FenixEdu Bible

Contents

1

Introduction

1.1 Project Overview

The FenixEdu project[1] started in 2002 at Técnico Lisboa[2], one of the top high education engineering schools in Portugal. FenixEdu's main goal was to develop an advanced academic Information System (IS) for Higher Education Institutions (HEI).

The architecture of the system was designed from the start to be fully web based, in order to provide wide availability and easy user interaction, independent of client software and operating systems, but assuring the high security standards, tight access control and log action control required by a critical information system.

The FenixEdu system is an integrated platform that works at all levels of the academic process. It implements a powerful academic Content Management System (CMS), which can be used at course, degree, department or institution level, an advanced Student Management System (SMS), a complete Learning Management System (LMS) and it also integrates all components required by a standard academic management platform. This includes management and support of all academic tasks, including on-line student applications and admission process, on-line enrolment and registration, evaluation and grade records, degree planning, design and aproval, full management of academic records, at graduation and post-graduation level, publication registration and archive of grades and student curricula, room, course and teacher scheduling and distribution, fee payment and control, quality control through on-line surveys and many other academic tasks. At the curricular unit (course) level, the system provides an information board and an easy to use CMS, planning and scheduling tools, publication of class summaries, publication of bibliographic references, documents and exercises, e-learning support and evaluation, student management and communication channels, RSS feeds, registration of the number of attendees and many other functionalities.

As stated before, the system provides also full support to all academic back office and management, including the workflow required by most academic processes. This includes design, planning, and

[1]The FenixEdu project code and documentation is available at `https://fenixedu.org`.

[2]Técnico Lisboa was formely known as Instituto Superior Técnico (IST)

approval of degree and courses (including contents, base bibliography, structure and planning), European Credit Transfer and Accumulation System (ECTS) validation, formal approval by the scientific board of courses, degrees and teaching staff, issue of diplomas and certificates, etc. The platform also handles thesis theme proposals, application and distribution, thesis registration, and associated academic tasks. The system includes also a complete curriculum management platform for teaching and research staff, enabling an overall view and balance of scientific production at individual, department and institutional levels.

The current version of the system encompasses a wide range of functionalities related to almost every component of the academic process. The complexity of the business model and of the underlying information system poses significant challenges from a software engineering perspective. As such, the system is often used both as a source of research problems and as a testbed for validating new research approaches that address the development of highly complex applications with rich domain models. One result of this tight integration between research and development was the introduction of an innovative software architecture for developing applications with persistent, transactional, rich domain models. This new architecture combines a Domain Specific Language (DSL) for implementing the structure of the domain model with a Software Transactional Memory (STM) to control concurrent accesses to the application's data. Both the DSL and the STM used in this new architecture were designed and developed from scratch to suit the needs of the FenixEdu project. However, since they are independent from the application layer, they were already extended to several other applications and adopted in different contexts. In this sense, the FenixEdu project turned out to be one of those rare cases of software systems that change the technological environment that surrounds them.

The FenixEdu academic platform is supported on an advanced identity management subsystem, which provides system wide user authentication to all computer and network services. The identity management backend is supported on Kerberos and LDAP (Lightweight Directory Access Protocol (LDAP), and provides web single sign-on based on Yale Central Authentication Service (CAS). The identity management subsystem also has full SAML 2.0 federation support based on OpenSAML, which provides full support for interoperability based on standard SAML (Security Assertion Markup Language) attributes, being fully compatible standard SAML based federations (namely Shibboleth based federations).

This identity management subsystem also supports strong authentication based on the Portuguese National Identity Card (named Cartão de Cidadão) and, moreover, is fully integrated with the European e-ID interoperability sub system, developed in the scope of the European project STORK. Digital signature of documents and actions is also possible using the Portuguese National Identity Card.

The overall system was developed from scratch as an open source project based on the LGPL (GNU Lesser General Public License), in order to offer a basis for open contributions, free distribution, free availability, and customization to special requirements and needs at different institutions. This distribution model enabled several higher education institutions to implement the FenixEdu system with the support of independent private companies and, at the same time, it enabled to enlarge the technical support of the platform, which does not rely today only on internal IST resources, given the wider implementation basis and technical support. This strategy contributed also to the long-term project sustainability.

Figure 1.1: Técnico Lisboa

1.2 Técnico Lisboa Overview

As stated before, the FenixEdu project was developed from scratch at Técnico Lisboa. Today, FenixEdu system is also implemented and deployed on other institutions, however, Técnico Lisboa continues to have the largest user basis, and to be the main source code contributor to the project. Not surprisingly, Técnico Lisboa implementation is the only one that makes full use of all platform features, as most of them stem directly from internal requirements.

Técnico Lisboa is the largest school of engineering in Portugal. It was founded in 1911 in the city of Lisbon, with the aim of developing top quality education and research in the areas of engineering, science and technology.

The mission of Técnico Lisboa is to create and to disseminate knowledge and to give our students the education and the knowledge tools to improve, to change, and to shape society through science, technology, and entrepreneurship by combining a top quality higher education, with Research, Development and Innovation (RD&I) activities, according to the highest international standards, immersing our students, alumni, faculty and staff in an exciting and global environment geared towards solving the challenges of the XXIst Century.

Técnico Lisboa is a research-oriented school, home to more than 6,000 undergraduate students (B.Sc.), 4,000 graduate students (MSc and PhD) and about 1,100 faculty members and researchers. Técnico Lisboa is settled in three campuses: (1) one in central Lisbon, which was the first campus and is currently the home to more than 9.000 students; (2) a technological and nuclear campus in Loures that focus mainly in research; and (3) another campus in Oeiras, at the Taguspark, a modern science and technology complex, home to more than 130 high-tech based companies, located in the southwest coast of Lisbon. Students can choose from a rich variety of MSc and PhD programs offered at Técnico Lisboa, covering the classical engineering and basic science fields as well as a number of the emerging scientific areas. A significant number of international students participate in Técnico Lisboa's Master and Doctoral programs.

Técnico Lisboa participates in worldwide cooperations with some of the best universities and R&D institutions. It is also a member of various European networks of prestigious schools in Engineering, Science and Technology, such as CLUSTER, TIME, and CESAER, committed to provide technical education at the highest international standards.

Research is a central activity at Técnico Lisboa, across several R&D units and disciplines. Teaching and research are closely connected and research and external consultancy contribute to more than half of the Técnico Lisboa turnover. Currently, Técnico Lisboa offers several Master and doctoral programs in collaboration with international partners (including Erasmus Mundus). Master and Doctoral students are regularly involved in these research activities, working together with Técnico Lisboa faculty and research staff, to face the challenges posed by ambitious national and international research projects.

2

Architecture

2.1 Technology

Fenix is a project has a great number of dependencies and uses several technologies that have been added to the project along with its development. Instead of talking about all of them we will just mention some of the more important ones to our architecture and its use will be described has we explain our architecture.

- Java7
- Java Server Pages
- Renderers
- Java Server Faces
- Struts
- Maven
- MySql
- FenixFramework
- Bennu

2.2 Domain Model

As you can see from the image, the same domain model is used across all the application code, this enables us to have a simpler architecture and is possible thanks to the use of FenixFramework.

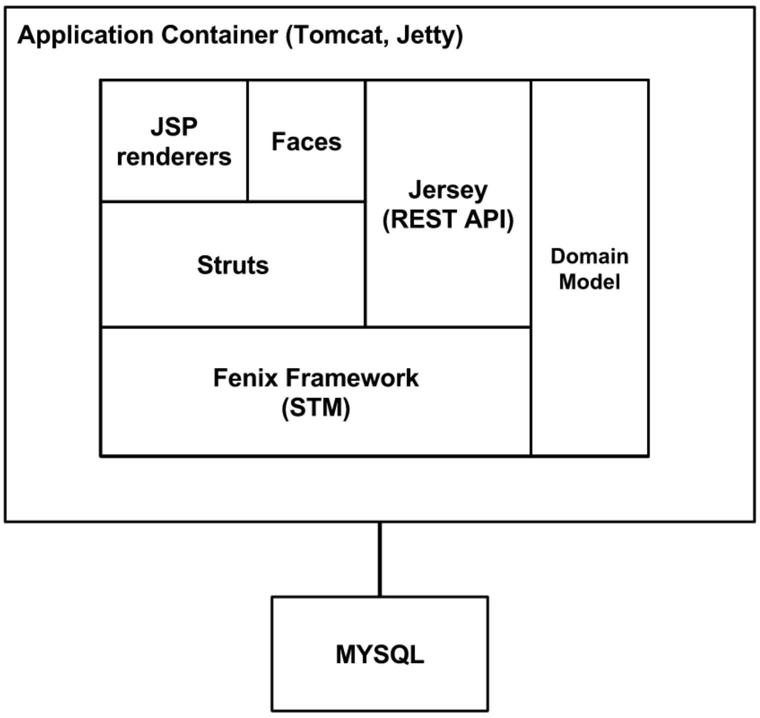


Figure 2.1: FenixEdu Architecture

Fenix Framework is a Java ORM that enables you to express the domain model as entities, the relations between the entities and its attributes. After having this specification, the framework automatically creates the Classes that implement the existing relations. You just have extend that classes and implement the application logic.

The framework takes care of the persistency layer, saving the changes to the entities on the database (MySQL). It also handles the transactions and provides ways to ensure consistency. All the data is kept on memory and the transactions are also done on memory. The updates done are saved on the database from time to time.

2.2.1 How to use transactions

To declare a transaction you have to write your transactional method and it needs to be annotated with the keyword @Atomic. This annotation can receive as argument the type of transaction it will perform, it can be READ if it just reads data, WRITE if it does any write, SPECULATIVE_READ if it normally just reads but sometimes it needs to write. The transactions can't be used directly on the Action's code, they should be on the domain. This transactions will be run on the Software Transactional Memory and if they fail doe to a write-write conflict then they will run more than one time. By these reason you have to be careful when contacting external services.

2.2.2 How are the relations saved

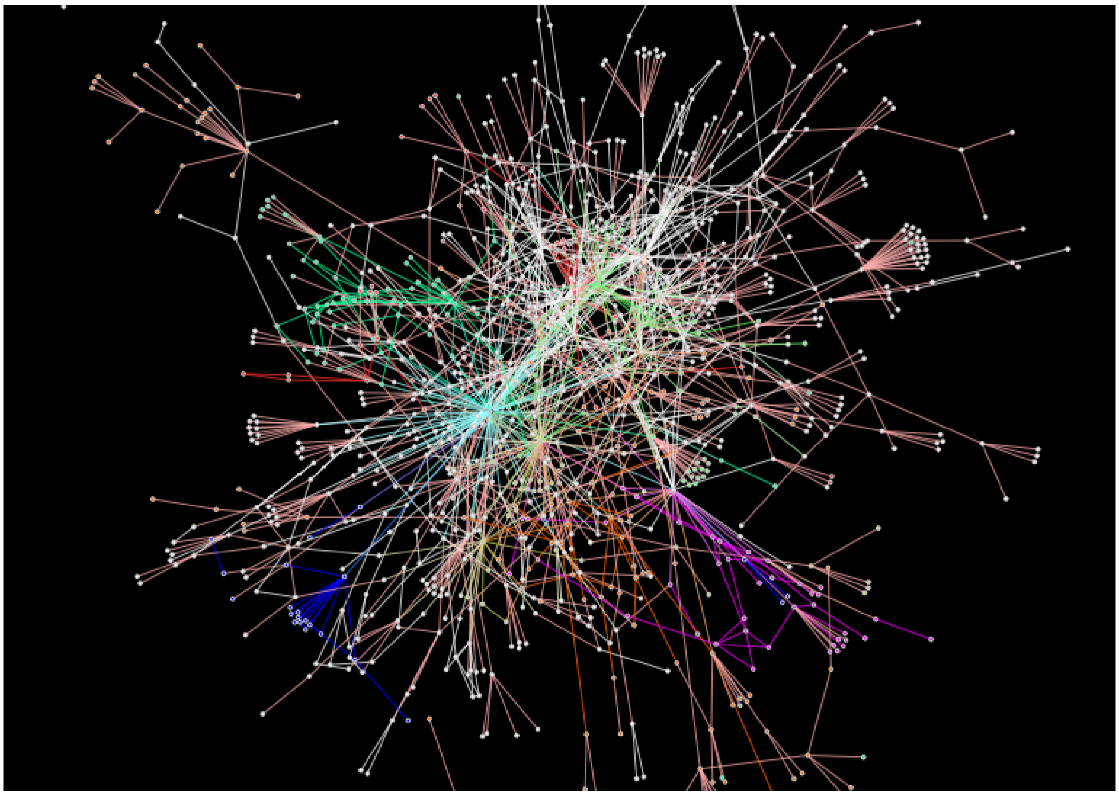


Figure 2.2: FenixEdu Domain Model Graph

The domain model has 1370 Entities, 1802 Relations and 54 Value Types. By this reason, it can be hard to find out the implications of changing a given entity or relation. We have developed a tool named Fenix Domain Browser[1], that enables you to graphically visualise the relations between entities.

Each class is represented on the Database as a Table. This table is named with the name of the class or its superclass. The properties of that class are columns of the table. The relations between the objects are also columns on that table having has values the Object ID of the related object. When a the relation is * to * then we have a new table <Class1_Name>_<Class2Name> with two columns with the Object IDs of the objects who are related.

The persistency layer is hidden from the programmer and the Fenix Framework makes the relation between the Java entities and the Database (MySQL). FenixFramework also handles the transactions and the consistency.

[1] `https://fenix-ashes.ist.utl.pt/fdb/`

2.3 Rest Interfaces

We have REST API that enables developers to build applications that use the academic information available. Before the developers can use this information, the students need to give the required access to the application.

2.4 Struts

Struts is a framework for web applications, It uses the Java Servlet API and implements the model view controller architecture (MVC). we are using the version 1.2.7[2].

The goal of struts is to separate the Model from the view and controllers. It Does the route between the requested path to the Controller and from the Controller to the View to be rendered.

Each Controller is implemented as a Dispatch Action, they fetch the information needed to the view and update the domain with the needed changes. It also express the forward between views.

Each view is written using Java Server Pages or Java Server Faces. This technologies enables you to write dynamic pages that are filled with information provided by the Actions.

2.4.1 How do STRUTS process a Request?

When a request arrives the Struts Controller Server, it inspects the information on the request and based on that it knows the Action for which it should dispatch the request. This Action will perform the appropriate business logic by reading, creating or updating the Domain Model. It also adds to the request all the information needed to render the view. After that, it returns a forward to the Controller Servlet that indicates the view that should be rendered. Finally the view is sent to the client as an HTTP response.

For example, if a client sends a request for:

```
http://<fenixEdu>/public/executionCourse.do?method=marks&executionCourseID
    =1610612925705
```

This url can be decomposed in several parts. The initial part indicates that it is related with the module named public, so the mappings definition with the Actions will be on the file named struts-public.xml

```
<action type”=”ExecutionCourseDA parameter”=”method path”=/”executionCourse>
```

This mappings can also be done using annotations directly on the Action, on this case the ExecutionCourseDA it would be:

[2] http://struts.apache.org/release/1.2.x/

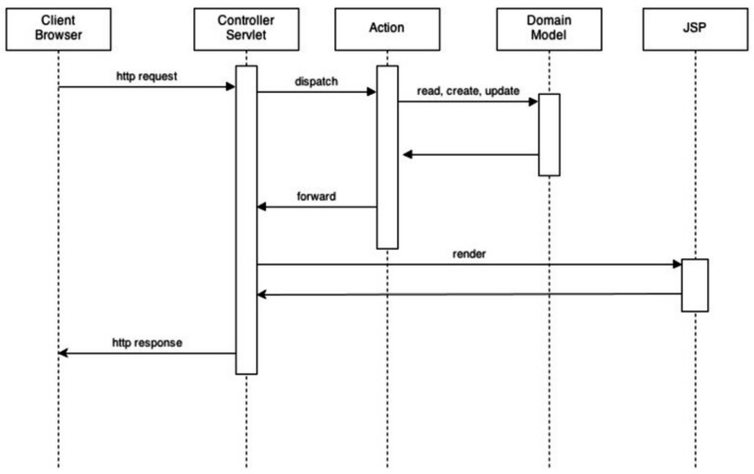


Figure 2.3: Struts Control Layer

```
@Mapping(module”=”public path”=/”executionCourse)
```

After discovering the Action, it needs to discover the method that will be invoked. This information is sent on the request and on this case we have “method=marks”, so it will invoke the method marks on the action ExecutionCourseDA. All the remaining information on the URL are the parameters of that request and are available to the method. On this case we are asking for a specific execution course.

After executing the method it will return the forward to the view to be rendered. This forward can also be written using two methods: by annotation on the action or by definition on the struts-Module.xml file.

2.5 Presentation

The main technology used to define our views is Java Server Pages (JSP). It is an HTML extension that enables you to build dynamic pages with behaviour. You define a template of HTML using Java embedded code. When used with struts (as it is the case) it provides you with lots of tags that cover the majority of functionality needed like print a given value from an object or iterate a collection of objects.

Besides JSP we also have use Java Server Faces (JSF) on our project. JSF is very similar to JSP, the main difference is that instead of setting the view parameters on the http request, they are loaded from a Bean. The Bean is a Java class that has all the information needed for the view. It also implements the behaviour for that view.

2.5.1 Renderers

There are some king of domain objects that are always rendered on the same way, to avoid having lots repeated code our team have developed Renderers. This technology basically provides a tag that knows how to directly render domain objects, depending on their type.

Part I

Installation and User Guide

3

Getting Started

3.1 Installing FenixEdu

The first part of this guide is to install FenixEdu. This procedure will and download all the files and create all the tables on the database necessary for the system to run.

3.1.1 Install the necessary software

Before you can install FenixEdu locally, you must install some software packages in the hosting machine. The following commands, intended for ubuntu-based operating systems, will add a new repository that allows you to install Oracle's JDK via apt-get and will install all the required software:

```
sudo apt-get install maven
sudo apt-get install git
sudo apt-get install mysql-server mysql-client
sudo add-apt-repository ppa:webupd8team/java
sudo apt-get install oracle-jdk7-installer
```

3.1.2 Configure Java and Maven

FenixEdu is a big web application. That is why the default memory parameters are not enough to compile and run an instance of FenixEdu.

To circumvent this issue, you must export both the `JAVA_OPTS` and `MAVEN_OPTS` to your operating system environment. To do this, you can add the following entries to your `.bashrc`:

```
export JAVA_OPTS="-server␣-Xms256m␣-Xmx1024m␣-XX:PermSize=384m"
export MAVEN_OPTS="$JAVA_OPTS␣-Dorg.apache.jasper.compiler.Parser.
    STRICT_QUOTE_ESCAPING=false"
```

After this, you should `source` your `.bashrc` file or simply restart your shell.

3.1.3 Create an empty database

Assuming that your mysql user is `root` and that the database name where you will install FenixEdu is called `fenixedu`, you must create such database. For that, you can execute the following one-liner:

```
mysql -uroot -e "create␣database␣fenixedu" -p
```

The command above will create an empty databased named `fenixedu`.

Attention:
The mysql user you will specify during the installation must have the necessary grants to write and create new tables.

3.1.4 Generate the FenixEdu software installation through the archetype

Use the following command:

```
mvn archetype:generate -DarchetypeGroupId=org.fenixedu -DarchetypeArtifactId=
    fenix-webapp-archetype -DarchetypeVersion=1.0.0 -DarchetypeRepository=
    https://fenix-ashes.ist.utl.pt/nexus/content/groups/fenix-ashes-maven-
    repository
```

You will need to provide some information concerning the artifact to be generated through an interactive prompt. An example of this interactive prompt project generation using the archetype follows:

```
Define value for property 'groupId': : com.example
Define value for property 'artifactId': : fenix-webapp
Define value for property 'version':  1.0-SNAPSHOT: :
Define value for property 'package':  com.example: : com.example.fenixedu
Define value for property 'databaseHost': : localhost
Define value for property 'databaseName': : fenixedu
Define value for property 'databasePassword': : pass
Define value for property 'databaseUsername': : root
Define value for property 'fenixVersion': : 2.0.5
```

```
[WARNING] Archetype is not fully configured
[INFO] Using property: groupId = com.example
[INFO] Using property: artifactId = fenix-webapp
[INFO] Using property: version = 1.0-SNAPSHOT
[INFO] Using property: package = com.example.fenixedu
[INFO] Using property: databaseHost = localhost
[INFO] Using property: databaseName = fenixedu
Define value for property 'databasePassword': : pass
[INFO] Using property: databaseUsername = root
[INFO] Using property: fenixVersion = 2.0.5
Confirm properties configuration:
groupId: com.example
artifactId: fenix-webapp
version: 1.0-SNAPSHOT
package: com.example.fenixedu
databaseHost: localhost
databaseName: fenixedu
databasePassword: pass
databaseUsername: root
fenixVersion: 2.0.5
 Y: : y
```

Attention:
Due to a Maven limitation, the password for the database cannot be empty. Hence, you must provide a non-empty password like `pass`, and later edit the file located at `<artifactId>/src/main/resources/fenix-framework.properties`, delete the password and save the file.

By now, Maven should have created a folder in your current working directory with the name of the artifactId that you provided. In the example above, the folder name is `fenix-webapp`.

3.1.5 Bootstrap FenixEdu

After Maven generates the installation code, you must bootstrap it. By bootstraping, we mean running a Maven plugin that will prompt you with a set of parameters to populate the configured database with the necessary information. To run such Maven task, run the `install` profile through the following commands:

```
cd fenix-webapp
mvn prepare-package -P install
```

After you run this command, FenixEdu will compile and run an interactive prompt application that will ask you a set of parameters needed to bootstrap your installation. An example of the parameters that will be prompt follows:

```
This process will guide you in installing FenixEdu in your School.
First in what country are you at?
Country (Three Letter ISO 3166—1) [USA]:
Using United States

Let’s setup your school. You need to set up your university and school name.
University Name [Example University]:
University Acronym [EU]:
School Name [Example Engineering School]:
School Acronym [EES]:

Next we need to setup the domain information about your applicatinos.
School Domain [ees.example.edu]:
School URL [http://ees.example.edu]:
School Email Domain [ees.example.edu]:
Installation Name [EES FenixEdu]:
Installation Domain [fenixedu.ees.example.edu]:

Now we need to create an administrator account.
Username [nurv]:
Name [FenixEdu Administrator]:
Email [nurv@ees.example.edu]:
Password:
Password (again):

Starting install process...
Installation Complete.
```

3.1.6 Run your FenixEdu Application

After bootstraping your FenixEdu installation, you are ready to run it. To do so, you must execute the following Maven command that will run a Tomcat Servlet Container with your FenixEdu installation:

```
mvn tomcat7:run
```

After the startup is finished, you can login into your FenixEdu installation by providing the Administrator account credentials you entered in the bootstrap step at the following URL:

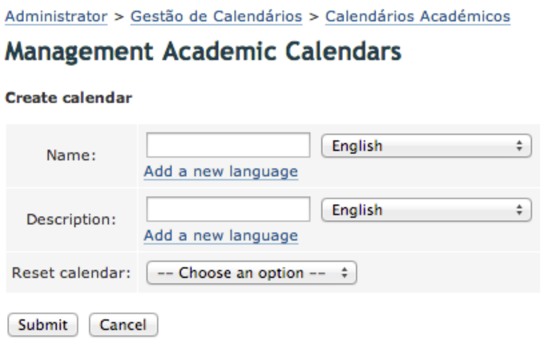
http://localhost:8080/fenix/

3.2 Configuring FenixEdu

Now that your application is running, it is necessary to configure it to a given school. We will guide you through the process of inserting all the information necessary to configure one degree with one course, and enrol one student. The data we are using here is for example purposes only, you should use data that suits the needs of your school.

3.2.1 Creating Curricular Calendars

The first step is to create the curricular calendars. This represents the periods where things are supposed to happen (like classes, enrolments, exams etc). To read more about this, take a look at <MISSING>. First go to **Administrator > Gestão de Calendários > Calendários Académicos** and create Create calendar.


Administrator > Gestão de Calendários > Calendários Académicos

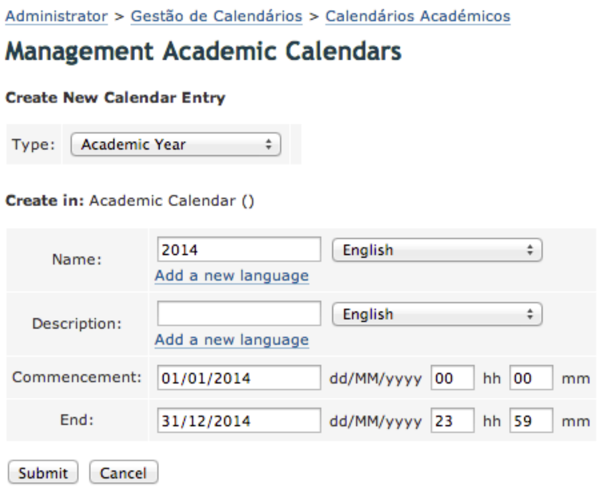
Management Academic Calendars

Create calendar

Name:	[ ] English / Add a new language
Description:	[ ] English / Add a new language
Reset calendar:	-- Choose an option --

Figure 3.1: Creating an academic calendar

Call this calendar *"Academic Calendar"* for this example. Next click **Create entry** to create a new year. Select type **Academic Year**, choose, name it the current year (e.g. 2014), the beginning at *01/01/2014*, *00h* and *00m*. and the end at *31/12/2014*, *23h* and *59m* (Image 3.2).


Administrator > Gestão de Calendários > Calendários Académicos

Management Academic Calendars

Create New Calendar Entry

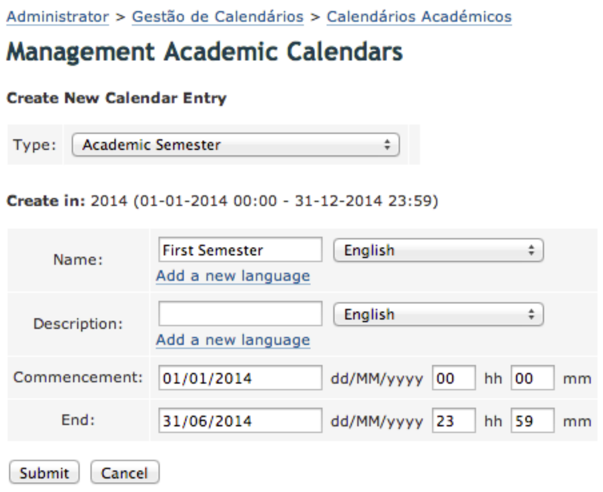
Type: Academic Year

Create in: Academic Calendar ()

Name:	2014 English / Add a new language
Description:	[ ] English / Add a new language
Commencement:	01/01/2014 dd/MM/yyyy 00 hh 00 mm
End:	31/12/2014 dd/MM/yyyy 23 hh 59 mm

Figure 3.2: Creating a academic year

Now lets create an academic semester. Click **Create entry**, select the type **Academic Semester** and name it *First Semester*, the beginning at *01/01/2014*, *00h* and *00m*. and the end at *31/06/2014*, *23h* and *59m* (Image 3.3).


Administrator > Gestão de Calendários > Calendários Académicos

Management Academic Calendars

Create New Calendar Entry

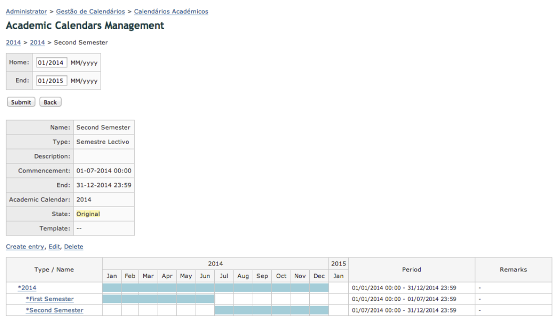
Type: Academic Semester

Create in: 2014 (01-01-2014 00:00 - 31-12-2014 23:59)

Name:	First Semester · English · Add a new language
Description:	· English · Add a new language
Commencement:	01/01/2014 dd/MM/yyyy 00 hh 00 mm
End:	31/06/2014 dd/MM/yyyy 23 hh 59 mm

Figure 3.3: Creating a academic year

Select the year in question to add another semester and do the same as before, but this time for a *Second Semester*, beginning at *01/07/2014*, *00h* and *00m*. and the end at *31/12/2014*, *23h* and *59m*. After this, the calendar should look like as shown on Image 3.4.


Administrator > Gestão de Calendários > Calendários Académicos

Academic Calendars Management

2014 > 2014 > Second Semester

Home:	01/2014 MM/yyyy
End:	01/2015 MM/yyyy

Name:	Second Semester
Type:	Semestre Lectivo
Description:	
Commencement:	01-07-2014 00:00
End:	31-12-2014 23:59
Academic Calendar:	2014
State:	Original
Template:	--

Create entry, Edit, Delete

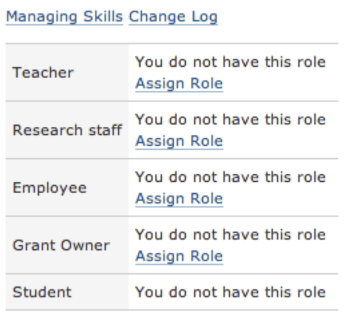
Type / Name	2014												2015	Period	Remarks
	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Jan		
*2014	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦		01/01/2014 00:00 - 31/12/2014 23:59	-
*First Semester	🟦	🟦	🟦	🟦	🟦	🟦								01/01/2014 00:00 - 01/07/2014 23:59	-
*Second Semester							🟦	🟦	🟦	🟦	🟦	🟦		01/07/2014 00:00 - 31/12/2014 23:59	-

Figure 3.4: The resulting calendar

3.2.2 Make the user an employee

The next step is to make this user an employee. Due to some restrictions, it is necessary to preform some administrative actions. These restrictions are expected to be removed in the future, and to be used a pure grant base system.

Select the tab **Área de Pessoal** and search for your username. Select the user from the search result, and you should get an information page like in Image 3.5



Managing Skills Change Log

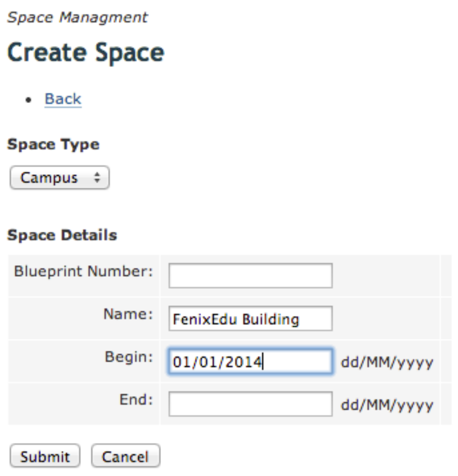
Teacher	You do not have this role Assign Role
Research staff	You do not have this role Assign Role
Employee	You do not have this role Assign Role
Grant Owner	You do not have this role Assign Role
Student	You do not have this role

Figure 3.5: Personal Information

Click the **Assign Role** link next to employee, and this user should get a employee number.

3.2.3 Create a Campus

We need to create a campus for this school. Click on the tab **Space Management**. Then click the link **Create Space**. Select the campus type, and then give it a name and a date when it should start operating (Image 3.6).



Space Managment

Create Space

- Back

Space Type

Campus

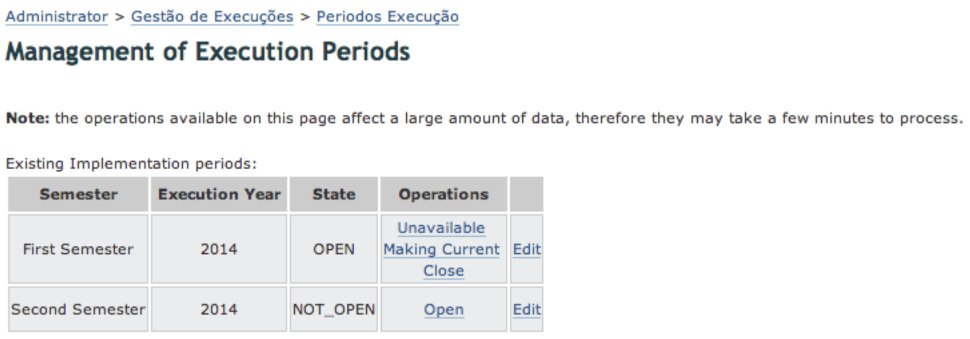
Space Details

Blueprint Number:		
Name:	FenixEdu Building	
Begin:	01/01/2014	dd/MM/yyyy
End:		dd/MM/yyyy

Figure 3.6: Creating a Campus

3.2.4 Activate the Current Period

Next we should activate the period we've created before. Go to **Administrator > Períodos Execução**, and you should see the two Semesters that you have created before. First click on **Open** on the *First Semester*. Next you should click **Make Current** to make that semester the currently active (Image 3.7).



Administrator > Gestão de Execuções > Periodos Execução

Management of Execution Periods

Note: the operations available on this page affect a large amount of data, therefore they may take a few minutes to process.

Existing Implementation periods:

Semester	Execution Year	State	Operations	
First Semester	2014	OPEN	Unavailable Making Current Close	Edit
Second Semester	2014	NOT_OPEN	Open	Edit

Figure 3.7: Make current period active

3.2.5 Create Associated Object

Now we need to create some primitive objects about the organization of your school. To do this, head to **Administrator > Gerir Objectos Associados**.

Create a Department

First lets create a Department. Click in the **Create** link below *Departments*. Fill the field with the following information:

- **Active:** checked
- **Code:** DCivil
- **Name:** Departamento Civil
- **Real Name:** Departamento Civil
- **Real Name (English):** Civil Department

When you are deploying your installation of FenixEdu you will need to create an object of these for each department in your institution.

Create a Academic Office

In the *Associated Objects* page click in **Create** link bellow *Administrative*. Select the type *Administrative Office* and click **Submit**.

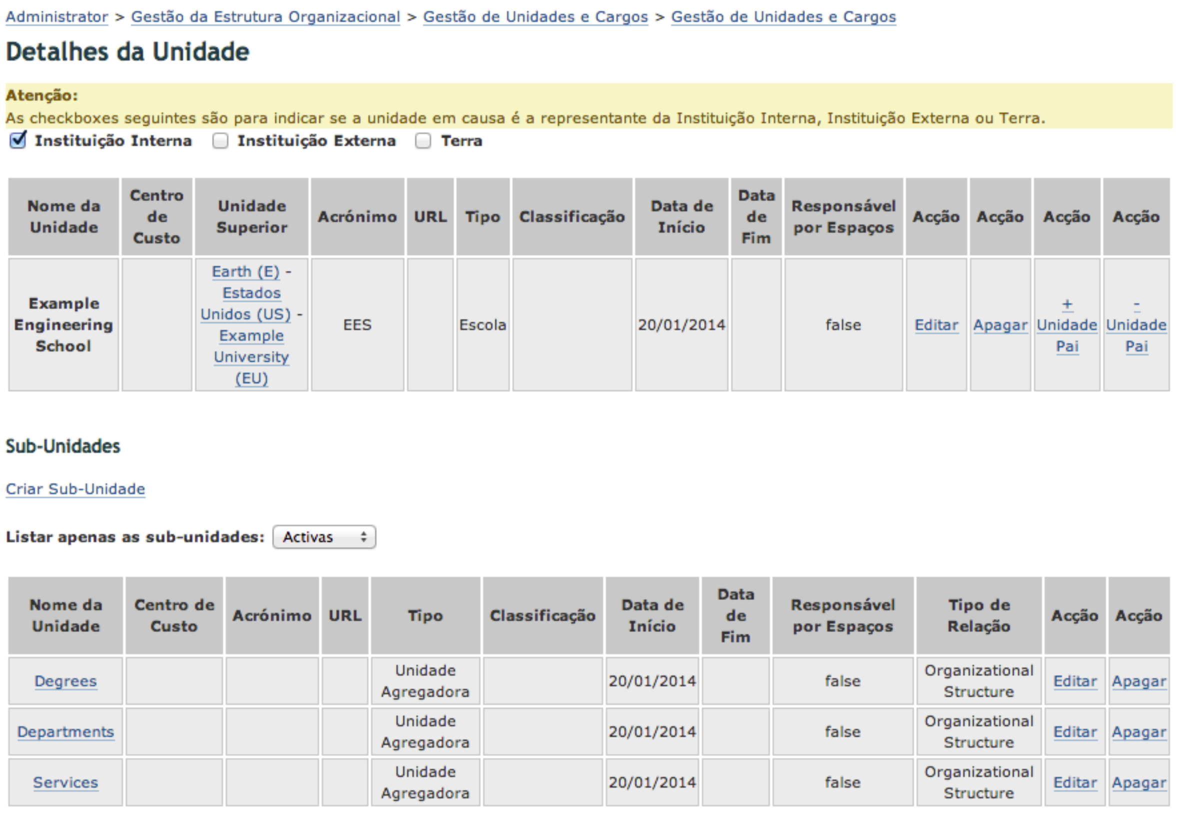
3.2.6 Units and Roles Management

Now we need to insert the entities we just created into the Organizational Structure. This tree structure represents the hierarchical relationship between entities. Each association is represented by a Unit. We are going to create a skeleton structure to allow us to proceed in configuring the system.

To manipulate this structure go to **Administrator > Gestão de Unidades e Cargos**. Here use the [+] sign to expand each tree node. Go to:

```
Earth > [your school country] > [your university] > [your school]
```

Click on your school link and you should get to the same page as in Image 3.8.



Administrator > Gestão da Estrutura Organizacional > Gestão de Unidades e Cargos > Gestão de Unidades e Cargos

Detalhes da Unidade

Atenção:
As checkboxes seguintes são para indicar se a unidade em causa é a representante da Instituição Interna, Instituição Externa ou Terra.

☑ **Instituição Interna** ☐ **Instituição Externa** ☐ **Terra**

Nome da Unidade	Centro de Custo	Unidade Superior	Acrónimo	URL	Tipo	Classificação	Data de Início	Data de Fim	Responsável por Espaços	Acção	Acção	Acção	Acção
Example Engineering School		Earth (E) - Estados Unidos (US) - Example University (EU)	EES		Escola		20/01/2014		false	Editar	Apagar	+ Unidade Pai	- Unidade Pai

Sub-Unidades

Criar Sub-Unidade

Listar apenas as sub-unidades: Activas

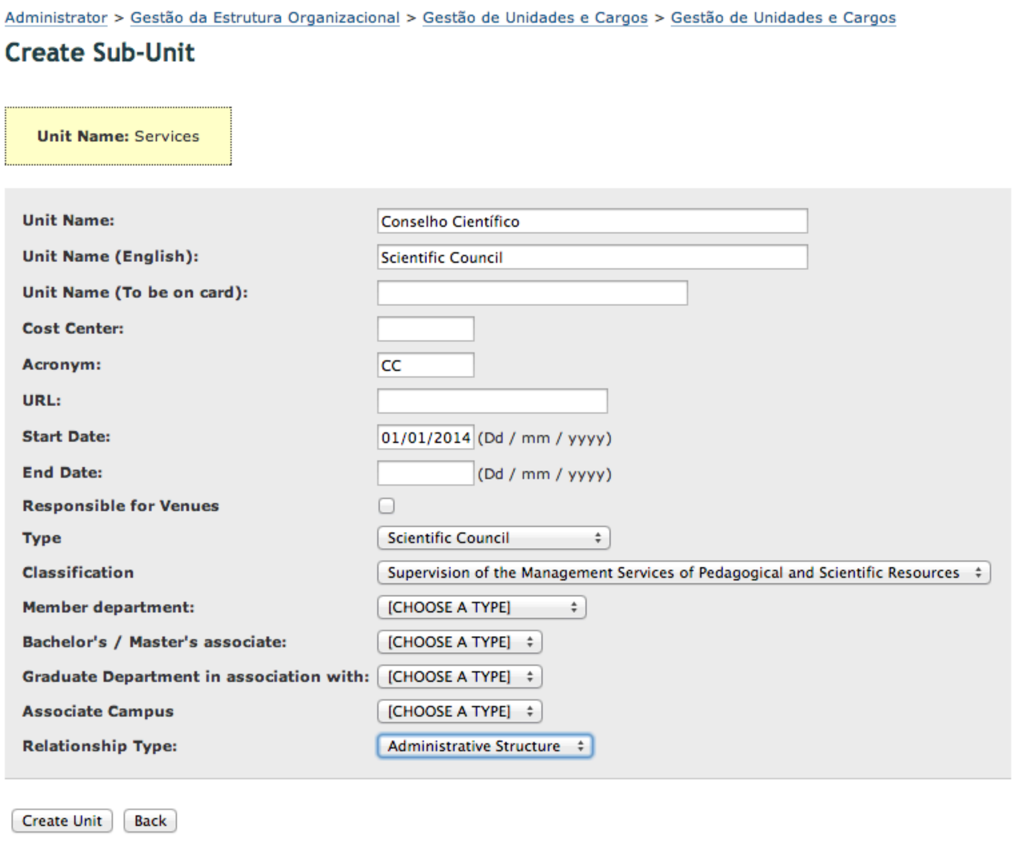
Nome da Unidade	Centro de Custo	Acrónimo	URL	Tipo	Classificação	Data de Início	Data de Fim	Responsável por Espaços	Tipo de Relação	Acção	Acção
Degrees				Unidade Agregadora		20/01/2014		false	Organizational Structure	Editar	Apagar
Departments				Unidade Agregadora		20/01/2014		false	Organizational Structure	Editar	Apagar
Services				Unidade Agregadora		20/01/2014		false	Organizational Structure	Editar	Apagar

Figure 3.8: Your school node

Create a Scientific Council

The first unit we need to create is the Scientific Council. This council is responsible for giving scientific approval of curricular degrees and courses.

On your school node page click in the **Services** Unit. On the *Services* node page then click the link **Criar Sub-unidade** to create a new unit. Fill the form with the same information as in Image 3.9


Administrator > Gestão da Estrutura Organizacional > Gestão de Unidades e Cargos > Gestão de Unidades e Cargos
Create Sub-Unit
Unit Name: Services
Create Unit
Back

Unit Name:	Conselho Científico
Unit Name (English):	Scientific Council
Unit Name (To be on card):	
Cost Center:	
Acronym:	CC
URL:	
Start Date:	01/01/2014 (Dd / mm / yyyy)
End Date:	(Dd / mm / yyyy)
Responsible for Venues	☐
Type	Scientific Council
Classification	Supervision of the Management Services of Pedagogical and Scientific Resources
Member department:	[CHOOSE A TYPE]
Bachelor's / Master's associate:	[CHOOSE A TYPE]
Graduate Department in association with:	[CHOOSE A TYPE]
Associate Campus	[CHOOSE A TYPE]
Relationship Type:	Administrative Structure

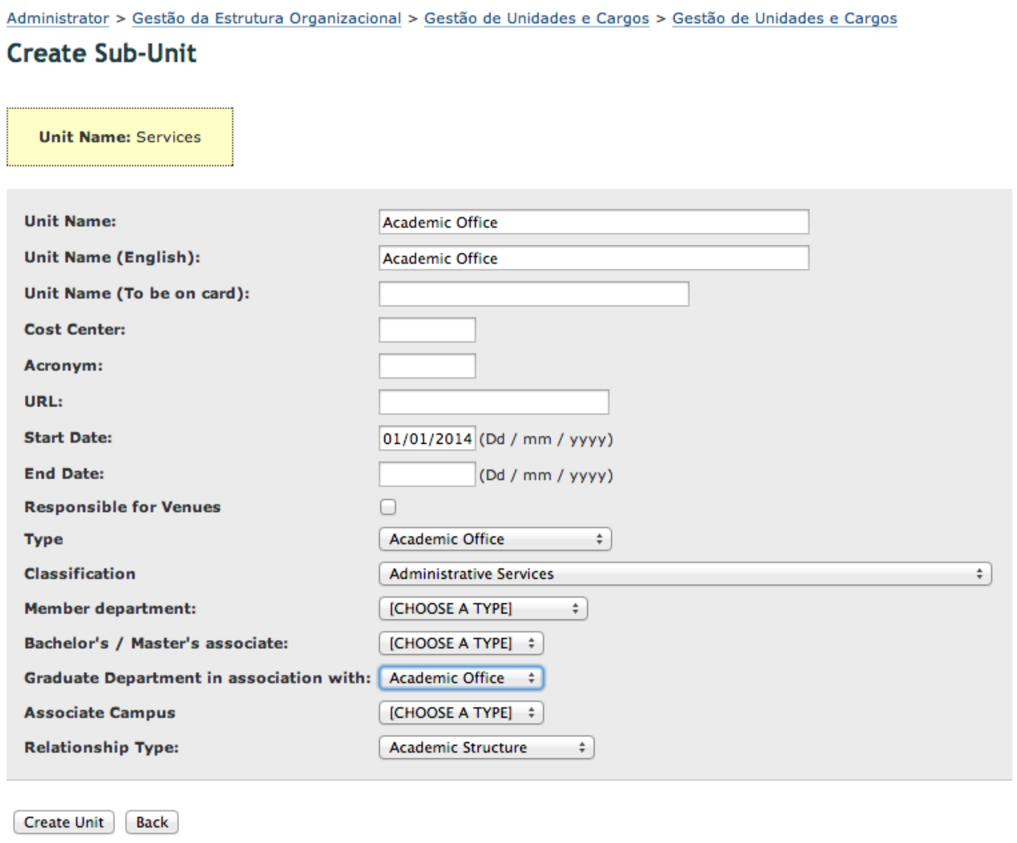
Figure 3.9: Creating Scientific Council

Attaching the Academic Office

Next we need to attach the Academic Office we previously created into the structure. Go back to the *Services* node page and click again the link **Criar Sub-unidade**. This time fill with the same information as in Image 3.10.

Create a Department

We now are going to attach the departments we previously created into the tree. To do that, go back to the school node page and but this time select the **Departments** node and then click the link **Criar Sub-unidade** to create a new unit. Fill that with the same information as in Image 3.11.



Administrator > Gestão da Estrutura Organizacional > Gestão de Unidades e Cargos > Gestão de Unidades e Cargos

Create Sub-Unit

Unit Name: Services

Unit Name:	Academic Office
Unit Name (English):	Academic Office
Unit Name (To be on card):	
Cost Center:	
Acronym:	
URL:	
Start Date:	01/01/2014 (Dd / mm / yyyy)
End Date:	(Dd / mm / yyyy)
Responsible for Venues	☐
Type	Academic Office
Classification	Administrative Services
Member department:	[CHOOSE A TYPE]
Bachelor's / Master's associate:	[CHOOSE A TYPE]
Graduate Department in association with:	Academic Office
Associate Campus	[CHOOSE A TYPE]
Relationship Type:	Academic Structure

Figure 3.10: Creating Academic Office

Create a Scientific Area within the Department

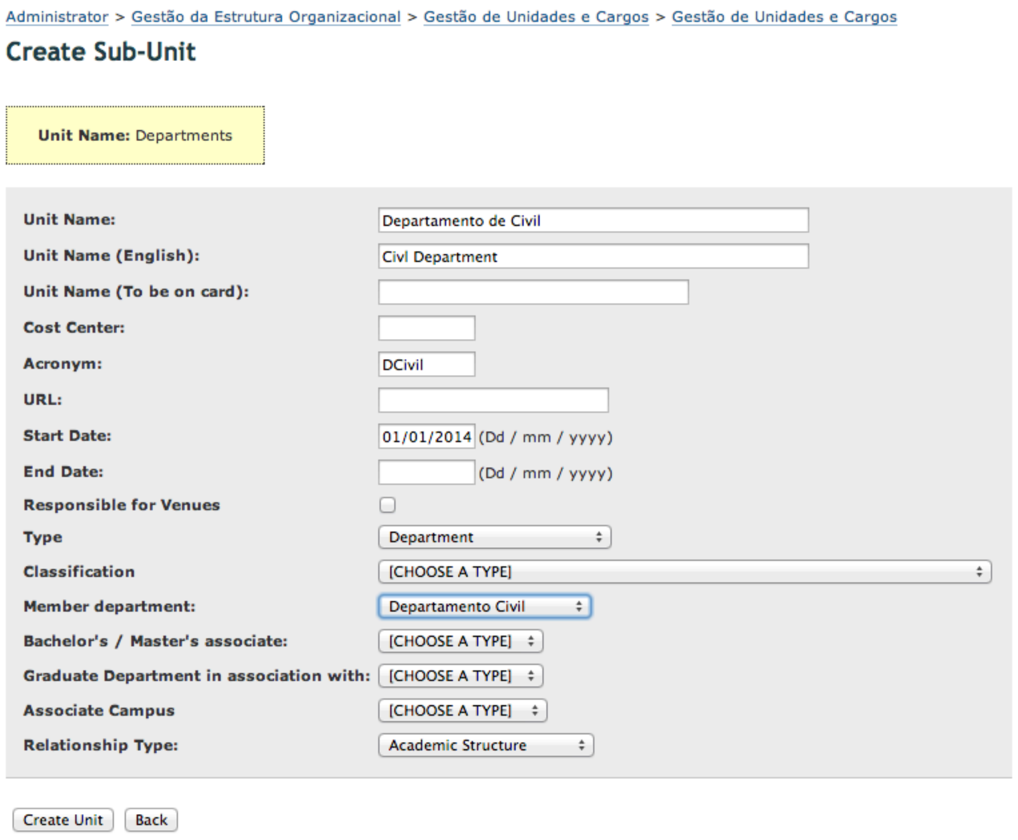
In FenixEdu, Courses are grouped together within a Scientific Area. You can see this as, for instance, the group of courses of *Distributed Systems* within the course of *Computer Science*.

To create one, first go to the department node page to add a Scientific Area and then click the link **Criar Sub-unidade**. Next fill with the same information as Image 3.12.

Create a Competence Course Group within the Scientific Area

Competence courses are courses descriptions that can be instanced for every execution. You can see this concept as the *Operative Systems* Competence Course can be instanced as *Operative Systems 2012*, *Operative Systems 2013* and *Operative Systems 2014*. These Competence Courses are organized in Competence Course Groups. We need to create one of those for our application.

To create one, first go to the scientific area node page that we've just created and then click the link **Criar Sub-unidade**. Next fill with same information as Image 3.13.


Administrator > Gestão da Estrutura Organizacional > Gestão de Unidades e Cargos > Gestão de Unidades e Cargos

Create Sub-Unit

Unit Name: Departments

Unit Name:	Departamento de Civil
Unit Name (English):	Civl Department
Unit Name (To be on card):	
Cost Center:	
Acronym:	DCivil
URL:	
Start Date:	01/01/2014 (Dd / mm / yyyy)
End Date:	(Dd / mm / yyyy)
Responsible for Venues	☐
Type	Department
Classification	[CHOOSE A TYPE]
Member department:	Departamento Civil
Bachelor's / Master's associate:	[CHOOSE A TYPE]
Graduate Department in association with:	[CHOOSE A TYPE]
Associate Campus	[CHOOSE A TYPE]
Relationship Type:	Academic Structure

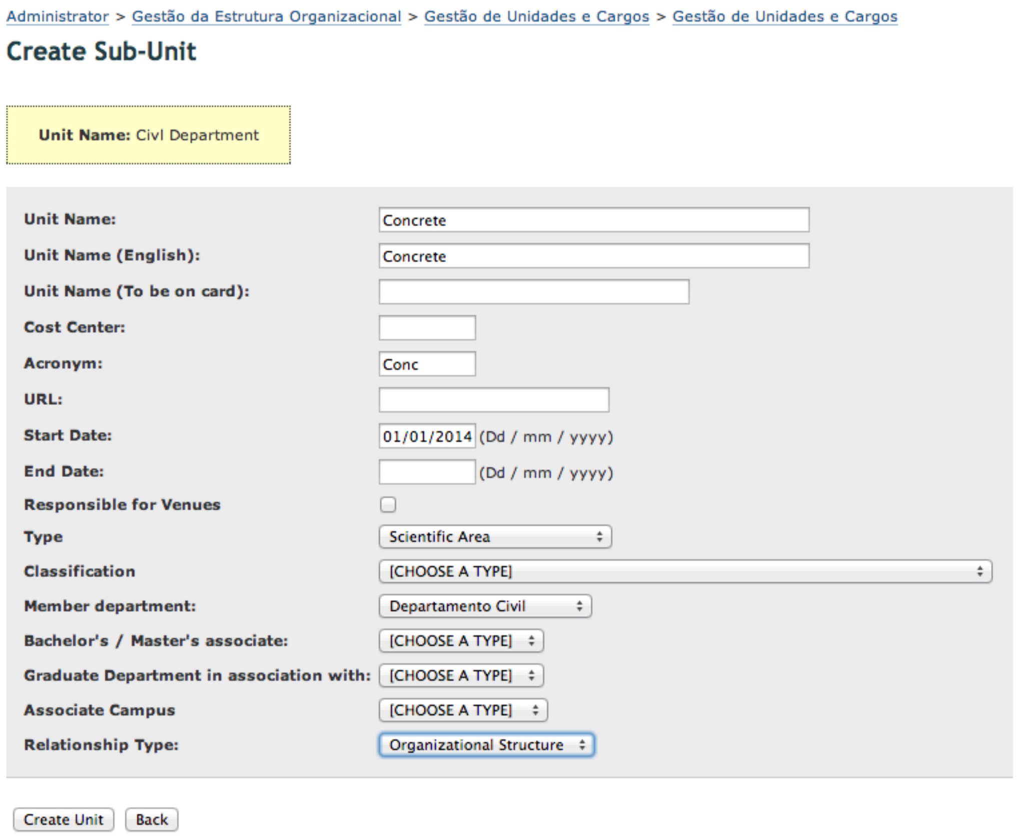
Figure 3.11: Creating a Department

Attach the user to the Department

We previously have upgraded the user to an employee. We now need to attach this employee to a place, in this case to a department. Only people working on a given department can operate over their degrees and courses. This means that you have to have at least one user for each department that you have created. We are in the process to replace this with a grant based authorization scheme.

For now, go to the **Administrator > Gerir Objectos Associados** and then click the **Create** bellow the *Associate Person to Unit*. Fill the form with the following information:

- **Username:** <your user>
- **Accountability Type:** Working Contract
- **Teacher:** *Unchecked*
- **Start:** 01/01/2014
- **Unit:** Civil Department


Administrator > Gestão da Estrutura Organizacional > Gestão de Unidades e Cargos > Gestão de Unidades e Cargos

Create Sub-Unit

Unit Name: Civl Department

Unit Name:	Concrete
Unit Name (English):	Concrete
Unit Name (To be on card):	
Cost Center:	
Acronym:	Conc
URL:	
Start Date:	01/01/2014 (Dd / mm / yyyy)
End Date:	(Dd / mm / yyyy)
Responsible for Venues	☐
Type	Scientific Area
Classification	[CHOOSE A TYPE]
Member department:	Departamento Civil
Bachelor's / Master's associate:	[CHOOSE A TYPE]
Graduate Department in association with:	[CHOOSE A TYPE]
Associate Campus	[CHOOSE A TYPE]
Relationship Type:	Organizational Structure

Figure 3.12: Creating Scientific Area

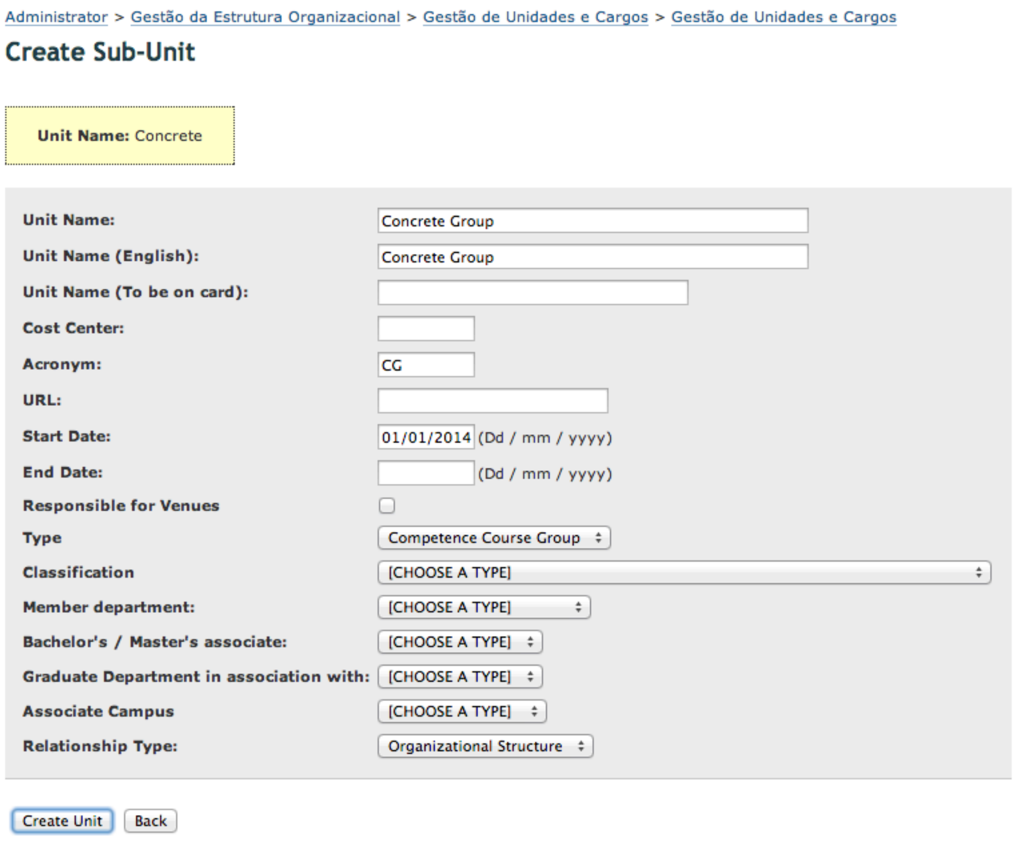
Hit *Submit*.

3.2.7 Competence Courses

Now we are going to create the competence courses. Their are created before creating courses because some courses might be administered to more than one degree (e.g. the same course of *Calculus* being thought to *Civil Engineering* and *Computer Science*) To learn more about competence courses check Section 5.1

Authorizations within Department Administrative Office

The user now needs to give himself authorization to manage the competence groups. To do this go to **Secretaria de Departamento > Competence Course**. Check the checkbox in from of the user you want to add to this group and then hit **Adicionar Pessoa(s)**.



Administrator > Gestão da Estrutura Organizacional > Gestão de Unidades e Cargos > Gestão de Unidades e Cargos

Create Sub-Unit

Unit Name: Concrete

Unit Name:	Concrete Group
Unit Name (English):	Concrete Group
Unit Name (To be on card):	
Cost Center:	
Acronym:	CG
URL:	
Start Date:	01/01/2014 (Dd / mm / yyyy)
End Date:	(Dd / mm / yyyy)
Responsible for Venues	☐
Type	Competence Course Group
Classification	[CHOOSE A TYPE]
Member department:	[CHOOSE A TYPE]
Bachelor's / Master's associate:	[CHOOSE A TYPE]
Graduate Department in association with:	[CHOOSE A TYPE]
Associate Campus	[CHOOSE A TYPE]
Relationship Type:	Organizational Structure

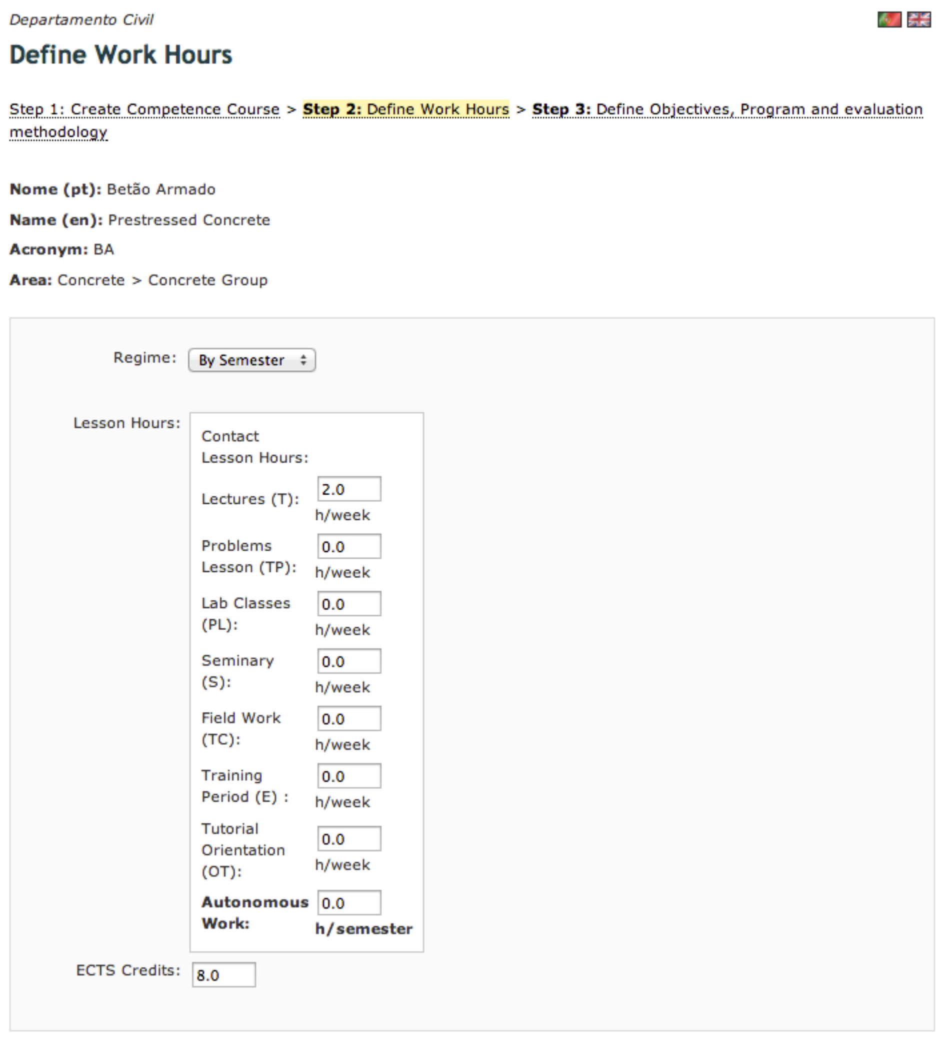
Figure 3.13: Creating a Competence Course Group

Creating Competence Courses

To create a competence course, head to **Bolonha > View (Competence Course)**. There you have the Scientific Areas and Competence Course Group you've just created before. Click in **Criar Disciplina** next to the competence course group, in our example *Concrete Group*.

In the first step, fill the form with the following information:

- **Início**: First Semester 2014
- **Nome (pt)**: Betão Armado
- **Nome (en)**: Prestressed Concrete
- **Nível**: 1º Ciclo (L)
- **Tipo**: Normal


Departamento Civil

Define Work Hours

Step 1: Create Competence Course > **Step 2:** Define Work Hours > **Step 3:** Define Objectives, Program and evaluation methodology

Nome (pt): Betão Armado

Name (en): Prestressed Concrete

Acronym: BA

Area: Concrete > Concrete Group

Regime: By Semester

Lesson Hours:

Contact Lesson Hours:

Lectures (T):	2.0	h/week
Problems Lesson (TP):	0.0	h/week
Lab Classes (PL):	0.0	h/week
Seminary (S):	0.0	h/week
Field Work (TC):	0.0	h/week
Training Period (E) :	0.0	h/week
Tutorial Orientation (OT):	0.0	h/week
Autonomous Work:	0.0	**h/semester**

ECTS Credits: 8.0

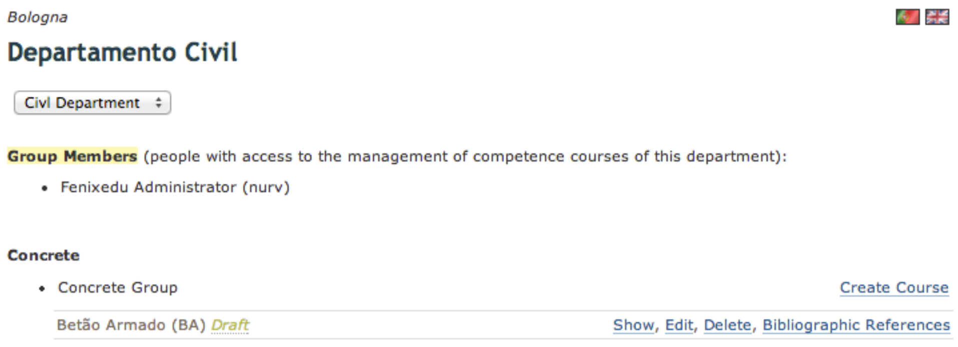
Figure 3.14: Defining Work Hours

And hit **Submit**. Next we need to define the work hours. Fill with the same information as in Image 3.14 and hit **Submit** Finally we need to define the objectives, program, and evaluation methodology. You can leave them blank and hit **Submit**.

You should end up with a curricular course as shown on Image 3.15

Publish Competence Course

Now its required to publish the competence course. Click in **Edit** operation on the course we just created. You are presented with a page where you can edit all the information you've entered before. We now are interestesd in editing the state, from *Draft* to *Publish*. Click on the first **Edit** link on that page.



Bologna

Departamento Civil

Civl Department

Group Members (people with access to the management of competence courses of this department):

- Fenixedu Administrator (nurv)

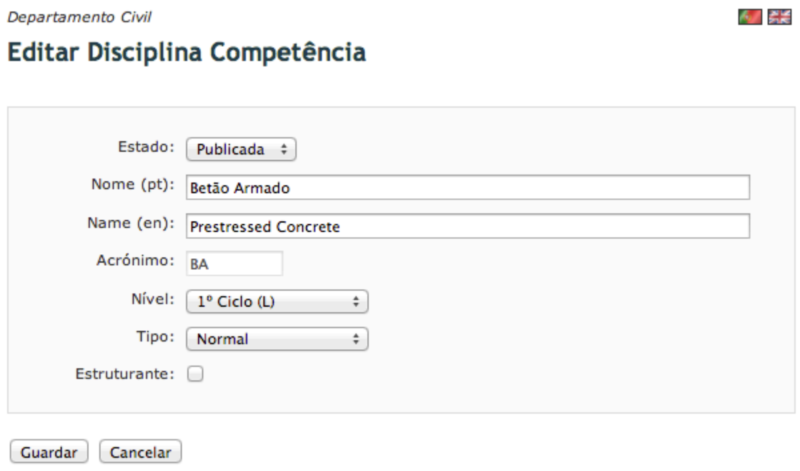
Concrete

- Concrete Group — Create Course

Betão Armado (BA) *Draft* — Show, Edit, Delete, Bibliographic References

Figure 3.15: Competence Course Created

You should be presented with a form as in Image 3.16. You should just change **Estado** from **Rascunho** to **Publicada** and hit **Submit**.



Departamento Civil

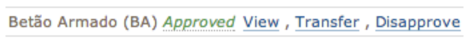
Editar Disciplina Competência

Estado:	Publicada
Nome (pt):	Betão Armado
Name (en):	Prestressed Concrete
Acrónimo:	BA
Nível:	1º Ciclo (L)
Tipo:	Normal
Estruturante:	☐

Figure 3.16: Publish Competence Course

Approve Competence Course

The last step is to approve the competence course. This is done by going to **Scientific Council > Competence Courses**. On the competence course you've just created, click in the operation **Approve**. The state should change to a green *Approved* like in the Image 3.17.



Betão Armado (BA) *Approved* View , Transfer , Disapprove

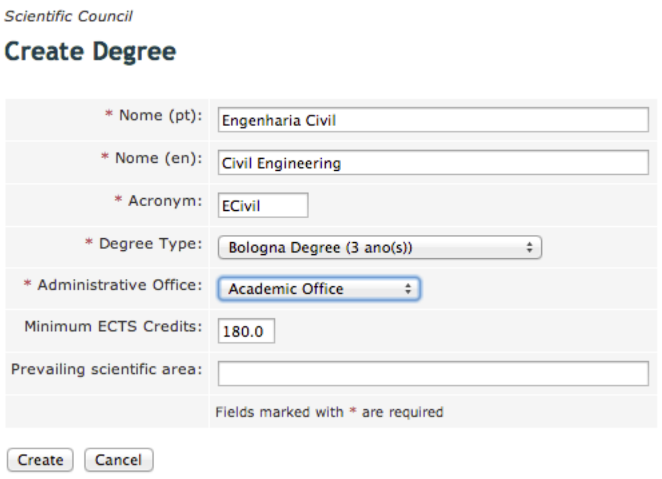
Figure 3.17: Competence Course Approved

3.2.8 Degrees and Curricular Plans

We have now enough data to start creating a degree.

Create a Degree

We now can create a degree. Go to **Scientific Council > Degrees and Curricular Plans** and then click **Criar Curso**. Fill the form with the same information as in Image 3.18. To learn more about Degrees check Chapter 4.



Scientific Council

Create Degree

* Nome (pt):	Engenharia Civil
* Nome (en):	Civil Engineering
* Acronym:	ECivil
* Degree Type:	Bologna Degree (3 ano(s))
* Administrative Office:	Academic Office
Minimum ECTS Credits:	180.0
Prevailing scientific area:	
	Fields marked with * are required

Figure 3.18: Degree Creation

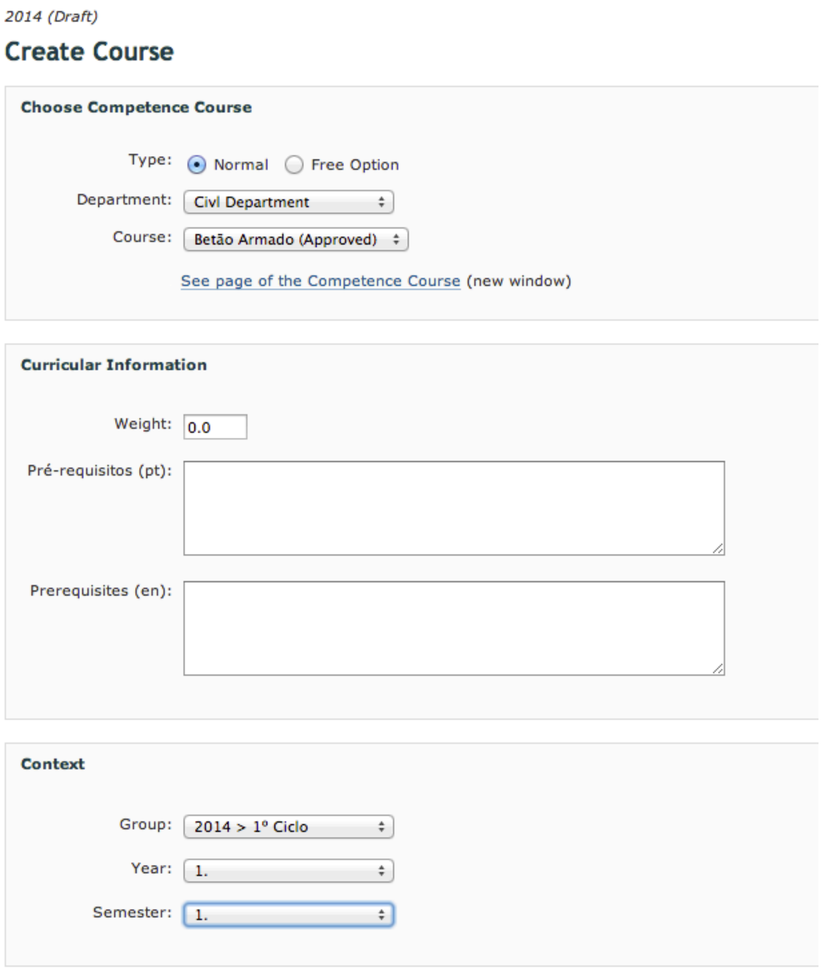
Create a Curricular Plans

Now is time to create a Curricular Plan. On the *Degree and Curricular Plan Management* click the operation **Create Plan**. Use 2014as the curricular plan name. To learn more about Curricular Plans check Section 4.4.

Change the Curricular Plans

We now need to actually start adding competence courses into this degree. Go to **Bolonha > View (Curricular Plan)**. The degree and curricular plans that we created should appear. Click in the operation **Gerir Plano**.

Next click the **Create course unit**, next to *First Cycle*. Fill the resulting form with the information in Image 3.19.



2014 (Draft)

Create Course

Choose Competence Course

Type: [● Normal] ○ Free Option

Department: Civl Department

Course: Betão Armado (Approved)

See page of the Competence Course (new window)

Curricular Information

Weight: 0.0

Pré-requisitos (pt):

Prerequisites (en):

Context

Group: 2014 > 1º Ciclo

Year: 1.

Semester: 1.

Figure 3.19: Creating Course Unit

Define Rules

Now we are going to define some rules. Rules are restrictions in the way the degree works. You can learn more about rules on Section 5.2.3. In the build Plan page, select **Define Rules**. Next select to define a rule in the *First Cycle*. Select **Nova Regra**. Insert the data as follows:

- **Rule Type:** Credits Limit
- **Credits:**
  - **Minimum:** 120.0
  - **Maximum:** 120.0

And hit **Submit**. A message bellow the *First Cycle* should appear saying that students are required to do 120 credits to be aprove at this course.

Approve the curricular plan

The final step is to approve this curricular plan. To do this is to we need to go the **Scientific Council > Degrees and Curricular Plans**.

Here we are going to select edit our current curricular plan *2014* and change it from Draft to Approved. Click the operation **Edit** next to the *2014*.

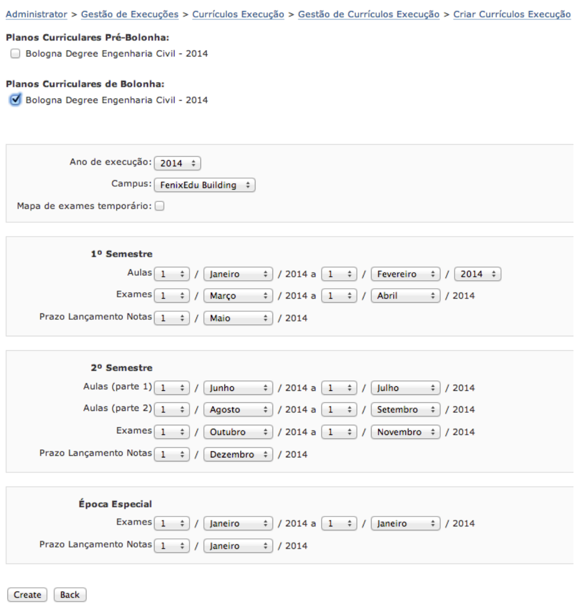
In *Edit Curricular Plan* just change the **Construction state** from *Draft* to *Approved* and click **Submit**. A message should appear saying that curricular plan changed successfully.

3.2.9 Executing Courses and Curricular Plans

Until now we've been defining our degree structure in a abstract sense, since they are not bounded to years yet. Now that we have a degree ready, we can now execute it. To learn more about execution courses check 5.3

Executing the Curricular Plans

Go to **Administrator > Currículos Execução > Criar Currículos Execução**. Select **Bolonha Degree** (the same we configured the Course with). A form should appear to set the the intervals where this course should run. For now you should fill it with the same information as in Figure 3.20.



Administrator > Gestão de Execuções > Currículos Execução > Gestão de Currículos Execução > Criar Currículos Execução

Planos Curriculares Pré-Bolonha:
☐ Bologna Degree Engenharia Civil - 2014

Planos Curriculares de Bolonha:
✓ Bologna Degree Engenharia Civil - 2014

Ano de execução: 2014
Campus: FenixEdu Building
Mapa de exames temporário: ☐

1º Semestre

Aulas	1 / Janeiro / 2014 a 1 / Fevereiro / 2014
Exames	1 / Março / 2014 a 1 / Abril / 2014
Prazo Lançamento Notas	1 / Maio / 2014

2º Semestre

Aulas (parte 1)	1 / Junho / 2014 a 1 / Julho / 2014
Aulas (parte 2)	1 / Agosto / 2014 a 1 / Setembro / 2014
Exames	1 / Outubro / 2014 a 1 / Novembro / 2014
Prazo Lançamento Notas	1 / Dezembro / 2014

Época Especial

Exames	1 / Janeiro / 2014 a 1 / Janeiro / 2014
Prazo Lançamento Notas	1 / Janeiro / 2014

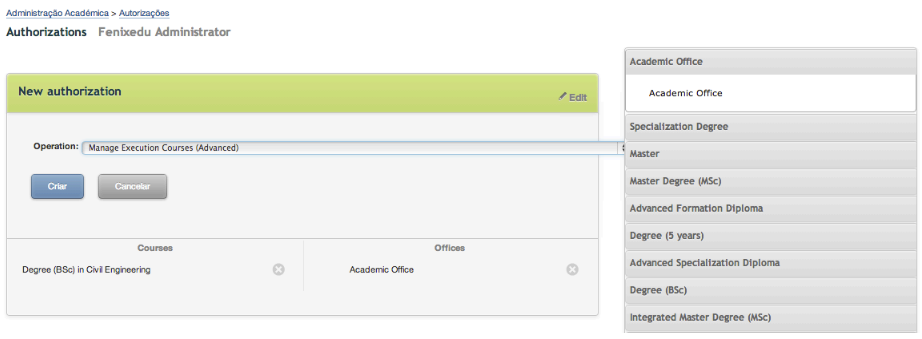
Figure 3.20: Executing the Curricular Plan

After entering all the information, A message should appear saying what curricular plans where correctly created.

Authorizações Académicas

In order to create execution courses we are required to have been granted to do so. Head to **Administração Académica > Autorizações**, click **manage** on the user.

To create a new Authorization just click in the **Criar nova Authorização**. Next, select the **Operation** and then drag into the **Offices** slot from the right **Academic Office > Academic Office** and to the Courses slot drag **Degree (BSc) > Engenharia Civil**. When you are done hit **Criar**. Image 3.21 shows a correctly filled authorization.


Administração Académica > Autorizações

Authorizations Fenixedu Administrator

New authorization

Operation: Manage Execution Courses (Advanced)

Courses	Offices
Degree (BSc) in Civil Engineering	Academic Office

Figure 3.21: Creating Authorization

You need to give yourself authorization for the following operations:

- Manage Execution Courses
- Manage Execution Courses (Adv)

We are going to need these later, so might as well add them right now.

- Matricular Aluno
- Inscrever Aluno
- View full Student Curriculum

Create Execution Courses for the Active Curricular Plans

After giving yourself the previous authorizations, a new set o operations should have appeared on the left menu, titled **Gestão das disciplinas de execução**. From that menu select **Criar Disciplinas de Execução**. Next select **Degree (BSc)**.

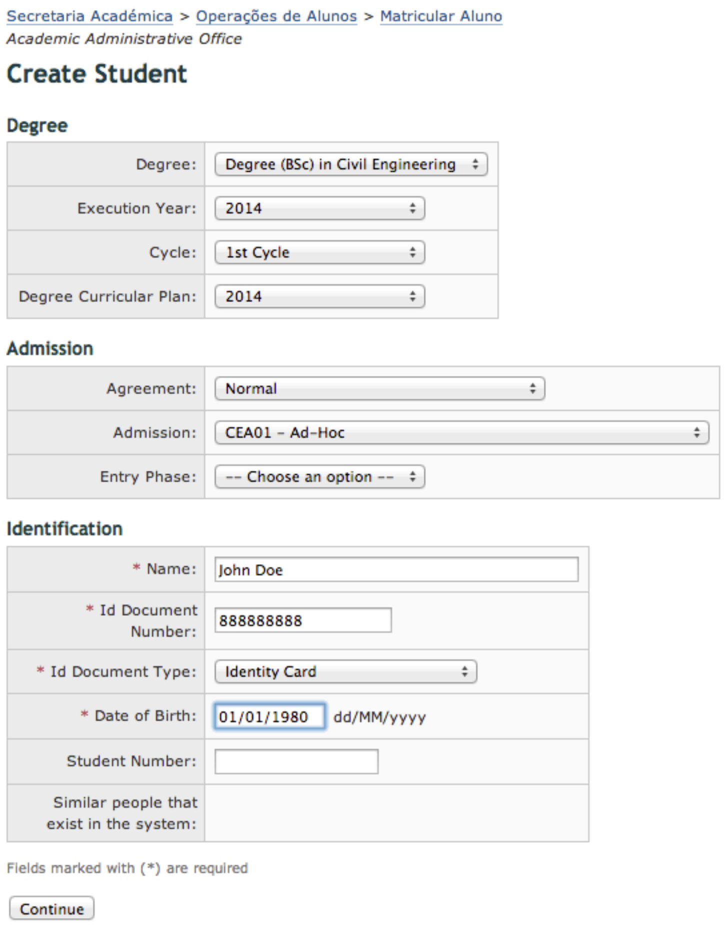
On the following screen select *First Semester - 2014* and select the curricular plan *Civil Engineering - 2014*. A message should appear saying the execution courses were successfully created.

3.2.10 Student Enrolment

Now our *Example Engineering School* has a degree being executed. We can now enrol a student into our school and enrol him in this degree.

Enrol a Student

To enrol a student just go to **Secretaria Académica > Matricular Aluno**. On the first form you will need to fill it with information the initial information about the student. Fill it with the information from Image 3.22.



Secretaria Académica > Operações de Alunos > Matricular Aluno
Academic Administrative Office

Create Student

Degree

Degree:	Degree (BSc) in Civil Engineering
Execution Year:	2014
Cycle:	1st Cycle
Degree Curricular Plan:	2014

Admission

Agreement:	Normal
Admission:	CEA01 - Ad-Hoc
Entry Phase:	-- Choose an option --

Identification

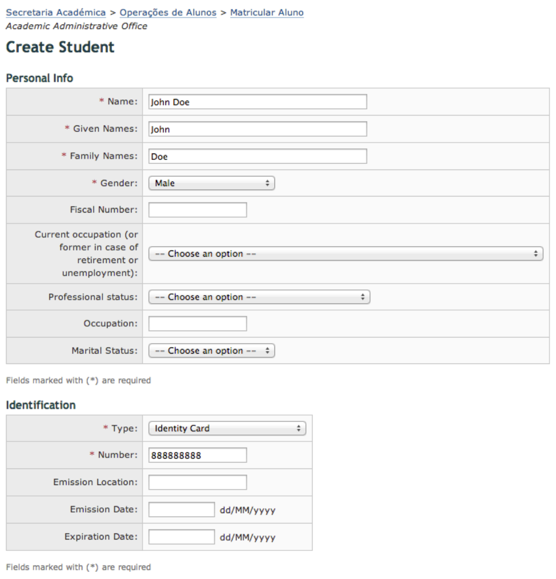
* Name:	John Doe
* Id Document Number:	888888888
* Id Document Type:	Identity Card
* Date of Birth:	01/01/1980 dd/MM/yyyy
Student Number:	
Similar people that exist in the system:	

Fields marked with (*) are required

Figure 3.22: First Form

After submitting a second form, even bigger should appear, with more detailed information. Fill it with information from Images 3.23, 3.24 and 3.25.

Attention:
In the *Previous completed degree* section of the form you are supposed to use *Portugal*. When you start writing it a drop down should appear. Select *Portugal* from that drop down.


Secretaria Académica > Operações de Alunos > Matricular Aluno
Academic Administrative Office

Create Student

Personal Info

* Name:	John Doe
* Given Names:	John
* Family Names:	Doe
* Gender:	Male
Fiscal Number:	
Current occupation (or former in case of retirement or unemployment):	-- Choose an option --
Professional status:	-- Choose an option --
Occupation:	
Marital Status:	-- Choose an option --

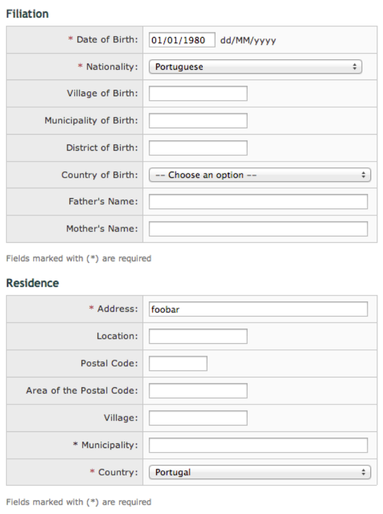
Fields marked with (*) are required

Identification

* Type:	Identity Card
* Number:	888888888
Emission Location:	
Emission Date:	dd/MM/yyyy
Expiration Date:	dd/MM/yyyy

Fields marked with (*) are required

Figure 3.23: First part of the Second Form


Filiation

* Date of Birth:	01/01/1980 dd/MM/yyyy
* Nationality:	Portuguese
Village of Birth:	
Municipality of Birth:	
District of Birth:	
Country of Birth:	-- Choose an option --
Father's Name:	
Mother's Name:	

Fields marked with (*) are required

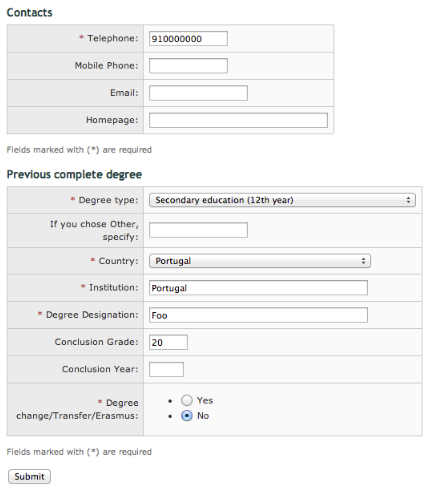
Residence

* Address:	foobar
Location:	
Postal Code:	
Area of the Postal Code:	
Village:	
* Municipality:	
* Country:	Portugal

Fields marked with (*) are required

Figure 3.24: Second part of the Second Form

A third form should appear for you to insert information about the parents qualifications. Fill this with information from Image 3.26



Contacts

* Telephone:	910000000
Mobile Phone:	
Email:	
Homepage:	

Fields marked with (*) are required

Previous complete degree

* Degree type:	Secondary education (12th year)
If you chose Other, specify:	
* Country:	Portugal
* Institution:	Portugal
* Degree Designation:	Foo
Conclusion Grade:	20
Conclusion Year:	
* Degree change/Transfer/Erasmus:	Yes [✓ No]

Fields marked with (*) are required

Figure 3.25: Third part of the Second Form

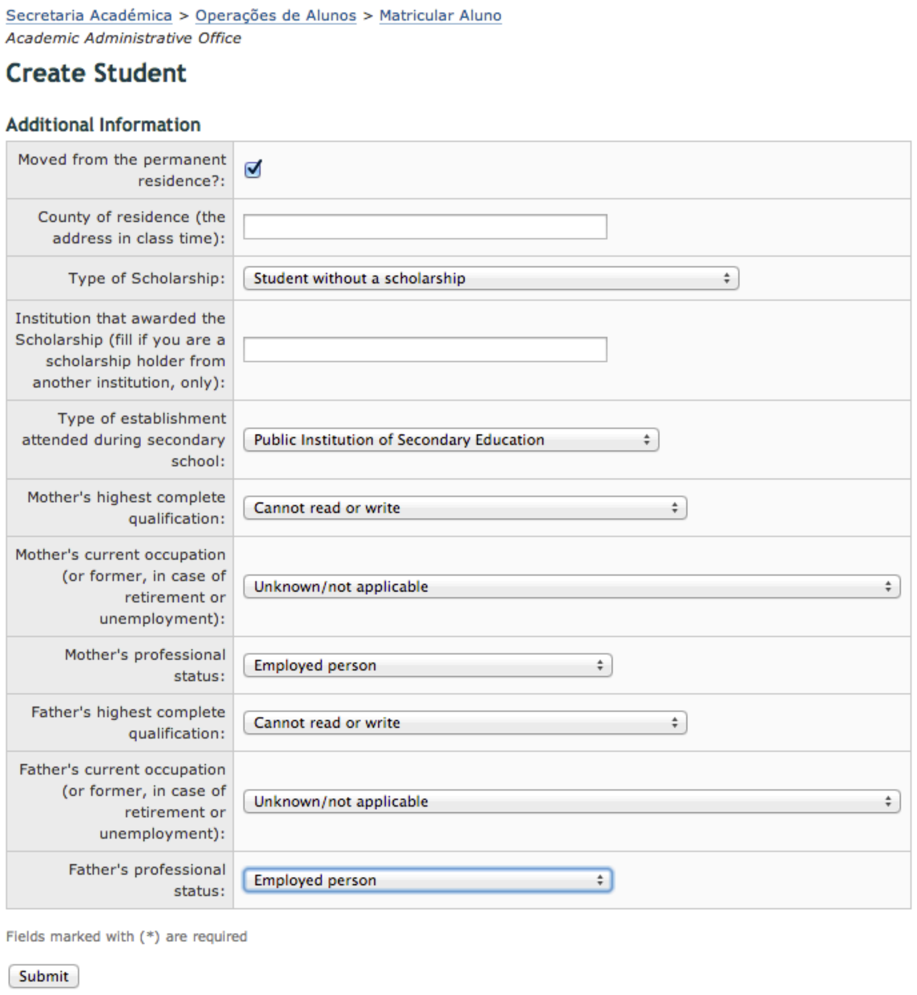
Finally a form should appear asking you to confirm the data you just inserted. Just hit **submit**.

Enrol a Student into Courses

At this moment we have *John Doe* enroled in our school on the degree we've created. However he is not enrolled in any courses. We now should add enrol this student into courses. First go to **Secretaria Académica > Visualizar Alunos** and in the *Number* use **1** (student numbers are attributed sequentially so the first student to be enrolled is number 1) and hit **Submit**.

The student page should appear and in registrations we can see that although he is studying in *Degree (BSc) in Civil Engineering* he has 0 enrolments this year. Lets change that. Click **See Process »**.

In the *Enrolment Process* look at *Curricular Plans for the Student* and select **Enrolment Management**. Next in *Enrolment in Courses* select **Enrolment in Courses**. In *Course Enrolment (with rules)* you should select what courses you want to enrol the student. In this case, we only have one. Select the checkbox to the right of *Prestressed Concrete* and hit **Save**. Our student is now enrolled.



Secretaria Académica > Operações de Alunos > Matricular Aluno
Academic Administrative Office

Create Student

Additional Information

Moved from the permanent residence?:	✓
County of residence (the address in class time):	
Type of Scholarship:	Student without a scholarship
Institution that awarded the Scholarship (fill if you are a scholarship holder from another institution, only):	
Type of establishment attended during secondary school:	Public Institution of Secondary Education
Mother's highest complete qualification:	Cannot read or write
Mother's current occupation (or former, in case of retirement or unemployment):	Unknown/not applicable
Mother's professional status:	Employed person
Father's highest complete qualification:	Cannot read or write
Father's current occupation (or former, in case of retirement or unemployment):	Unknown/not applicable
Father's professional status:	Employed person

Fields marked with (*) are required

Figure 3.26: Third Form

4 Degrees

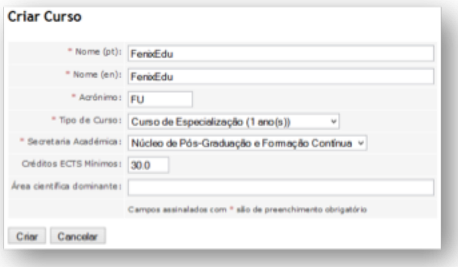
FenixEdu suports the creation of the following degree types *Bachelors*, *Masters*, *Doctorate*, *Advanced Formation* and *Advanced Specialization*.

To create Degrees in FenixEdu is in Scientific Council. This operation is made through the **Scientific Council Portal > Degrees** and **Curricular Plans > Create Degree**.

4.1 Create Degree

In the creation form of new degrees, as shown in Image 4.1, all fields marked with (*) are mandatory.

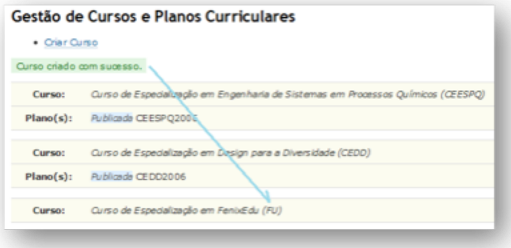
Depending on the *Degree Type* the system will fill the *Minimum ECTS Credits* field. The value can be ajusted.


Criar Curso

* Nome (pt):	FenixEdu
* Nome (en):	FenixEdu
* Acrónimo:	FU
* Tipo de Curso:	Curso de Especialização (1 ano(s))
* Secretaria Académica:	Núcleo de Pós-Graduação e Formação Contínua
Créditos ECTS Mínimos:	30.0
Área científica dominante:	
	Campos assinalados com * são de preenchimento obrigatório

Figure 4.1: Creating a Course

When the fields are filled in properly click the **Create** button. A message is shown indicating the successfull creation of the degree, as seen in Image 4.2.



Gestão de Cursos e Planos Curriculares

- Criar Curso

Curso criado com sucesso.

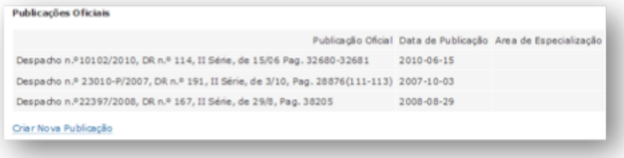
Curso:	*Curso de Especialização em Engenharia de Sistemas em Processos Químicos (CEESPQ)*
Plano(s):	Publicado CEESPQ2006
Curso:	*Curso de Especialização em Design para a Diversidade (CEDD)*
Plano(s):	Publicado CEDD2006
Curso:	*Curso de Especialização em FenixEdu (FU)*

Figure 4.2: Course Created

4.2 Edit a Degree

It allows updates to the information inserted on the Degree creation (Degree name, both in english and portuguese). This information can only be updated for the current academic year.

In the Edit option it's also available the possibility of inserting *Oficial Publications*, meaning that every time the Degree is published in the Official Laws Decret Journal (when changes to the Curricular Plan occur) it must be introduced the publication through **Create New Publication**, like seen in image 4.3.



Publicações Oficiais

Publicação Oficial	Data de Publicação	Area de Especialização
Despacho n.º10102/2010, DR n.º 114, II Série, de 15/06 Pag. 32680-32681	2010-06-15	
Despacho n.º 23010-P/2007, DR n.º 191, II Série, de 3/10, Pag. 28876(111-113)	2007-10-03	
Despacho n.º22397/2008, DR n.º 167, II Série, de 29/8, Pag. 38205	2008-08-29	

Criar Nova Publicação

Figure 4.3: Oficial Publications

This information must be always updated, since it's used in official documents like the Diploma Supplement.

4.3 Remove a Degree

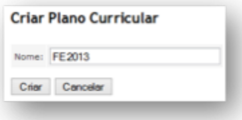
Allows that the Degree is deleted (as long as it does not have any associated information, like a Curricular Plan).

4.4 Curricular Plans

A Degree can have more than one Curricular Plan. If there are substancial changes to a Curricular Plan, a new plan should be created.

4.4.1 Create Curricular Plan

Allows the creation of Curricular Plans associated to the Degree. To create a Curricular Plan, we must give it a name (an acronym, as shown below in image 4.4).

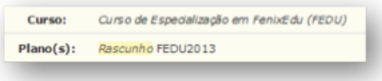


Criar Plano Curricular

Nome: FE2013

Figure 4.4: Creating a curricular plan

The new Curricular Plans state is *Draft*, meaning that they can not be seen in the other Portals in the system, nor can they be used para perform any action (like enroll students). In Image 4.5 we can see how they look.



Curso:	*Curso de Especialização em FenixEdu (FEDU)*
Plano(s):	*Rascunho* FEDU2013

Figure 4.5: A curricular plan

4.4.2 Managing Curricular Plans

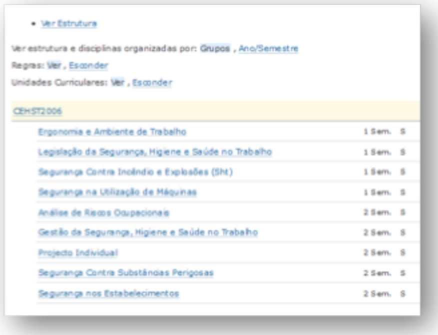
The Curricular Plans are managed through different Portals. We're goint to describe which operations can be made on the Curricular Plans in the different Portals.

4.4.3 Managing Curricular Plans on Scientific Council

The following operations are avaible through Cientific Council:

View

Through this option you can view the information of the Curricular Plan. Beside the courses (that can be viewed by group or semester/year) that are in the Curricular Plan, the rules (of enrolment in courses) and the structure (groups) of the Curricular Plan can also be consulted, as the Image 4.6 ilustrates.



- Ver Estrutura

Ver estrutura e disciplinas organizadas por: Grupos , Ano/Semestre

Regras: Ver , Esconder

Unidades Curriculares: Ver , Esconder

CEHST2006

Ergonomia e Ambiente de Trabalho	1 Sem.	S
Legislação da Segurança, Higiene e Saúde no Trabalho	1 Sem.	S
Segurança Contra Incêndio e Explosões (Sht)	1 Sem.	S
Segurança na Utilização de Máquinas	1 Sem.	S
Análise de Riscos Ocupacionais	2 Sem.	S
Gestão da Segurança, Higiene e Saúde no Trabalho	2 Sem.	S
Projecto Individual	2 Sem.	S
Segurança Contra Substâncias Perigosas	2 Sem.	S
Segurança nos Estabelecimentos	2 Sem.	S

Figure 4.6: Viewing a curricular plan

Edit

This option allows you to:

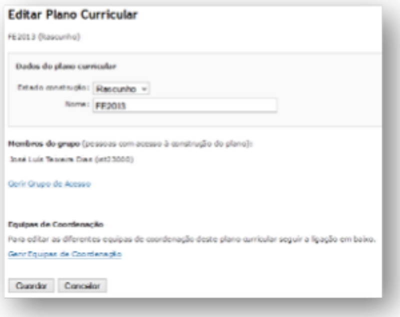
- **Curricular Plan State** - you can see the current state of the Curricular Plan and the Curricular Plan name (it can be edited).
  - *Draft* - it's the initial state of the Curricular Plan.
  - *Published* - it means the Curricular Plan is almost finished. This middle state is usefull so that some final adjustments can be done. This state no longer allows that the Plan be edited in the Bologna Portal (it can only be updated in the Academic Administration Portal, or the Scientific Council has to change the state back to *Draft*).
  - *Approved* - it's the final state in which all the courses and rules are defined. Any update to the Curricular Plan in this state can only be done in the Academic Administration (the Approved state can not be changed by the Scientific Council, so changing the state to Approved is a very sensitive operation).
- **Manage Access Group** - Where you can add or remove members of the group of people that are able to participate in the definition of the Curricular Plan.
- **Manage Coordination Teams** - Where you can add or remove members of the Degree Coordination team, no the academic years in which the Degree had/has an execution.

This options are summed up in the next Image 4.7.

Delete

The system only allows to delete Curricular Plans that still don't have any associated courses. To delete a Curricular Plan, click on Delete and on the next page click the **Confirm** button.

A confirmation popup box is displayed to ensure that you really want to delete the Curricular Plan. Click on **OK** to delete or **Cancel** to cancel the operation, like shown on image 4.8.



Editar Plano Curricular

FE2013 (Rascunho)

Dados do plano curricular

Estado construção: Rascunho

Nome: FE2013

Membros do grupo (pessoas com acesso à construção do plano):

José Luís Teixeira Dias (ist23000)

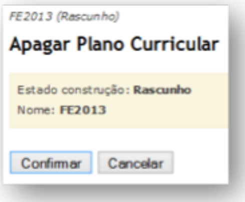
Gerir Grupo de Acesso

Equipas de Coordenação

Para editar as diferentes equipas de coordenação deste plano curricular seguir a ligação em baixo.

Gerir Equipas de Coordenação

Figure 4.7: Editing a curricular plan



FE2013 (Rascunho)

Apagar Plano Curricular

Estado construção: **Rascunho**

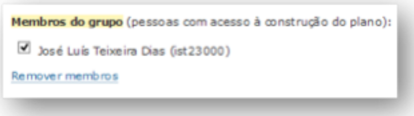
Nome: **FE2013**

Figure 4.8: Delete a curricular plan

Group

It allows to add or remove members of the group of people with access to define a Curricular Plan (it's done in the Bologna Portal).

- Remove Members, select the member or members to remove from the group and click on **Remove Members**, as in the image 4.9.

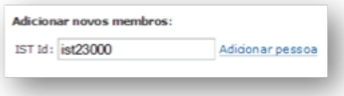


Membros do grupo (pessoas com acesso à construção do plano):

☑ José Luís Teixeira Dias (ist23000)

Figure 4.9: Remove a member plan

- Add Person, adds persons to the group of people that can define a Curricular Plan. Insert the person username and click **Add** person, like in the Image 4.10.



ist23000

Figure 4.10: Add a member

4.4.4 Manage Curricular Plans (Bologna Portal)

It's through this portal that the Curricular Plans are defined, created by the Cientific Council. To manage the Curricular Plan click on **View** under the *Curricular Plans* section.

All the existing Curricular Plans are listed. The funcionality Manage Plan will be available if you are part of the authorized group.

5

Courses

In FenixEdu the courses are represented in three types. We will describe the various types of existing courses, which information is stored and managed in each of these types and the relationship between the various courses:

- **Competence Courses** - Are the basis of information relating to curricular units. The management of these courses is the responsibility of the Department where it belongs (competence courses can only belong to a single department). The information in a racing discipline is:
  - Workload
  - Credits
  - Objectives
  - Program
  - Evaluation Method
  - Base Bibliography (Primary e Secundary)
- **Curricular Courses** – This is the association between a *Competence Course* and a *Curricular Plan*. For a Course to used in a Curricular Plan, it must be approved before. The year and semester where this curricular plan is offered is determernined by the curricular plan. This contains the following information.
  - Group
  - Curricular Year
  - Semseter
  - Begining
  - End

- **Execution Courses** - This is the association between the academic year and semester in which the *Curricular Courses* was in operation. You can only create execution courses when the Curriculum Plan where they are located is also running. The information contained in execution is a discipline as follows:
  - Course Site
  - Schedule
  - Sumaries
  - Announcements

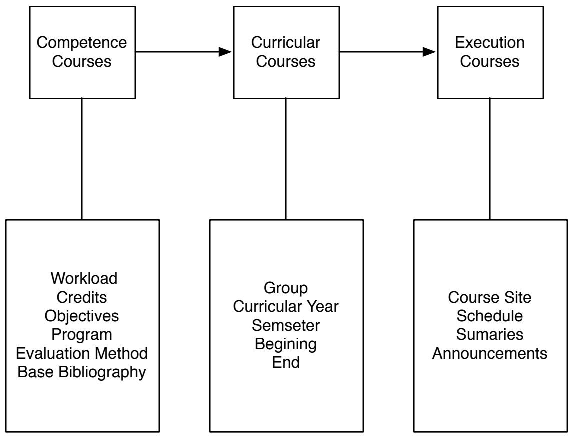


Figure 5.1: Course types and their relationship within FenixEdu

5.1 Competence Course

The responsability to manage courses belongs to the Departments (creating courses or new versions of them) and the Scientific Council (Approving new competence courses and their versions). As mentioned before they are basis for creating curricular units, and only after being approved can they be used in curricular plans.

5.1.1 Creating Competence Courses

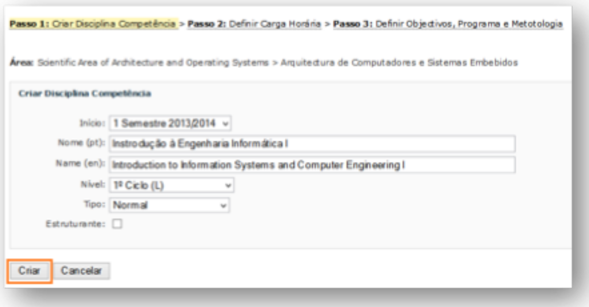
Before you can create competence courses you need to have authorization to do so. This is granted in **Secretaria do Departamento> "Grupos de Gestão "> Disciplinas Competência**.

To create a competence course, go to **Bolonha > Disciplinas Competência> Consultar**. It should appear the existent competence courses of that department. Each course is attached to a *Scientific Area*, and a *Scientific Area* is composed of *Competence Course Groups*. Click in **Criar Disciplina** on the *Scientific Area/Competence Course* Group you want to create a course. It takes 3 steps.

Step One: Creating the Competence Course

Fill the academic year and semester, from which the course starts being available on the system, the name (in PT and EN), the level (First Cycle, Second Cycle, etc.) and the type of course (Normal or Dissertation) like shown in Image 5.2.

Click Create and move to the second step.



Passo 1: Criar Disciplina Competência > **Passo 2:** Definir Carga Horária > **Passo 3:** Definir Objectivos, Programa e Metotologia

Área: Scientific Area of Architecture and Operating Systems > Arquitectura de Computadores e Sistemas Embebidos

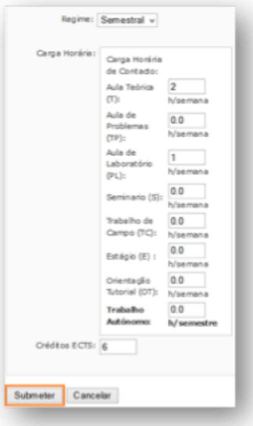
Criar Disciplina Competência

Inicio:	1 Semestre 2013/2014
Nome (pt):	Instrodução à Engenharia Informática I
Name (en):	Introduction to Information Systems and Computer Engineering I
Nivel:	1º Ciclo (L)
Tipo:	Normal
Estruturante:	☐

Figure 5.2: Creating the Competence Course

Steop Two: Defining Workload

On this screen fill the information relative to the workload (Theorical,Problems, etc.) and the ECTS. After filling this form, click **Submit** and move to the next step.



Regime: Semestral

Carga Horária:

Carga Horária de Contacto:

Aula Teórica (T):	2	h/semana
Aula de Problemas (TP):	0.0	h/semana
Aula de Laboratório (PL):	1	h/semana
Seminario (S):	0.0	h/semana
Trabalho de Campo (TC):	0.0	h/semana
Estágio (E) :	0.0	h/semana
Orientação Tutorial (OT):	0.0	h/semana
Trabalho Autónomo:	0.0	**h/semestre**

Créditos ECTS: 6

Figure 5.3: Defining Workload

Step Three: Define Objectives Program and Methodology

Fill with information regarding objectives, program and methodology for this competence course.

Click **Submit** to finally create the competence course.

Afther adding the coures it should appear within the department competence course list, in the *Scientific Area/Competence Course* you selected, as is shown in Image 5.4

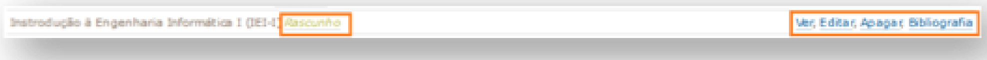

Introdução à Engenharia Informática I (IEI-I) Rascunho Ver, Editar, Apagar, Bibliografia

Figure 5.4: Created Competence Course

5.1.2 Managing Competence Courses

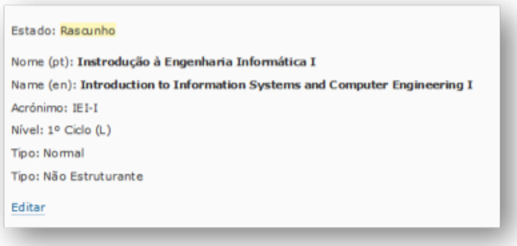
A competence course can be in one of two states.

- **Draft** - This is the initial state. In this state the course is still in construction in the department and is only visible to the group of people who can access manage competence courses in this departments.
- **Published** - The discipline has all the information and is ready to approved by the Scientific Council.

Also the competence courses that weren't approved by the Scientific Council has the following avaible options:

- **View** - View all the information avaible.
- **Edit** - Edit the information of for that competence course. Is through the Edit operation that the state can be updated from *Draft* to *Publish*. This state can be changed until the competence course is approved by the Scientific Council. After that, the course can only be changed on the *Version Manager* (see Section 5.1.5).
- **Delete** - Deletes the competence course (as long as it hasn't been approved). *This is irreversible.*
- **Bibliografia** - Adding the base bibliografy (principal and secondary).

  To add bibliografic references, click *Insert Bibliografic Reference* and fill the form with the blibliografic information as well the type of bibliography as shown in Image 5.6.



Estado: Rascunho

Nome (pt): **Introdução à Engenharia Informática I**

Name (en): **Introduction to Information Systems and Computer Engineering I**

Acrónimo: IEI-I

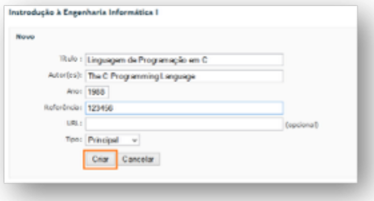
Nível: 1º Ciclo (L)

Tipo: Normal

Tipo: Não Estruturante

Editar

Figure 5.5: Changing Competence Course State



Introdução à Engenharia Informática I

Novo

Título :	Linguagem de Programação em C
Autor(es):	The C Programming Language
Ano:	1988
Referência:	123456
URL :	(opcional)
Tipo:	Principal

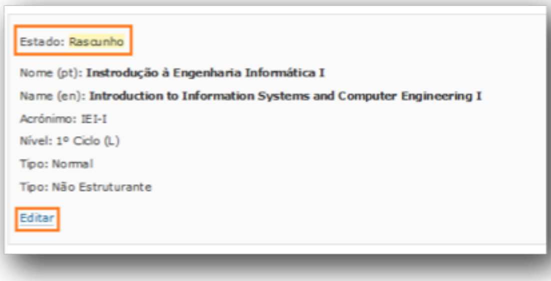
Criar Cancelar

Figure 5.6: Adding Bibliographic References

5.1.3 Publishing Competence Courses

When all the information for a new competence course has been added, you need to published it for it to be avaible on the Scientific Council so that it can be approved.

Edit the course, and next edit the state information as shown on Image 5.7.



Estado: Rascunho

Nome (pt): **Instrodução à Engenharia Informática I**

Name (en): **Introduction to Information Systems and Computer Engineering I**

Acrónimo: IEI-I

Nível: 1º Ciclo (L)

Tipo: Normal

Tipo: Não Estruturante

Editar

Figure 5.7: Editing competence course state

5.1.4 Approve Competence Courses

To approve competence courses you should go to **Portal do Conselho Científico> "Processo de Bolonha"> Disciplinas Competência** and select the department you want to approve competence courses.

There it should appear all the competence courses for that department. Click the *Approve* link next to the competence course. A green label next to the competence should appear as shown on Image 5.8

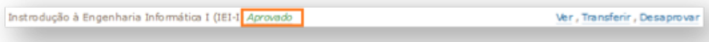

Instrodução à Engenharia Informática I (IEI-I Aprovado Ver , Transferir , Desaprovar

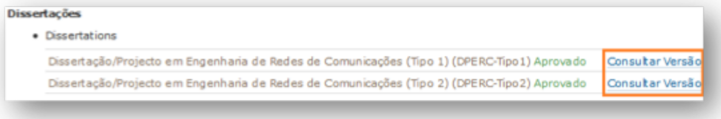
Figure 5.8: Approving a competence course

In Scientific Council you can do the following following actions:

- **View** - Show all the information of a competence course (Workload, Objectives, Program).
- **Transfer** - Transfer a competence course to another Department, Scientific Area or Competence Course Group.
- **Unprove** - Revert the competence course from *Approved* to *Published*

5.1.5 Competence Course Versions

Sometimes is necessary to make some corrections to curricular units. For instance changing the ECTS number from 6 to 4.5. Creating versions of a competence course is done in the Department where the course belongs. To make a new version go to **Disciplinas Competência > Gestão de Versões**. A list of the approved competence courses should appear like in Image 5.9.


Dissertações

- Dissertations

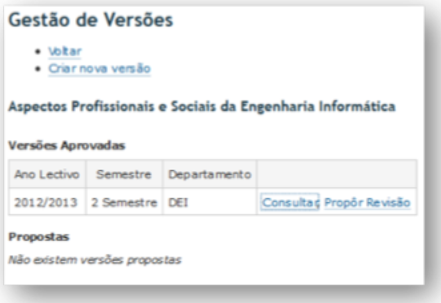
Dissertação/Projecto em Engenharia de Redes de Comunicações (Tipo 1) (DPERC-Tipo1) Aprovado Consultar Versão
Dissertação/Projecto em Engenharia de Redes de Comunicações (Tipo 2) (DPERC-Tipo2) Aprovado Consultar Versão

Figure 5.9: Competence course versions

To view the details from a competence course click **Consultar**. A interafce should appear that allows you to view current versions and create new ones, like as shown on Image 5.10.

Create a new version

To create a new version click **Criar nova Versão**. In the next page you should select the academic year and semester where this new version should appear (the system does not allow overlapping versions). Add also the information about the competence course and fill the *Proposal Reason*. Check Section 5.1.1 for more information.



Gestão de Versões

- Voltar
- Criar nova versão

Aspectos Profissionais e Sociais da Engenharia Informática

Versões Aprovadas

Ano Lectivo	Semestre	Departamento	
2012/2013	2 Semestre	DEI	Consultar Propôr Revisão

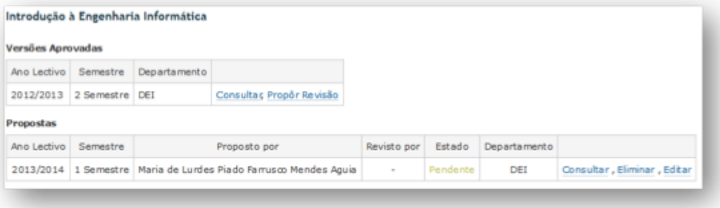
Propostas

Não existem versões propostas

Figure 5.10: Competence course versions

After completing this information, click **submit**. In the next page you can update the base bibliography for a give course. To conclude this process click **Create**.

After creating a version, a new *Proposal*, with a *Pending* state, should appear in the Version Page as in Image 5.11.



Introdução à Engenharia Informática

Versões Aprovadas

Ano Lectivo	Semestre	Departamento	
2012/2013	2 Semestre	DEI	Consultar Propôr Revisão

Propostas

Ano Lectivo	Semestre	Proposto por	Revisto por	Estado	Departamento	
2013/2014	1 Semestre	Maria de Lurdes Piado Farrusco Mendes Aguia	-	Pendente	DEI	Consultar , Eliminar , Editar

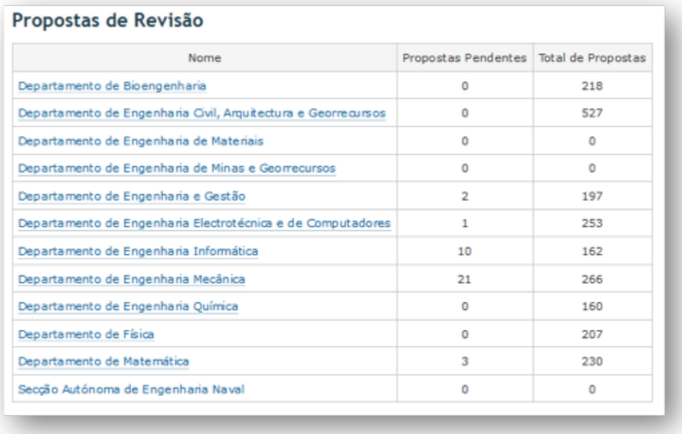
Figure 5.11: A new Version

Approve a competece course version

To approve competece courses go to **Portal do Conselho Científico > Processo de Bolonha** and click **Propostas de Versões**. A list of Proposals, pending and overall, for each department as shown on Image 5.12.

To approve a proposal you should, click on the department containg that version, and then click Approve.

After a new version is approved, all students enroled before the academic year and semester selected will be associated with the previous version, and new enrolments will use the new version.


Propostas de Revisão

Nome	Propostas Pendentes	Total de Propostas
Departamento de Bioengenharia	0	218
Departamento de Engenharia Civil, Arquitectura e Georrecursos	0	527
Departamento de Engenharia de Materiais	0	0
Departamento de Engenharia de Minas e Georrecursos	0	0
Departamento de Engenharia e Gestão	2	197
Departamento de Engenharia Electrotécnica e de Computadores	1	253
Departamento de Engenharia Informática	10	162
Departamento de Engenharia Mecânica	21	266
Departamento de Engenharia Química	0	160
Departamento de Física	0	207
Departamento de Matemática	3	230
Secção Autónoma de Engenharia Naval	0	0

Figure 5.12: Proposal Versions

Propose a revision to a version

Sometimes you may need to do some correction (maybe ECTS, workload, bibliography, etc.) in a approved version of a competence course. To do that on go to Version page and select a approved version and click **Propose Revision**. Fill all fields and click **Submit**.

A revision proposal stays in a pending state until approval is given by the Scientific Council.

5.2 Curricular Course

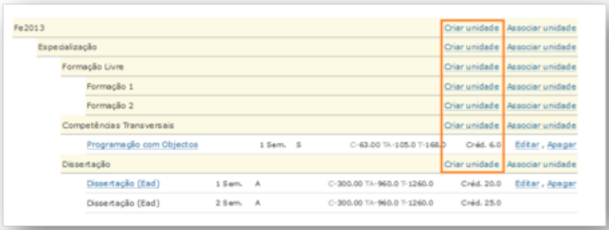
After approved, competence courses can be used in the construction of Curricular Plans. Curricular courses agregate several information about a course, for instance the curricular year and semester as well the curricular plan group it belongs.

5.2.1 Create a Curricular Unit

To create curricular units in curricular plains, go to **Portal da Administração Académica> "Gestão da Estrutura de Ensino"**, click **Estrutura de Cursos** and then in the operation **Gerir Plano** for the curricular plan you want to change.

Next click in **Criar Unidade**, in the curricular Plan where you want to add the new curricular unit like is shown on Image 5.13

On the next page, fill with information relative to the curricular unit and what is context (group, year, semester, begining and end). An image of the form can be seen on Image 5.14. There are two types of curricular units that can be added to the curricular plans.

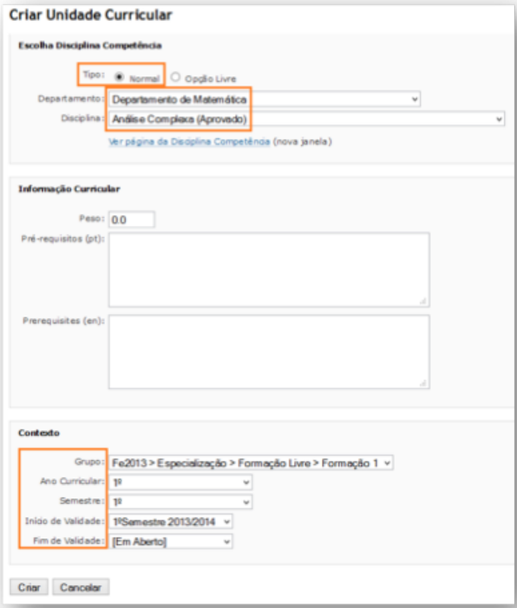


Fe2013					Criar unidade	Associar unidade
Especialização					Criar unidade	Associar unidade
Formação Livre					Criar unidade	Associar unidade
Formação 1					Criar unidade	Associar unidade
Formação 2					Criar unidade	Associar unidade
Competências Transversais					Criar unidade	Associar unidade
Programação com Objectos	1 Sem.	S	C-63.00 TA-105.0 T-168.0		Créd. 6.0	Editar , Apagar
Dissertação					Criar unidade	Associar unidade
Dissertação (Ead)	1 Sem.	A	C-300.00 TA-960.0 T-1260.0		Créd. 20.0	Editar , Apagar
Dissertação (Ead)	2 Sem.	A	C-300.00 TA-960.0 T-1260.0		Créd. 25.0	

Figure 5.13: Curricular Units

- **Normal Curricular Units** - There are curricular units that exist in Departments (competence courses).
- **Free Option Curricular Units** - Are slots that are created within a curricular plan that can be filled by other units from other curricular plans. This give the student the possibility to choose what course to do. In the case of **Free Options** you also need to add a name to the curricular unit.

After filling the form with all the information, click **Criar**.



Criar Unidade Curricular

Escolha Disciplina Competência

Tipo:	[✓ Normal] Opção Livre
Departamento:	Departamento de Matemática
Disciplina:	Análise Complexa (Aprovado)
	Ver página da Disciplina Competência (nova janela)

Informação Curricular

Peso:	0.0
Pré-requisitos (pt):	
Prerequisites (en):	

Contexto

Grupo:	Fe2013 > Especialização > Formação Livre > Formação 1
Ano Curricular:	1º
Semestre:	1º
Início de Validade:	1ºSemestre 2013/2014
Fim de Validade:	[Em Aberto]

Figure 5.14: Add a Curricular Unit

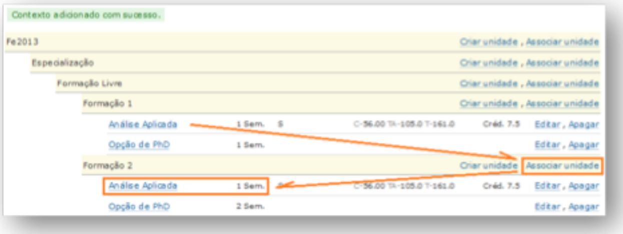
You can do the following operations over curricular units.

- **Edit** - This allows changing all the information for a curricular unit. You can also add a new context to a curricular unit.
- **Apagar** - Curricular units can be erased if they aren't associated with students.

5.2.2 Associate a Curricular Unit

Associate Unit allows that a curricular course can to be associated with more than one context. This allows for curricicular units to appear in differente semesters or years.

To do this, Click **Associate Unit** and fill with information related to the group you want. Finally finally click **Associate**. A curricular unit associated to two different contexts is shown on Image 5.15.



Contexto adicionado com sucesso.

Fe2013					Criar unidade , Associar unidade
Especialização					Criar unidade , Associar unidade
Formação Livre					Criar unidade , Associar unidade
Formação 1					Criar unidade , Associar unidade
Análise Aplicada	1 Sem.	5	C-56.00 TA-105.0 T-161.0	Créd. 7.5	Editar , Apagar
Opção de PhD	1 Sem.				Editar , Apagar
Formação 2					Criar unidade , Associar unidade
Análise Aplicada	1 Sem.		C-56.00 TA-105.0 T-161.0	Créd. 7.5	Editar , Apagar
Opção de PhD	2 Sem.				Editar , Apagar

Figure 5.15: Curricular unit in two different contexts

5.2.3 Defining Rules

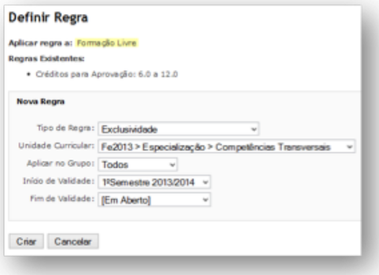
Curricular units and groups can have rules that affect how students can enrol in them. Among other, you can set:

- Requirements between curricular units.
- Mutual exclusion between groups and curricular units
- Enrolment only with the Degree Coordinator's Authorization
- Maximum of credits can a student can take on given group

To add a new rule, click **Define Rules**. Next on the group or curricular unit click **Define Rules**. Now you can choose, either to add a **New Rule** or **Compose Rules**.

New Rule

Depending on the type of rule (group or curricular unit) you may have several options. In Image 5.16 we are create a Exclusivity rule between the group *Formação Livre* and the group **Competencias Transversais**. This rule implies that students can only enrol in curricular units from one of the groups. The rules have a **Validity** associated so that rules can be inserted into different curricular plans.


Definir Regra

Aplicar regra a: Formação Livre

Regras Existentes:

- Créditos para Aprovação: 6.0 a 12.0

Nova Regra

Tipo de Regra:	Exclusividade
Unidade Curricular:	Fe2013 > Especialização > Competências Transversais
Aplicar no Grupo:	Todos
Início de Validade:	1ºSemestre 2013/2014
Fim de Validade:	[Em Aberto]

Figure 5.16: Creating a rule

When a rule is created it should appear in the curricular plan right bellow affected target.

Compose Rules

A rule composition allows that existing rules can be composed using logic operators & (AND) and || (OR). To add rule composition , access the Curricular Plan an in the group or curricular unit you want, click in **Define Rule**.

On the existing rules for that group appear. Click in **Compose Rule** and select from the existing rules those you want to use and the logic operator. Click **Create to finish**.

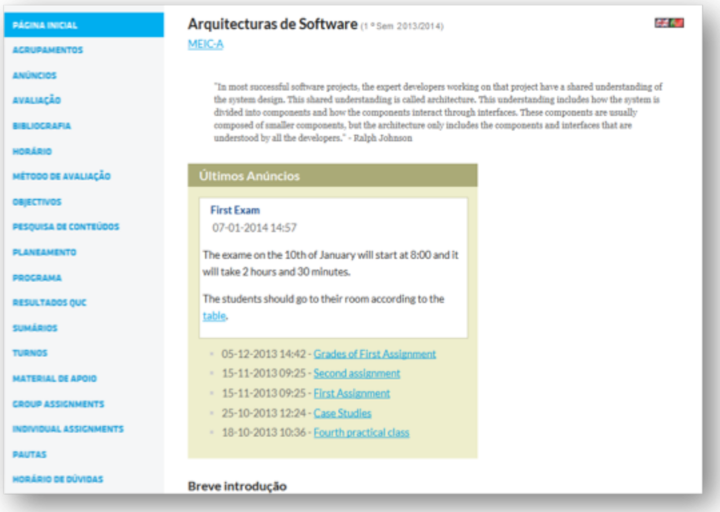
You can also **Edit** the begging or end of a validity and **Remove** the rule.

5.3 Execution Courses

Creating execution courses is task that is done every semester (or every course period). For every Curricular unit that are avaible in active curricular plans there should be created a execution course. Execution courses are managed in **Portal Administração Académica > Gestão das Disciplinas de Execução**.

Execution courses have all the information from the competence courses (ECTS, program, workload, bibliography) and from curricular courses (year and semester), as well the contents avaible in course site. Each execution course as also associated the following information:

- Course Site
- Faculty
- Atending Students
- Evaluations
- Schedule
- Sumaries
- Annoucements



Arquitecturas de Software (1 º Sem 2013/2014)

MEIC-A

"In most successful software projects, the expert developers working on that project have a shared understanding of the system design. This shared understanding is called architecture. This understanding includes how the system is divided into components and how the components interact through interfaces. These components are usually composed of smaller components, but the architecture only includes the components and interfaces that are understood by all the developers." - Ralph Johnson

Últimos Anúncios

First Exam
07-01-2014 14:57

The exame on the 10th of January will start at 8:00 and it will take 2 hours and 30 minutes.

The students should go to their room according to the table.

- 05-12-2013 14:42 - Grades of First Assignment
- 15-11-2013 09:25 - Second assignment
- 15-11-2013 09:25 - First Assignment
- 25-10-2013 12:24 - Case Studies
- 18-10-2013 10:36 - Fourth practical class

Breve introdução

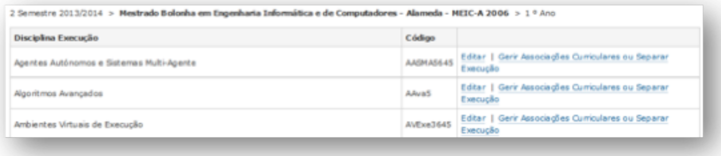
Figure 5.17: Course site

5.3.1 Creating Execution Coures

The action **Criar Disciplinas de Execução** allows you to create all the execution courses corresponding to one or more courses in execution in that execution period (year/semester). First select he type of course and click **Continue**. Next select the execution period, the courses you want to create the execution courses and click **Create**. In the next page a message with the information about all the execution courses created should appear.

5.3.2 Editing Execution Courses

Editar Disciplina Execução allows to **Edit** (including association and seperation of curricular courses) as well **Remove** for that execution period. First select the degree type. Next select he execution degree and curricular year from the execution course you wish to edit. Click **Continue**. A list with all execution courses for that degree/curricular year shoud appear.



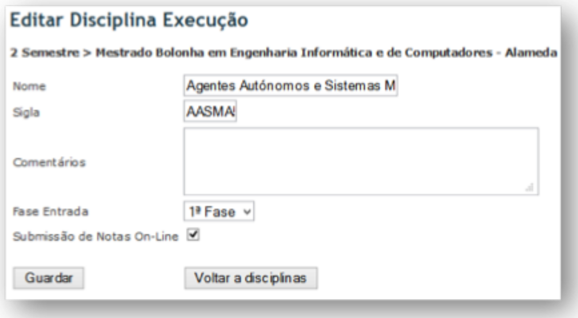
2 Semestre 2013/2014 > **Mestrado Bolonha em Engenharia Informática e de Computadores - Alameda - MEIC-A 2006** > 1 º Ano

Disciplina Execução	Código	
Agentes Autónomos e Sistemas Multi-Agente	AASMA5645	Editar | Gerir Associações Curriculares ou Separar Execução
Algoritmos Avançados	AAva5	Editar | Gerir Associações Curriculares ou Separar Execução
Ambientes Virtuais de Execução	AVExe3645	Editar | Gerir Associações Curriculares ou Separar Execução

Figure 5.18: List of execution courses

Edit

Allows editing the execution course information. Typically this operation is used to update the name or acronym.



Editar Disciplina Execução

2 Semestre > Mestrado Bolonha em Engenharia Informática e de Computadores - Alameda

Nome	Agentes Autónomos e Sistemas M
Sigla	AASMA
Comentários	
Fase Entrada	1ª Fase
Submissão de Notas On-Line	✓

Figure 5.19: Editing execution courses

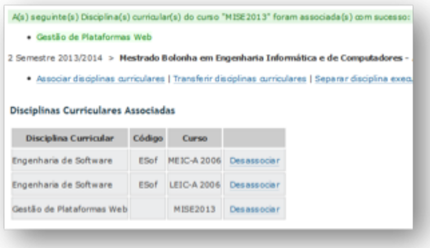
Gerir Associações Curriculares ou Separar Execução

Attention:
You should exercise extreme caution when doing these operations, as they interact with several information.

This functionality allows to do certain tasks that join or seperate execution courses. You have the following operations avaible:

- **Associate curricular coures**
  Allows binding more curricular courses to a execution courese. (that don't have yet a execution course). Select the degree where the curricular course belongs and press **Continue**. Next you have a list with all the associations already made as shown on Image 5.20

  To unbind a curricular course, just click **Disassociate**.



A(s) seguinte(s) Disciplina(s) curricular(s) do curso "MISE2013" foram associada(s) com sucesso:

- Gestão de Plataformas Web

2 Semestre 2013/2014 > **Mestrado Bolonha em Engenharia Informática e de Computadores -**

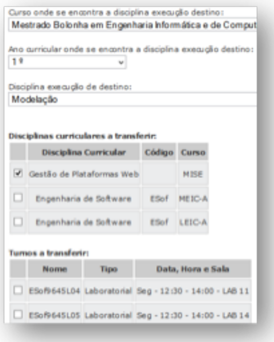
- Associar disciplinas curriculares | Transferir disciplinas curriculares | Separar disciplina exec

Disciplinas Curriculares Associadas

Disciplina Curricular	Código	Curso	
Engenharia de Software	ESof	MEIC-A 2006	Desassociar
Engenharia de Software	ESof	LEIC-A 2006	Desassociar
Gestão de Plataformas Web		MISE2013	Desassociar

Figure 5.20: Associate curricular courses

- **Transfer curricular coures**
  To transfer curricular courses to another execution course, select the degree, the execution course and the curricular year as shown on Image. You can also transfer schedules by checking the shifts you want to transfer.



Curso onde se encontra a disciplina execução destino:
Mestrado Bolonha em Engenharia Informática e de Comput

Ano curricular onde se encontra a disciplina execução destino:
1º

Disciplina execução de destino:
Modelação

Disciplinas curriculares a transferir:

	Disciplina Curricular	Código	Curso
✓	Gestão de Plataformas Web		MISE
☐	Engenharia de Software	ESof	MEIC-A
☐	Engenharia de Software	ESof	LEIC-A

Turnos a transferir:

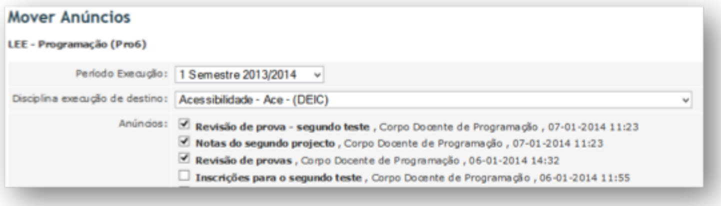
	Nome	Tipo	Data, Hora e Sala
☐	ESof9645L04	Laboratorial	Seg - 12:30 - 14:00 - LAB 11
☐	ESof9645L05	Laboratorial	Seg - 12:30 - 14:00 - LAB 14

Figure 5.21: Transfer curricular courses

- **Seperate execution course**
  If you wish to seperate curricular courses into new execution coureses click **Seperate**. Select the curricular courses and shifts you want to transfer and click **Ok**. A new execution course should appear with the selected curricular coures and shifts.

Move Annoucements

You may want to transfer the annoucements from a execution course to another. The annoucements can be movied from a execution period to another. First select the execution period you want. All execution courses in that period should appear. Next select the execution course to where you want to move the annoucements. Check the annoucements you want to move and click submit.



Mover Anúncios

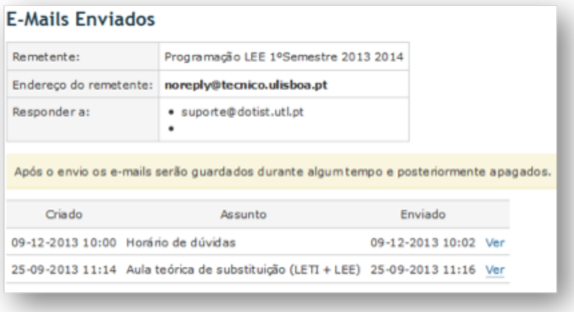
LEE - Programação (Pro6)

Período Execução:	1 Semestre 2013/2014
Disciplina execução de destino:	Acessibilidade - Ace - (DEIC)
Anúncios:	✓ **Revisão de prova - segundo teste** , Corpo Docente de Programação , 07-01-2014 11:23 ✓ **Notas do segundo projecto** , Corpo Docente de Programação , 07-01-2014 11:23 ✓ **Revisão de provas** , Corpo Docente de Programação , 06-01-2014 14:32 ☐ **Inscrições para o segundo teste** , Corpo Docente de Programação , 06-01-2014 11:55

Figure 5.22: Move Annoucements

Sent EMails

You can also see the emails sent within the context of a execution course. The mail records are stored for a while. By clicking in **Sent Emails** you should get a list of emails sent, as shown on Image 5.23.



E-Mails Enviados

Remetente:	Programação LEE 1ºSemestre 2013 2014
Endereço do remetente:	**noreply@tecnico.ulisboa.pt**
Responder a:	• suporte@dotist.utl.pt •

Após o envio os e-mails serão guardados durante algum tempo e posteriormente apagados.

Criado	Assunto	Enviado	
09-12-2013 10:00	Horário de dúvidas	09-12-2013 10:02	Ver
25-09-2013 11:14	Aula teórica de substituição (LETI + LEE)	25-09-2013 11:16	Ver

Figure 5.23: Sent emails

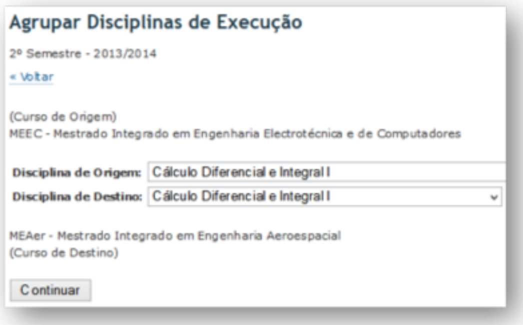
Remove

You can remove the execution course, only when there isn't any information associated with it (schedule, sumaries, announcements, etc.). To remove a execution course click in **Delete**. A confirmation should appear. Please keep in mind that this operation is irreversible.

5.3.3 Join Execution Courses

This operation allows joining two execution coures into one, meaning all the information from a execution course (faculty, sumaries, annoucements, students, etc.) will be connected into a single execution course. You can only join execution coureses within the same execution period (year/semester). To join select execution coureses select the execution period, Origin degree and Destination Degree. Click **Continue**.

Next, select the origin execution course and the destiny execution coruse. and click **Continue**. You should be presented with a warning message. Click ok to continue.Please keep in mind that this operation is irreversible.


Agrupar Disciplinas de Execução

2º Semestre - 2013/2014

« Voltar

(Curso de Origem)
MEEC - Mestrado Integrado em Engenharia Electrotécnica e de Computadores

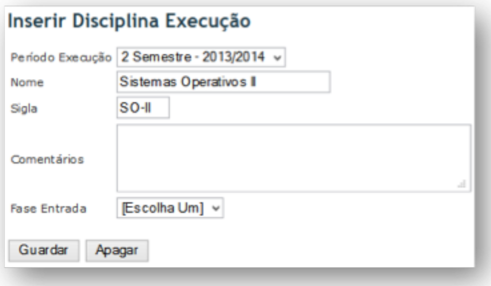
Disciplina de Origem:	Cálculo Diferencial e Integral I
Disciplina de Destino:	Cálculo Diferencial e Integral I

MEAer - Mestrado Integrado em Engenharia Aeroespacial
(Curso de Destino)

Figure 5.24: Joining execution coureses

5.3.4 Add Execution Courses

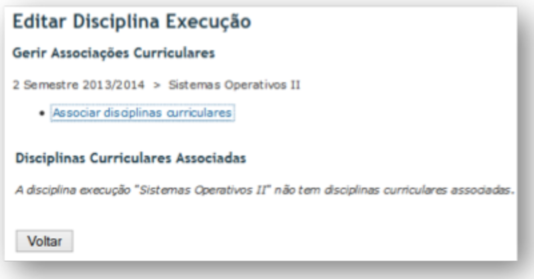
This option allows to create execution courses manually, and later being associated with curricular coureses. Click **Add Execution Course**, set the execution period, name, and acronym. Click **Save**.


Inserir Disciplina Execução

Período Execução	2 Semestre - 2013/2014
Nome	Sistemas Operativos II
Sigla	SO-II
Comentários	
Fase Entrada	[Escolha Um]

Figure 5.25: Adding a execution course

To use the execution course, click in **Edit Execution** course and select the execution period. Check the option *Courses not associated with curriculums*. and click **Continue**. A list with execution courses not associated with curricular courses should appear. Next follow the instructions from Section 5.3.2. to associate curricular courses.



Editar Disciplina Execução

Gerir Associações Curriculares

2 Semestre 2013/2014 > Sistemas Operativos II

- Associar disciplinas curriculares

Disciplinas Curriculares Associadas

A disciplina execução "Sistemas Operativos II" não tem disciplinas curriculares associadas.

Figure 5.26: Execution courses without curricular courses

Part II

Developer Manual

6 Fenix Framework

"Fenix Framework allows the development of Java-based applications that need a transactional and persistent domain model"[1]

This chapter describes in detail the major components of the Fenix Framework, which provides core functionality to FenixEdu. Section 6.1 describes the Domain Modelling Language, used to describe the application's domain model. Section 6.2 describes the high-level architecture of the Framework, briefly describing its major components and their interaction. Section 6.3 describes the process of Code Generation. Section 6.4 presents the Java Versioned STM (JVSTM), and its integration with the Fenix Framework. Understanding the Fenix Framework is critical for developing with the FenixEdu platform.

6.1 Domain Modelling Language

The Fenix Framework is aimed at enterprise-class applications with a rich domain model in an object-oriented paradigm. Such applications typically consist of class hierarchies representing entities with relationships among them, forming an interconnected graph.

The Domain Modelling Language (DML) is a Domain-Specific Language designed to represent such domain models, separating the domain's structure from its behaviour. The DML is designed with modularity as a core concern, allowing for incremental and modular domain definition.

In a DML file, programmers write the domain definition in a Java-like language. A class definition consists of the class name, the entity slots (either primitive or value types), and the super class. Listing 6.1 shows an example of a simple domain with a `User` entity. Note that just like in Java, it is possible to define the visibility of the persistent slots.

[1] `http://fenix-framework.github.io`

```
public class User {
  public String username;
  public DateTime lastLogin;
  protected String password;
}
```

Relations in DML are named, first-class citizens that represent relationships between two classes. Relations are always bi-directional, meaning that updating one side of the relation will automatically update the other side. Relations can be concealed in one of the sides (meaning that it will not be possible to access it), however their state is still kept.

It is possible to define `one-to-one`, `one-to-many` and `many-to-many` relationships, and it is possible to define boundaries on the multiplicity of each relation (for example, a `User` may have anywhere between 0 and 30 `DomainOperationLogs`). Any violation to these constraints would put the relation in an inconsistent state and cause the modification to be discarded.

To-many relations in the Fenix Framework have `Set` semantics, meaning that an object can only be present in a relation once. Also, there are no ordering guarantees when accessing the relation.

```
relation UserHasExecutedOperations {
  public User playsRole author {
    multiplicity 1..1;
  }
  public DomainOperationLog playsRole operationLog {
    multiplicity *;
  }
}
```

Listing 6.1 shows how a relation between a `User` and his records of his operations (`DomainOperationLogs`) could be described in the DML. The relation is given a name that describes the relationship between the two classes, as well as names to describe the role each class takes in the relation. The multiplicity is defined for each of the roles, in this case, one `User` has zero or more `DomainOperationLog`, and one `DomainOperationLog` has between one and one (exactly one) `User`.

From the domain definition, the Fenix Framework generates Java getters and setters for the properties and relations. For each class described in the DML, two Java classes are created: the domain class, in which programmers can include business logic, and a `Base` class (which the domain class extends) containing generated methods to access the persistent entities of the object. The visibility of the generated methods is determined by the visibility declared in the DML.

Consider the DML on Listing 6.1. Listing 6.1 shows the Code generated for the `User` class and its corresponding base class, as well as the `DomainOperationLog` base class, containing the API generated from the domain definition. For each slot declared in the DML a pair of Getter and Setter is generated, providing access to the persistent field. For relations there are two types of generated methods, depending on the multiplicity of the relation in the class. For instance, as one `DomainOperationLog` has one `User`, the generated methods are simple getters/setters for the `User`, as if it were a simple slot. On the other side of the relation, as a `User` has multiple

DomainOperationLog objects, the generated methods return a Set containing all the elements in the relation. The Framework also generates two methods to add and remove an element from the Set.

Whereas in the code shown in this document the body of such methods is /* Generated */, the actual code will depend on the backend used in runtime. More details about the Code Generation process are given in Section 6.3.

```
public class User extends User_Base {
  public User() {
    super();
  }
}

public abstract class User_Base extends
                                       AbstractDomainObject {

  public String getUsername() { /* Generated */ }
  public void setUserame(String value) { /* Generated */ }

  protected String getPassword() { /* Generated */ }
  protected void setPassword(String value) { /* Generated */ }

  (...)

  public Set<DomainOperationLog> getOperationLogSet() { /* Generated */ }
  public void addOperationLog(DomainOperationLog value) {/* Generated */}
  public void removeOperationLog(DomainOperationLog value) {/* Generated */}

}

public abstract class DomainOperationLog_Base extends
                                      AbstractDomainObject {

  public User getAuthor() { /* Generated */ }
  public void setAuthor(User author) { /* Generated */  }

}
```

6.1.1 Value Types

In the DML there is a distinction between entities and value objects. Whereas an entity is transactional and persistent, a value object is immutable and not persistent. Value objects are used as the values for slots of DML classes, and must be of well-known types, known as Value Types.

A value type contains information regarding the Java Type (such as java.math.BigDecimal), an alias (such as BigDecimal), and information regarding how the object will be externalised/internalized.

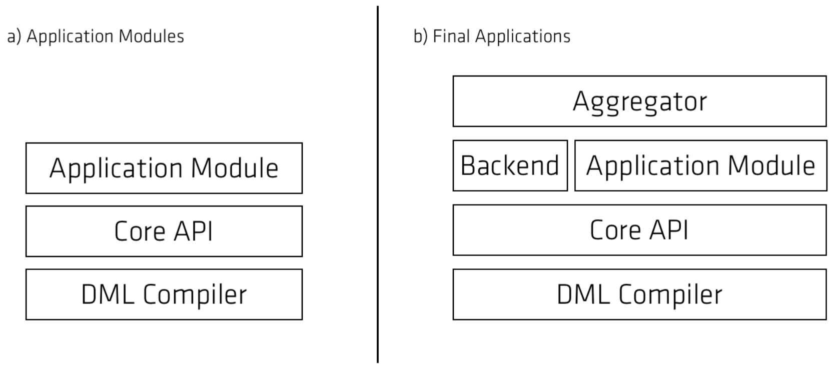


Figure 6.1: Fenix Framework's Layered Architecture

There are two categories of Value Types: Builtin (Java Primitives and their wrappers, Enums, JodaTime types, JsonElement, and byte arrays) and user-defined. User-defined types allow the programmer to use any type they wish as a slot, provided the type is immutable (explanation as to why is provided ahead) and can be expressed in terms of other Value Types.

The Framework knows how to handle Builtin Value Types (i.e. how to store/retrieve from persistent support). User-defined, on the other hand, require that the programmer specify how the type is externalised/internalized. The externalised type must be either a Builtin Value Type, or one or more used-defined types that ultimately externalise to a builtin type.

The Framework provides support to transform any Value Type to/from JSON, meaning that a JsonElement slot is enough to keep any value the Fenix Framework is able to handle.

6.2 Architecture

The main goal of the architecture of the Fenix Framework is to provide a "write once, run everywhere" paradigm for applications that have transaction and persistence requirements. The programmer simply writes his application against a public API and is able to run it on multiple backends. With this architecture, applications are portable across several technologies (MySQL, Neo4j, Hibernate, etc), and are easily testable, by providing a discardable persistence context.

Figure 6.1 shows the major modules that constitute the Fenix Framework.

The DML Compiler module contains the parser responsible for reading DML files and creating an in-memory description of the Domain Model, as well as all the necessary classes to represent it. It also contains the base `DomainObject` interface, which all objects of the Domain Model implement. Also present in this module are the base Code Generators used to create the Base classes for all domain objects.

The core of the Framework is in the `Core API` module. Transaction management APIs, configuration, entry points, backend interfaces, are all defined in this module.

Many backend-independent modules are provided with the Fenix Framework bundle. These include persistent Abstract Data Types (B+Trees, Linked Lists, etc), support for Consistency Predicates, statistics collection tools, indexing and transaction introspection. These modules are used internally by the various backends, but can (and in most cases should) also be used by the programmers.

Backends provide the concrete implementations of transactional and persistence support, and are required for applications to work.

6.2.1 Public API

One of the major advantages of the Fenix Framework is that applications built on top of it are independent of the underlying persistence and transactional backend. However, to guarantee this property, the modules that form such applications must be compiled against the Public API of the Framework.

The public API is split in two major components:

- **DML Compiler** Allows applications to have runtime access to the structure of the domain and allows registering relation listeners that will be invoked each time a relation is modified. This module also defines the API that is generated based on the DML, which must be supported by all backends.
- **Core API** Provides the entry point for the domain's object graph through the `DomainRoot` class, the mechanism to retrieve a Domain Object from its unique identifier, to validate if a given Domain Object is still valid, transaction APIs that can be used to either mark a piece of code as transactional (@Atomic annotation) or to manually manage the lifecycle of transactions, and utilities to configure the Framework.

With this separation, the classpath of the application modules is not polluted with implementation details, allowing for a faster development and test cycle.

6.2.2 Backends

Backends are a crucial part of the Fenix Framework. They provide concrete implementations of the transactional and persistence support. Application modules should not depend directly on the backends, as their API is private and as such subject to change even among minor versions, and having a dependency on a specific backend means that portability must be sacrificed.

The fact that backends have a clear separation from the Public API allows for a much faster evolution of the backend's implementation. Major internal changes can be done without affecting the end-users directly, even in a revision release, whereas changes to the public API require either a major or minor release.

6.3 Code Generation

As previously described, access to persistent fields of domain objects is done using generated methods in Base classes.

Code generation is closely tied to the specific backends, as it is typically used to support the process of persisting an object to a database. An example of an operation performed in generated code is binding a `PreparedStatement` with the values of the object's slots, or externalizing the object to JSON.

There are two major components in code generation:

- The default code generator, from which every other generator inherits, defines the public generated API for domain classes. Its major use-case is to compile backend agnostic application modules that only need base classes because of their API (modules aren't bundled with base classes, those are generated on-demand on depending modules or applications).
- Backend-provided code generators. These generators extend the base ones, thus providing the same API, using backend-specific artifacts to fullfil the API. Backends can also use code generators to optimise runtime performance, by injecting in generated code values that otherwise would have to be computed at runtime.

Whereas this Code Generation architecture allows for great optimizations (it allows backends to perform complex operations without resorting to reflection), it comes with a tradeoff: Forcing the domain classes to inherit from Base classes hinders the readability of the code (as the user is required to either check the DML or the base class to find out the super class), makes it impossible to invoke constructors of the super class (as the Base class only generates the no-arg constructor), does not allow the programmer to choose the visibility of the generated methods (as they are always generated public) among other issues.

The Code Generation step also provides a mechanism to transfer compile-time information to the runtime. Information such as the App Name (which is used at runtime to generate the graph of DML dependencies), the name of the Backend used to compile the final application (to allow for automatic initialization), as well as pass user-defined parameters to runtime.

6.4 JVSTM

The Java Versioned Software Transactional Memory (JVSTM) is a pure-Java implementation of a Software Transactional Memory.

The JVSTM uses the concept of Versioned Boxes (VBoxes) to make a memory location transactional, keeping the history of values for that position, one for each version of the box. Reads and writes to VBoxes are tracked by the JVSTM in a per-transaction basis.

Each transaction begins at a given moment, acquiring the version number at that moment. The version number is used during the transaction to ensure that all reads get the correct value at the time of the transaction's start, thus providing Opacity guarantees. This allows for conflict-free read-only transactions, as concurrent transactions writing to the read boxes will write a new version instead of overwriting its previous value.

6.5 JVSTM/OJB

The JVSTM is integrated with the Fenix Framework, as one of the multiple available backends. This document focuses on the backend named `jvstm-ojb`. This is the backend currently used in production. This backend uses the JVSTM for the transactional support, and OJB for persistence.

In this backend, Base classes use a VBox to store the value objects transactionally, thus taking advantage of the JVSTM. The generated getters and setters are backed by `VBox.get()` and `VBox.set()`, and can be invoked only from within a transaction.

7 Bennu

Bennu[1] is the foundation for building modular Java web applications based on Fenix Framework (see Chapter 6) .

Bennu covers the following core features:

- Runtime installation configuration
- Runtime application menu configuration
- User authentication
- Access groups
- Rest server infrastructure
- Transactional IO
- Scheduling system (crontab like configuration)
- Indexing and search
- Theming
- Internationalization

[1] `http://github.com/FenixEdu/bennu`

7.1 Users and Security

Bennu provides user accounts and authentication. The User accounts are supported in Bennu only at the domain level. To manage accounts take a look at the `bennu-user-management`[2] project.

Besides the authentication mechanism itself Bennu ensures that at any point in the source code the user in session is accessible statically through:

```
org.fenixedu.bennu.core.security.Authenticate.getUser()
```

or from the HttpSession using the property:

```
Authenticate.LOGGED_USER_ATTRIBUTE
```

7.1.1 Authentication Mechanisms

There are two supported authentication mechanisms: CAS[3] and password based authentication.

CAS authentication can be enabling by setting the properties in the `configuration.properties` file: `cas.enabled=true` and `cas.serverUrl` to your cas server installation url.

The password authentication mechanism is enabled when CAS isn't. So just set `cas.enabled=false`. Passwords are stored in the database, salted and hashed. No interfaces for password recovery or change are provided by the framework, to use this authentication mechanism you must provide them yourself.

7.1.2 Authentication Listeners

Bennu provides a way of registering event listeners over any successful login. To do so you must implement:

```
org.fenixedu.bennu.core.security.AuthenticationListener
```

and register it in the startup of the application using:

```
Authenticate.addAuthenticationListener(AuthenticationListener)
```

[2]https://github.com/FenixEdu/bennu-user-management
[3]http://www.jasig.org/cas

7.2 Access Groups

Bennu represents groups of users as instances of the **Group** entity. A group can be mapped to a set of users, translated to a String expression back and forth, or composed with other groups with logic operands.

At any given time there is at most one instance of Group with the same expression. In practice, Groups are immutable domain entities, and instances of groups are never obtained from constructor invocation, instead they are obtainer by: parsing a group expression (see: 7.2.5), invoking the group type **getInstance(...)** method, or by invoking an operation over another instance of a group.

7.2.1 Programatic API

All groups support the same basic API. First there are methods to inspect if a user is part of the group, or to get all the users that are part of the group. Both these two types of methods can be invoked with a date parameter, in that case, the group tries to evaluate the request on the state of the system at that date.

```
Set<User> getMembers();
Set<User> getMembers(DateTime when);
boolean isMember(User user);
boolean isMember(User user, DateTime when);
```

You can also apply operand to any group, the result is the group that represents the result of applying the operand to the target group, without changing the target. Whenever possible the group is compressed to the most canonical form, for example, x.not().not() is x.

```
Group and(Group group); — Intersect with given group.
Group or(Group group); — Unite with given group.
Group minus(Group group); — Subtract with given group.
Group not(); — Negate the group.
Group grant(User user); — Grants access to the given user.
Group revoke(User user); — Revokes access to the given user.
```

As said before groups can be parsed back and forth to a String expression.

```
static Group parse(String expression);
String expression();
```

Finally the groups can be used in a reversed way: intead of asking if a user is a member, or get the members of a group, we ask what are the groups a given user has access to. This is potentially a costly operation, so although is acessible over the User API, it should accessed, only for the logged user through the Authenticate API:

```
Set<Group> Authenticate.accessibleGroups();
```

7.2.2 Core Groups

There are a few provided group types, a way of providing your own implementation (see 7.2.4), and a way of placing a tag over a group (see 7.2.3).

The core groups are four logical conditions, and an explicit user set:

- **AnyoneGroup** Group of all users in the system, anyone is a member. Obtained with `AnyoneGroup.getInstance()`.
- **NobodyGroup** Group of no users, nobody is a member. Obtained with `NobodyGroup.getInstance()`.
- **LoggedGroup** Group of users logged in. Obtained with `LoggedGroup.getInstance()`.
- **AnonymousGroup** Special group for anonymous (not logged in) 'users'. Obtained with `AnonymousGroup.getInstance()`.
- **UserGroup** Explicit set of users. The user order is irrelevant, so there should be only one instance for the same set of users. Obtained with `UserGroup.getInstance(Set<User>)`.

Plus the compositions:

- **UnionGroup** The members are the union of all member sets of the contained groups. Group order is irrelevant. Obtained with `UnionGroup.getInstance(Set<Group>)`
- **IntersectionGroup** The members are the intersection of all member sets of the contained groups. Group order is irrelevant. Obtained with `IntersectionGroup.getInstance(Set<Group>)`
- **DifferenceGroup** The members are the result of removing the members of all but the first group to the members of the first group. In this case order is relevant. Obtained with `DifferenceGroup.getInstance(Set<Group>)`
- **NegationGroup** The members are all the users in the system that are nor part of the contained group. Obtained with `NegationGroup.getInstance(Group)`

7.2.3 Dynamic Group

Dynamic Group work as tags, that can be moved from one group to another over time, keeping history when they do. Tags offer semantic over the underlying group, like a name over it, for example, you can set a `managers` tag over the group of users John, Mary and Steve.

Operators work the same way as with any other group, they do not modify the dynamic group target, so by invoking `grant` with some user to the `managers` group will not make him/her a manager. To change a DynamicGroup, that is, to move a tag to another group invoke:

```
DynamicGroup.changeGroup(Group);

for example:
DynamicGroup managers = DynamicGroup.getInstance(”managers”);
managers.changeGroup(managers.grant(User.findByUsername(”userX”)));
```

7.2.4 Custom Group

Custom Groups are the extention point of the access group system, they allow the developer to provide their own group implementation, without loosing any of the Group’s features.

Lets say for example that you have Departments in your organization with people being part of them, and you need a group of people from a given department. Start by defining an entity in the dml like **DepartmentMemberGroup** related to the **Department** by a one to one relation, then annotate the class with **@CustomGroupOperator** specifying the operator that will serve as the key for the group expression:

```
@CustomGroupOperator(value = ”dep”)
public class DepartmentMemberGroup extends DepartmentMemberGroup_Base {
    protected DepartmentMemberGroup(Department department) {
        super();
        setDepartment(department);
    }
}
```

Make sure the constructor is private or protected. Instantiation is not to be left unguarded. Next implement the CustomGroup API, providing an implementation for the isMember and getMembers methods that delegate to the department:

```
@Override
public Set<User> getMembers() {
    return getDepartment().getMembers();
}

@Override
public Set<User> getMembers(DateTime when) {
    return getDepartment().getMembers(when);
}

@Override
public boolean isMember(User user) {
    return getDepartment().isMember(user);
}

@Override
public boolean isMember(User user, DateTime when) {
    return getDepartment().isMember(user, when);
}
```

Also provide a user readable description of your group:

```
@Override
public String getPresentationName() {
    return getDepartment().getName() + "␣Members";
}
```

Next let's handle the language traslation and the mechanism for obtaining unique instances. The group language for custom groups has the form:

```
<operator>(<arg1>, <args2>, ...)

in this case, for example:
dep(Sales)
```

The operator is specified in the **@CustomGroupOperator** annotation and must be unique in your codebase. The args are specified using another annotation: **@CustomGroupArgument** over a static method that resturns an argument translator. Argument translator are able to turn plain Strings into domain entities and vice versa, in this case the target Department of your group:

```
@CustomGroupArgument
public static SimpleArgument<Department, DepartmentMemberGroup> department() {
    return new SimpleArgument<Department, DepartmentMemberGroup>() {
        @Override
        public Department parse(String acronym) {
            return Department.fromAcronym(acronym);
        }

        @Override
        public String extract(DepartmentMemberGroup group) {
            return group.getDepartment().getAcronym();
        }

        @Override
        public Class<? extends Department> getType() {
            return Department.class;
        }
    };
}
```

This arguments are then passed to a **getInstance(...)** method that you must implement:

```
@Atomic
public static DepartmentMemberGroup getInstance(final Department department) {
    return select(DepartmentMemberGroup.class, new Predicate<
        DepartmentMemberGroup>() {
        @Override
        public boolean apply(DepartmentMemberGroup group) {
            return group.getDepartment().equals(department);
        }
    }, new Supplier<DepartmentMemberGroup>() {
        @Override
        public DepartmentMemberGroup get() {
            return new DepartmentMemberGroup(department);
        }
    });
}
```

Finally to comply with the reverse queries from the user: the `User.accessibleGroups()` method. You must implement a method named groupsForUser that returns the groups of this type that the given user is a member of:

```
public static Set<Group> groupsForUser(User user) {
    Set<Group> groups = new HashSet<>();
    for (Department department : user.getDepartments()) {
        if (department.getDepartmentMemberGroup() != null) {
            groups.add(department.getDepartmentMemberGroup());
        }
    }
    return groups;
}
```

7.2.5 Group Language

The group language is composed of operands (union, intersection, difference and negation), the four logic leaf groups, and the UserGroup

```
(<group> | <group> | ..... )
(<group> & <group> & ..... )
(<group> - <group> - ..... )
!<group>
anyone
nobody
logged
anonymous
U(username1, username2, ...)
```

The dynamic groups are just an # followed by the name, for example:

```
#managers
```

A custom group is the operator, followed by the arguments bewteen parenthesis:

```
<operator>(<arg1>, <arg2>, ...)
```

To obtain a group instance from the language use `Group.parse(String)` method, for example:

```
Group.parse(”((dep(Sales)␣-␣U(john,␣mary))␣|␣#managers)”);
```

meaning everyone in the sales department, except john and mary or the managers

7.3 Rest Infrastructure

Bennu provides a small layer of integration with Jersey[4]. All **Resources** annotated with **@Path** will be automatically mapped to:

[4] https://jersey.java.net/

```
/api/<module–name>/<rest–of–path>
```

To all **Resources** extending **JsonAwareResource** a mechanism for translating objects into json is also provided. Three families of methods are provided:

- **viewers**: turn objects into json
- **creators**: turn json into a new object of a given type
- **updaters**: apply the changes given in json format to a given object instance

These methods rely on a registry system of type translators. The registry is based on annotations: `JsonCreator<T>`, `JsonUpdater<T>`, `JsonViewer<T>`, or simply `JsonAdapter<T>` that extends the other three.

As an example, this is the adapter for the **Group** type:

```
@DefaultJsonAdapter(Group.class)
public class GroupJsonAdapter implements JsonAdapter<Group> {
    @Override
    public Group create(JsonElement json, JsonBuilder ctx) {
        return Group.parse(json.getAsJsonObject().get(”expression”).
            getAsString());
    }

    @Override
    public Group update(JsonElement json, Group obj, JsonBuilder ctx) {
        return Group.parse(json.getAsJsonObject().get(”expression”).
            getAsString());
    }

    @Override
    public JsonElement view(Group group, JsonBuilder ctx) {
        JsonObject object = new JsonObject();
        object.addProperty(”expression”, group.expression());
        object.addProperty(”name”, group.getPresentationName());
        object.addProperty(”accessible”, group.isMember(Authenticate.getUser()
            ));
        return object;
    }
}
```

With this defined, in a **JsonAwareResource** one could simply return:

```
return view(DynamicGroup.getInstance(”managers”));
```

And the endpoint in question with return the json format of the **Group**.

The methods in the **JsonAwareResource** can be parameterized to use custom JsonAdapters.

7.4 Transactional IO

Bennu provides a way to handle files that are linked to the domain in a transactional way. There are two key concepts here - the **GenericFile**: a first class domain entity that can be related to anything you like and the **Storage**: an abstraction over different types of ways to store byte arrays, that handle the storing and fetching of the byte array content.

The basic domain model is:

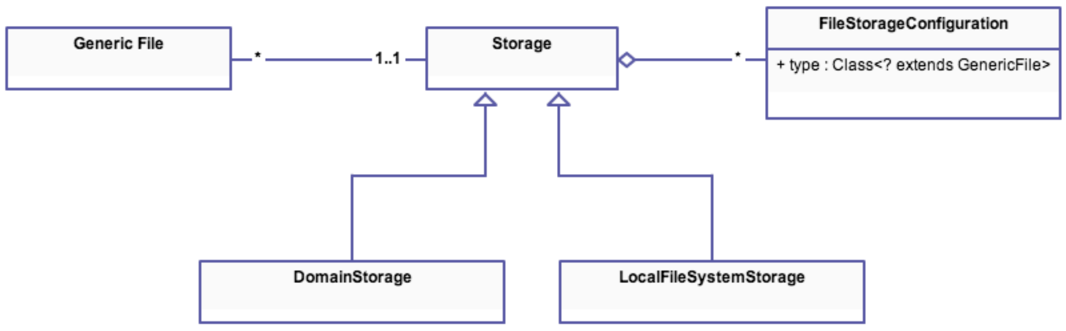


Figure 7.1: IO Domain Model

Upon instantiating a **GenericFile** the system searches for the configuration that says on which **Storage** the byte array will be saved, then saves it keeping a link between the **GenericFile** and the **Storage**.

Here is an example **GenericFile** implementation:

```
public class Picture extends Picture_Base {
    public Picture(String filename, Group availability, byte[] content) {
        super();
        init(filename, filename, content);
        setAvailability(availability);
    }
}
```

7.5 Scheduling

In several web applications there is the need of running periodic tasks on the system. Bennu provides a simple way of doing so. To create tasks that can be scheduled the developer needs to implement a **CronTask** and annotate it with **@Task**, like so:

```
@Task(englishTitle = ”Logger␣test”)
public class LoggingTask extends CronTask {
    @Override
    public void runTask() {
        DateTime now = new DateTime();
        DateTime timestamp;
        do {
            timestamp = new DateTime();
            taskLog(timestamp.toString());
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        } while (timestamp.isBefore(now.plusMinutes(1)));
    }
}
```

By default the task runs in write mode, to make it read-only change the annotation to:

```
@Task(englishTitle = ”Logger␣test”, readOnly = true)
```

Inside the `runTask()` you can call methods annotated with `@Atomic(mode=TxMode.WRITE)` as in any other piece of code.

Configuring a task to run periodically is done in the application's interface specifying a crontab like schedule periodicity[5]

Alternatively to developing a task in the codebase there is the possibility of submitting your code directly in the interface to run immediatly. To do so extend **CustomTask** instead, without annotation:

```
public class LoggingTask extends CustomTask {
    @Override
    public void runTask() {
        ...
```

[5] http://en.wikipedia.org/wiki/Cron

```
    }
}
```

And use the Custom Task submission interface.

7.6 Indexing and Search

Bennu provides a way to index and search domain entities, based on Lucene[6]. To make an indexable entity simply implement **Indexable** and **Searchable** interfaces. The **Indexable** is implemented by any entity that is able to provide index fields. The **Searchable** is implemented by any entity that when changed implies the recalculation of **Indexable**s.

For example, to index users by email and username, annotate the class User with both interfaces and implement the methods in the following way:

```
public class User extends User_Base implements Indexable, Searchable {
    public static enum UserIndexableFields implements IndexableField {
        USERNAME, EMAIL;

        @Override
        public String getFieldName() {
            return name().toLowerCase();
        }
    }

    @Override
    public IndexDocument getDocumentToIndex() {
        IndexDocument index = new IndexDocument(this);
        index.indexString(UserIndexableFields.USERNAME, getUsername());
        if (getEmail() != null) {
            index.indexString(UserIndexableFields.EMAIL, getEmail());
        }
        return index;
    }

    @Override
    public Set<Indexable> getObjectsToIndex() {
        return Collections.<Indexable> singleton(this);
    }
}
```

[6] https://lucene.apache.org

To search start by creating a **Search** object, then using fluent methods, parameterize your search. Available parameterizations include:

- Tokenized input match
- Entire phrase input match
- Wildcard match
- Fuzzy matches
- Range queries

Every type of parameterization can include a field, that if not specified the search is done on all fields. Also, the parameterizations have an Occurrence value, one of: Must, Should, Must Not.

An example Search parameterization could be:

```
Search s = new Search().tokens(UserIndexableFields.USERNAME, ”john”, Occur.
    MUST).wildcard(UserIndexableFields.EMAIL, ”*@gmail.com”, Occur.MUST)
```

Meaning Users with username john with a gmail email.

Searches can also be sorted over several fields. To add a sort criteria continue using the fluent methods:

```
s.sort(UserIndexableFields.USERNAME, false) // false if for reverse
```

Finally after all parameterizations invoke:

```
List<User> matches = s.search(User.class);
```