```
def is_master_acc?
    self.id == User.first.id
end
```

Once both the **before_action** methods are called, the **give_role** is executed. It calls the **add_role** method provided by the rolify gem [8], again checking if it is the master account.

```
def give_role
  @user.add_role(@role) unless @user.is_master_acc?
  respond_to do |format|
   format.js
  end
end
```

The method then responds to the .js format. This renders the .js file with the same name as the controller method.

```
$('<%= "#user_#{@user.id}" %>').empty();
$('<%= "#user_#{@user.id}" %>').html("<%= escape_javascript(render :partial => 'role',

→ locals: {user: @user, role: @role}) %>")
```

In turn, that clears the div containing the dropdown menu, and renders the _role.html.erb partial,

which creates the drop down menu, and calls

<%= form_tag(user_give_role_path(user, :rol => role), remote: **true**) **do** %> <%= submit_tag "#{role.capitalize}", class: "btn btn-block btn-#{@colours[role]}"%> <% end %>

on each role besides the newly assigned one, to create the proper buttons.

4. IN CONCLUSION

In all crafting areas, it's easier to accomplish something if you have the tools of the trade. Developing web applications does not deviate from that course. This thesis aims to establish the Ruby on Rails framework as one of the best choices when it comes to web development, since it offers swift and agile development, exellent workflow and enjoys amazing community support. Through its DRY and CoC philosophies, it takes away all the chores from developing web applications, leaving programmers to focus on the fun parts.

ABBREVIATIONS AND ACRONYMS

MVC	Model-View-Controller
RoR	Ruby on Rails
DRY	Don't Repeat Yourself
CoC	Convention Over Configuration
ORM	Object Relational Mapping
AR	Active Record
ERB	Embedded Ruby
HAML	HTML Abstraction Markup Language
AJAX	Asynchronous JavaScript and XML

REFERENCES

- [1] "Ruby on Rails" Ruby on Rails [Online]. Available: http://rubyonrails.org. [Accessed: Mar 8, 2016]
- [2] "GitHub" GitHub [Online]. Available: https://github.com. [Accessed: Mar 8, 2016]
- [3] "Ruby Gems" Ruby Gems [Online]. Available: https://rubygems.org. [Accessed: Mar 8, 2016]
- [4] "The Ruby Toolbox" The Ruby Toolbox [Online]. Available: https://www.ruby-toolbox.com. [Accessed: Mar 8, 2016]
- [5] "Active Record Basics" Active Record Basics [Online]. Available: http://guides.rubyonrails.org/active_ record_basics.html [Accessed: Mar 8, 2016]
- [6] "Application Controller" *Application Controller* [Online]. Available: http://api.rubyonrails.org/classes/ ActionController/Base.html [Accessed: Mar 8, 2016]
- [7] "Action Controller Base" Action Controller Base [Online]. Available: http://api.rubyonrails.org/classes/ ActionController/Base.html [Accessed: Mar 8, 2016]
- [8] "Rolify" Role management library with resource scoping [Online]. Available: https://github.com/RolifyCommunity/rolify [Accessed: Mar 8, 2016]
- [9] "Devise" *Flexible authentication solution for Rails with Warden* [Online]. Available: https://github.com/ plataformatec/devise [Accessed: Mar 8, 2016]
- [10] "Active Record Scoping Default Methods" Active Record Scoping Default Methods [Online]. Available: http://api.rubyonrails.org/classes/ActiveRecord/Scoping/Default/ClassMethods.html [Accessed: Mar 9, 2016]
- [11] "CarrierWave" *Classier solution for file uploads for Rails, Sinatra and other Ruby web frameworks* [Online]. Available: https://github.com/carrierwaveuploader/carrierwave [Accessed: Mar 9, 2016]
- [12] "https://github.com/mislav/will_paginate" *Pagination library for Rails, Sinatra, Merb, DataMapper, and more* [Online]. Available: https://github.com/mislav/will_paginate [Accessed: Mar 9, 2016]
- [13] "HAML" *HTML Abstraction Markup Language* [Online]. Available: http://www.haml.info [Accessed: Mar 9, 2016]
- [14] "Slim" A fast, lightweight template engine for Ruby [Online]. Available: http://slim-lang.com [Accessed: Mar 9, 2016]
- [15] "Cocoon" *Dynamic nested forms using jQuery made easy* [Online]. Available: https://github.com/ nathanvda/cocoon [Accessed: Mar 9, 2016]
- [16] "Bootstrap-sass" Official Sass port of Bootstrap 2 and 3 [Online]. Available: https://github.com/twbs/ bootstrap-sass [Accessed: Mar 9, 2016]
- [17] "Best_in_place" A RESTful unobtrusive jQuery Inplace-Editor and a helper as a Rails Gem [Online]. Available: https://github.com/bernat/best_in_place [Accessed: Mar 9, 2016]
- [18] "jQuery Turbolinks" *jQuery plugin for drop-in fix binded events problem caused by Turbolinks* [Online]. Available: https://github.com/kossnocorp/jquery.turbolinks [Accessed: Mar 9, 2016]
- [19] "Faker" A library for generating fake data such as names, addresses, and phone numbers. [Online]. Available: https://github.com/stympy/faker [Accessed: Mar 9, 2016]
- [20] "CoffeeScript" CoffeeScript is a little language that compiles into JavaScript. [Online]. Available: https://http://coffeescript.org [Accessed: Mar 9, 2016]