WEB-ПРИЛОЖЕНИЕ «ИСТОРИЧЕСКИЕ ОБЪЕКТЫ Г. ТОМСКА И ТОМСКОЙ ОБЛАСТИ»

Д.А. Кустов
Томский политехнический университет
kustov.denic@gmail.com

Введение

Существует немало информационных систем, которые позволяют получить информацию о памятниках и исторических объектах, а также посмотреть их расположение на карте. Однако большинство из таких систем содержат однообразную и не очень подробную информацию. Разработка же данного приложения обусловлена наличием базы данных с уникальным контентом. В этой БД содержится подробная информация о большом числе исторических объектов г. Томска и Томской области.

База данных

База данных работала под управлением СУБД Firebird. Эта СУБД имеет плохую поддержку, малое количество инструментов, а также сложно интегрируется с популярными средами разработки программного обеспечения. Кроме того, структура БД имела недостатки: наличие связей 1:1, отсутствие связей между некоторыми таблицами. В связи с этим, в рамках разработки приложения необходимо было исправить структуру БД и перевести ее под управление СУБД Microsoft SQL Server, не имеющую перечисленных недостатков.

С учетом выявленных недостатков в структуре имевшейся БД была спроектирована концептуальная модель новой базы: определен необходимый набор сущностей и атрибутов; таблицы, имевшие связи 1:1, были объединены; были восстановлены отсутствовавшие необходимые связи. В качестве инструмента проектирования использовался Sybase PowerDesigner 15.2. На рисунке 1 приведен фрагмент концептуальной модели, где проиллюстрировано объединение таблиц: атрибуты таблиц старой базы «KULTURE_OBJECT» и «OBJECT» были помещены в таблицу «Объект». Для повышения удобства работы с БД были созданы домены с типами данных атрибутов.

На основе разработанной концептуальной модели была сгенерирована физическая модель. По физической модели были созданы таблицы БД. Далее требовалось перенести данные из старой базы в новую. Для этого файл старой БД в формате СУБД Firebird был сконвертирован в файл формата СУБД MS SQL Server при помощи утилиты ESF Database Migration Toolkit. Были написаны необходимые скрипты для выборки данных из старой базы и вставки в новую. С целью повышения надежности работы БД тип данных ключевых атрибутов был изменен с int на uniqueidentifier (GUID, Globally Unique Identifier – статистически уникальный 128-битный идентификатор) с сохранением имевшихся зависимостей. Так как преобразовать int напрямую в uniqueidentifier невозможно [1], было осуществлено промежуточное преобразование в тип varbinary.

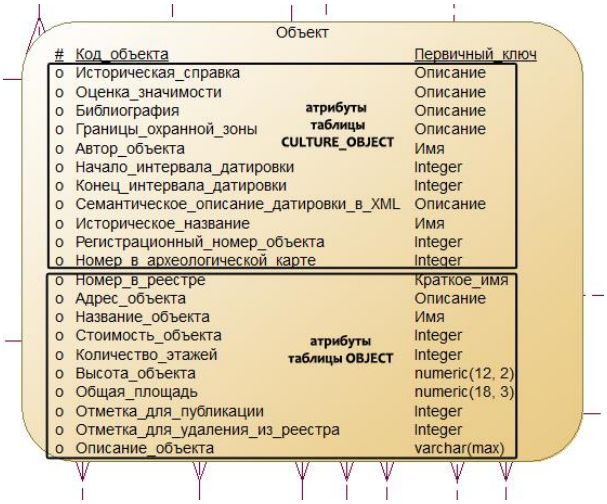

Рис. 1. Сущность, объединяющая атрибуты двух таблиц старой БД

Платформа ASP.NET MVC

Для реализации приложения была выбрана платформа ASP.NET MVC.

Шаблон MVC (Model – View – Controller), лежащий в основе данной платформы, подразумевает взаимодействие трех компонентов: контроллера (Controller), модели (Model) и представления (View).

Контроллер представляет класс, с которого начинается работа приложения. Этот класс обеспечивает связь между моделью и представлением. Получая вводимые пользователем данные, контроллер, исходя из внутренней логики, при необходимости обращается к модели и генерирует соответствующее представление.

Представление – это визуальная часть, или пользовательский интерфейс приложения, например, html-страница, через которую пользователь, взаимодействует с приложением.

Модель представляет набор классов, описывающих логику используемых данных.

Упрощенная схема взаимодействия компонентов приведена на рисунке 2.

Рис. 2. Схема взаимодействия компонентов MVC

Приложение

К созданному в Microsoft Visual Studio проекту приложения была подключена база данных, и на основе ее таблиц при помощи ADO.NET Entity Framework были сгенерированы классы. ADO.NET Entity Framework – это платформа доступа к данным, позволяющая работать с данными в виде объектов, не обращаясь напрямую к таблицам и атрибутам БД.

Далее были разработаны необходимые согласно шаблону MVC компоненты приложения.

Для класса контроллера были созданы функции, которые выбирают необходимые данные из БД, обрабатывают их и передают представлению. Функция List() необходима для вывода полного списка объектов, она выбирает все данные из таблицы с объектами; Map(Guid? id) получает координаты для выбранного объекта; Details(Guid? id) получает всю информацию о выбранном объекте.

Для реализации картографической составляющей был выбран сервис «Google Карты», так как API этого сервиса прост в интеграции.

Для функций контроллера созданы необходимые представления. Представление List выводит в виде таблицы основную информацию о всех объектах; Map выводит карту: в полученных координатах ставится маркер; Index необходимо для совместного вывода представлений Map и List (рисунок 3); Details выводит подробную информацию об объекте.

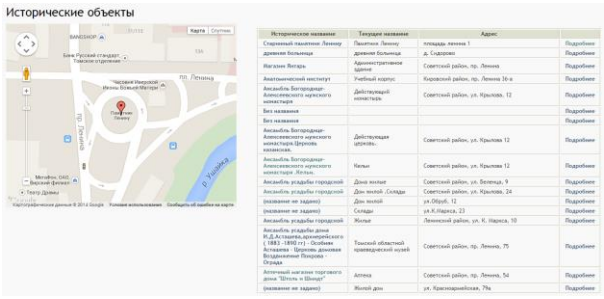

Рис. 3. Скриншот запущенного приложения

Работа пользователя с приложением осуществляется следующим образом (рисунок 4):

1. Когда пользователь открывает главную страницу, вызывается функция List(), выбирающая список всех объектов из БД. Этот массив передается представлению, которое отображается пользователю в виде Web–страницы.

2. После того, как пользователь выбирает какой-либо объект из списка, вызывается функция контроллера Details или Map, принимающая на вход идентификатор выбранного объекта. Функция обращается к БД и получает необходимые данные. Сгенерированные контроллером представления отображаются пользователю: выводится подробная текстовая информация об объекте, на карте ставится маркер в полученных из БД координатах (карта подгружается представлением с серверов Google).

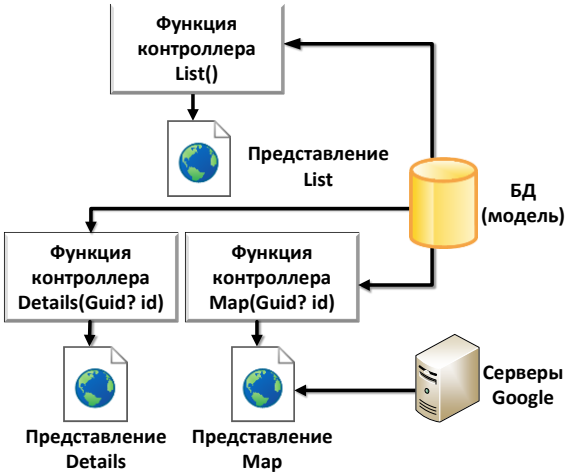

Рис. 4. Схема функционирования приложения

Заключение

В результате работы было разработано Web-приложение, позволяющее просматривать информацию об исторических объектах, их расположение на карте. Приложение находится в стадии разработки, поэтому еще не обладает достаточно широкими возможностями.

Литература

1. Преобразование типов данных (компонент Database Engine). [Электронный ресурс]. – Режим доступа: http://msdn.microsoft.com/ru-ru/library/ms191530(v=sql.110).aspx, свободный.

2. Онлайн-книга «Изучаем ASP.NET MVC 4». [Электронный ресурс]. – Режим доступа: http://metanit.com/sharp/mvc/index.php, свободный.

3. JavaScript API Google Карт (версия 3). [Электронный ресурс]. – Режим доступа: https://developers.google.com/maps/documentation/javascript/?hl=ru, свободный.

4. ASP.NET MVC 4: Как использовать несколько моделей в одном представлении. [Электронный ресурс]. – Режим доступа: http://codehammer.ru/?p=249, свободный.