Санкт-Петербургский государственный университет

Математическое обеспечение и администрирование информационных систем
242 группа

Тучина Анастасия Игоревна

Разработка сервиса для сдачи и проверки домашних работ с помощью WebSharper: База данных

Курсовая работа

Научный руководитель:
доц. Литвинов Ю.В.

Санкт-Петербург
2018

Оглавление

Введение

Одной из важных частей системы обучения является сдача и проверка домашних работ. Удобство используемых средств для сдачи и проверки заданий влияет на эффективность обучения: студент тем быстрее получит конструктивную критику своей работы, чем комфортнее преподавателю будет ее проверять.

Цель проекта – создание сервиса для сдачи и проверки домашних работ с использованием WebSharper

Общие принципы использования сервиса:

- Преподаватель создает курс, на который студент может записаться, и указывает, какие задания нужно выполнить для получения зачета.
- Студент записывается на выбранные им курсы.
- Преподаватель, который ведет курс, может отклонить или принять заявку на вступление.
- Студент размещает решения задач на странице курса.
- Преподаватель получает решение, и либо помечает его как выполненное, либо запрашивает изменения.
- Ведется таблица с успехами всех студентов на странице каждого курса.

WebSharper – инструмент с открытым исходным кодом, предоставляемый IntelliFactory, предназначенный для создания веб-приложений полностью на F# (с версии 4.0 и на C#), обеспечивающий преобразование кода на F# (C#) в JS/CSS/HTML на этапе компиляции[1].

Разработка проектов с помощью WebSharper – трудоемкая деятельность, требующая определенного опыта в области веб-разработки и знания некоторых концепций, характерных для веб-программирования. Также существует проблема недостаточного количества документации: существующая представляется в виде набора методов, зачастую без

указания случаев применения и подробного описания. Несмотря на это, данный проект интересен тем, что практически не существует проектов с открытым исходным кодом, использующих WebSharper, поэтому в дальнейшем он может послужить как пример работы с данным инструментом для других желающих его освоить.

1. Постановка задачи

- Выбрать СУБД и фреймворк для взаимодействия с базой данных
- Реализовать модель
- Выбрать подход к созданию базы данных
- Организовать взаимодействие с базой данных

2. Обзор существующих решений

- **HwProj**[2]
  Сервис для сдачи и проверки домашних работ, запущенный в 2014 году, основные принципы использования которого были частично позаимствованы для проекта. Концепция HwProj:

  – Сервис предусматривает две роли: преподаватель и студент. Это делает необходимым каким-либо образом различать их, например, хранить информацию о студентах и преподавателях в разных таблицах

  – Преподаватель создает курсы, на которые студенты могут записываться. Исходя из этого, можно создать еще одно отношение – курс. Причем нужно различать понятие курса и "прочтения" курса, так как один курс могут одновременно вести разные преподаватели. Также нужна специальная развязочная таблица, где будет храниться информация о студентах, записавшихся на курсы, чтобы реализовать связь many-to-many между сущностями "Студент" и "Курс"

  – Каждый курс имеет список заданий, которые студентам необходимо выполнить для получения зачета по дисциплине. Это осуществляется с помощью так называемых банков заданий.

  – Студенты видят только выданные преподавателем задания, поэтому нужны специальные таблицы, которые будут хранить информацию о них.

  – Информация с каждой пары хранится с помощью таблицы "Лекции", к которым могут прилагаться какие-либо объявления или материалы

  – Студенты отправляют решения преподавателю, который должен их оценить. Для этого можно использовать сущность "Решение", хранящую ссылку на решение и его статус(не загружено/запрос изменений/загружены исправления/принято)

HwProj реализован на Ruby, что затрудняет сопровождение сервиса, так как мало кто программирует на этом языке и владеет им на достаточном для этого уровне.

3. Реализация

3.1. Общее описание и инструменты

В целях достижения кроссплатформенности была выбрана платформа .NET Core. Код для работы с базой данных написан на C#, для обеспечения доступа к ним используется Entity Framework Core. Обращение к базе данных реализуется с помощью контекста данных DbContext(класс, осуществляющий подключение к базе данных), например, полученные от пользователя данные добавляют в БД через экземпляр DbContext. Управление данными происходит с помощью PostgreSQL.

3.2. Особенности реализации

3.2.1. Выбор типа СУБД

Управление использованием базы данных приложения осуществляется с помощью реляционной СУБД. Данный тип СУБД выбран по следующим причинам:

- Используемые данные строгая структурированы, при этом структуры почти не подвержены изменениям.
- Соответствие принципам ACID, что минимизирует вероятность неожиданного поведения системы.

3.2.2. Выбор фреймворка для доступа к данным

В текущей реализации использован один из наиболее популярных фреймворков, делающих возможным обращение к базе данных из кода, для .NET – Entity Framework Core[3].
Причины использовать Entity Framework Core:

- Использование технологии ORM, что позволяет работать с данными в терминах классов или объектов и наоборот, преобразовать данные классов в данные, пригодные для хранения в БД.

- Автоматически генерирует классы-модели в соответствии с сущностями и их атрибутами(при использовании подхода DB-first), позволяющие рассматривать каждый кортеж как объект.
- Возможность использовать LINQ для обращения к базе данных(LINQ to Entities и LINQ to Objects).

3.2.3. Модель

Структура базы данных:
Модель БД приложения представлена на рис. 1

- Student – список всех зарегистрированных студентов.
- Teacher – список всех зарегистрированных преподавателей.
- Hometask – банк домашних заданий.
- TestTask – банк тестовых заданий.

Используя прототипы заданий из банков, можно создавать экземпляры планируемых задач.

- Course – банк доступных для создания курсов.
- Homework – список планируемых домашних заданий для каждого из курсов в банке.
- Test – список планируемых тестовых заданий для каждого из курсов в банке.

Списки планируемых заданий обеспечивают преподавателю возможность выбирать только из относящихся к курсу задач, а не из всех существующих в банке, что гораздо удобнее, при этом различаются домашние и тестовые задания.

- OngoingCourse – список проводящихся на данный момент курсов.

Имеет смысл различать понятие курса и ”прочтения” курса.

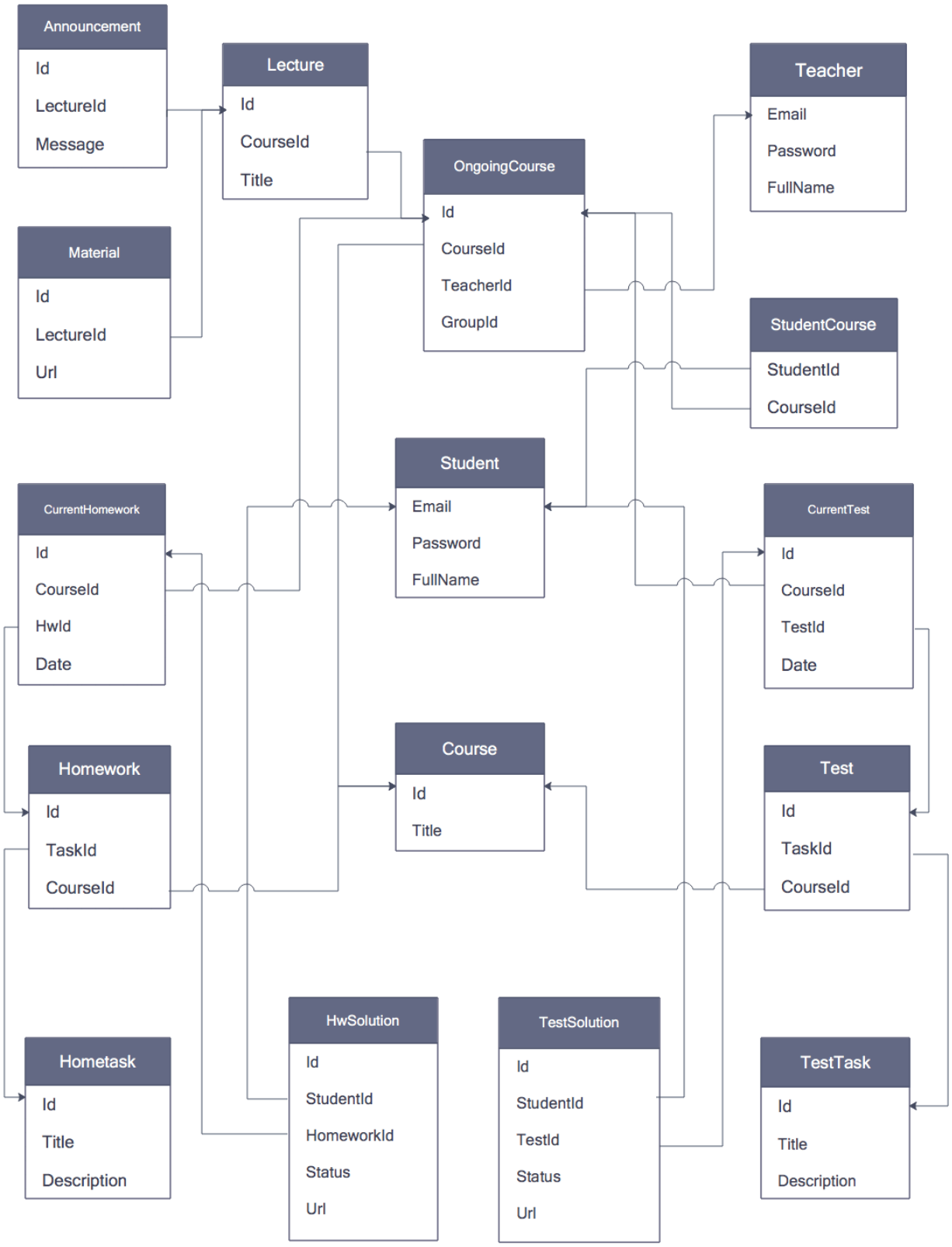

Рис. 1: Модель базы данных приложения

- CurrentHomework – список выданных домашних заданий для всех курсов.
- CurrentTest – список выданных тестовых заданий для всех курсов.

Дает возможность преподавателю делать для студентов видимыми только выданные задания.

- HwSolution – список загруженных студентами решений домашних заданий.
- TestSolution – список загруженных студентами решений тестовых заданий.
- StudentCourse – список, хранящий данные о том, какой студент на какие курсы записан.
- Lecture – список добавленных преподавателями лекций для различных курсов.
- Material – список добавленных преподавателями материалов к лекциям, представляемых в виде URL-ссылки.
- Announcement – список добавленных преподавателями объявлений к лекциям.

Заключение

Что было сделано:

- Выбран подход для создания БД
- Спроектирована база данных
- Реализовано взаимодействие с БД с помощью PostgreSQL и Entity Framework Core

Список литературы

[1] Websharper Documentation - WebSharper 4.x for F# // — `https://developers.websharper.com/docs/v4.x/fs`. (дата обращения: 04.06.2018)

[2] HwProj - сайт для проверки домашних работ // — `http://hwproj.me`. (дата обращения: 04.06.2018)

[3] Entity Framework Core - Entity Framework Core Quick Overview // — `https://docs.microsoft.com/en-us/ef/core/`. (дата обращения: 04.06.2018)