ACAD Oracle

Architecture Notebook

There is guidance within this template that appears in a style named InfoBlue. This style has a hidden font attribute that allows you to toggle whether it is visible or hidden in this template. Use the Microsoft® Word® menu **Tools > Options > View > Hidden Text** check box to toggle this setting. There is also an option for printing: **Tools > Options > Print**.

# Purpose

This document describes the philosophy, decisions, constraints, justifications, significant elements, and any other overarching aspects of the system that shape the design and implementation.

# Architectural goals and philosophy

The architectural shall be able to handle simultaneous use and respond efficiently. Although the main purpose of the application is to provide one only functionality that is to suggest the best combination of disciplines to course, it shall persist some data in order to do that. This way the system shall have a database. The system will not need to adapt to any legacy system, and by the nature of the project, it will not suffer by performance issues.

* It shall be a client-server architecture.
* It shall allow multiple people accessing the system at the same time (session control).
* It shall be secure and reliable.
* It shall have data persistency.

# Assumptions and dependencies

* The development team shall understand the concept of MVC, or shall have experience working in a multi-layered architecture.
* The team shall know how to work with relational databases and how to write T-SQL.
* The developers shall know how to program in C#.
* The team shall have understanding of how web applications and HTTP works.
* The web server shall have IIS 7 up and running.
* A SQL Server Express database shall be installed in the database server.
* The system shall be connected to the network in order to be used by the client. If either the database server and/or application server cannot have network, the system will not be usable.
* The client has a browser installed in order to use the system.

# Architecturally significant requirements

* Generate suggested disciplines based on restrictions.
* Log in the system.
* Manage course.
* Manage disciplines.
* Manage classes.

# Decisions, constraints, and justifications

* The system shall contain three core layers: UI, Business Layer and Data Layer. The advantage of this is that logic can be implemented once in the business layer and be available to the components throughout the UI layer. In addition, the database layer is isolated from the UI.
* Web application shall be implemented using the Microsoft MVC framework 4.0 with C# language. The framework is robust and reliable, and also have an easy to implement authentication functionality and session management.
* The data layer shall be implemented using the Microsoft Entity Framework. It enables quick setup and easy implementation to communicate with the database.
* The database shall be Microsoft Sql Server 2012. It has compatibility with the entity framework, is a relational database and easy to setup.
* The system shall use the Repository Pattern. The UI cannot access the database directly, only through the repositories. Each entity will have its repository.

This is the repository pattern diagram:



* The Controllers in the MVC project can reference the repositories. But only their interfaces. The implementation shall be injected using the injection pattern. The SimpleInjector library shall be used to inject the dependencies in the controller’s constructor. In order to work, there shall be the SimpleInjector setup when starting the application. An example of the Injector setup of the TurmaRepository:

container.Register<ITurmaRepository, TurmaRepository>();

This way, in the controller’s constructor, just add the ITurmaRepository object and the implementation will be injected by simple injector.

The reason to use this pattern is to reduce the coupling, increasing maintenance easy.

# Architectural Mechanisms

## Web oriented: can be accessed through a browser and doesn’t need to install a specific client application.

## Persistency: to save the data used to suggest the disciplines to be coursed.

## Exception Handling: to display useful user friendly messages.

## Logging Mechanism: to keep error and informational logs for troubleshooting. Log4net shall be used for logging.

## Entity Relationship Mapping framework: to make it easy to maintain the persistency. Microsoft Entity Framework 5. It is easy to setup and integrates well with Asp.NET MVC framework.

## DependencyInjection framework: to reduce coupling. SimpleInjector will be used for it. The documentation for SimpleInjector can be found here: <http://simpleinjector.codeplex.com/wikipage?title=MVC%20Integration>

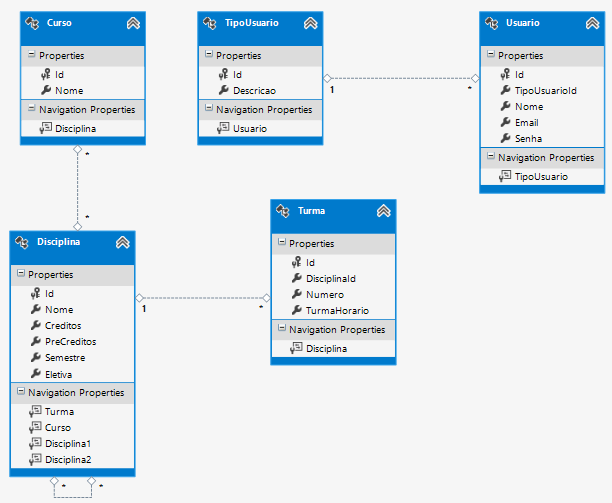
# Key abstractions

* Discipline: the input as the coursed by the student, and output as the suggested by the process.
* Student: the user of the system. The entity that will contain the other abstractions which the process will generate the suggestions.
* Course: they contain the disciplines.
* Restrictions: the user input so the system can generate the suggestions respecting the restrictions of the user. They will not be persisted, they will be evaluated in runtime and provided by the user when generating the suggested disciplines.
* Classes: will define the day/time the discipline can be coursed. A discipline can have more than one class.

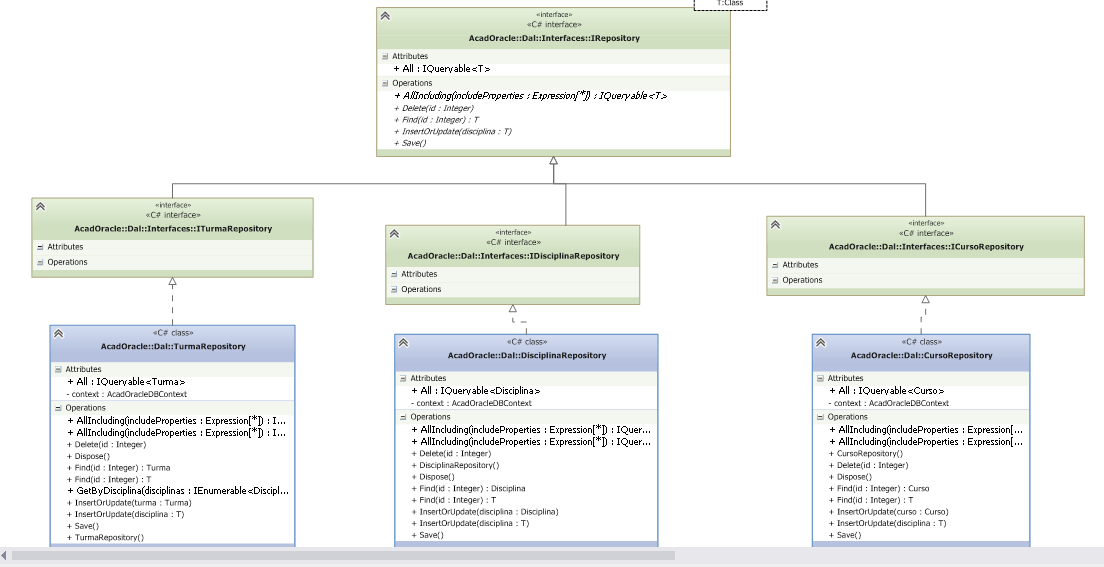
# Layers or architectural framework



**Entity Relationship Diagram**



**Repository Structure**



# Architectural views

## Recommended views

* **Use case:**

