УДК 664.012

О ПРИМЕНЕНИИ ПАТТЕРНА MVC В ОБРАЗОВАТЕЛЬНОМ ПРОЦЕССЕ ДЛЯ ДОСТУПА К БАЗЕ ДАННЫХ ПОЛЬЗОВАТЕЛЕЙ СОЦИАЛЬНОЙ СЕТИ

И.В. Акиншева, И.П. Овсянникова, В.А.Пантюхов
Могилевский государственный университет продовольствия,
г. Могилев, Республика Беларусь

В настоящее время использование архитектурного паттерна MVC (Model, View, Controller) является популярным у разработчиков сетевых приложений.

Model-View-Controller (MVC, «Модель-Представление-Контроллер», «Модель-Вид-Контроллер») — схема разделения данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: модель, представление и контроллер — таким образом, что модификация каждого компонента может осуществляться независимо.

Модель (Model) предоставляет данные и реагирует на команды контроллера, изменяя свое состояние.

Представление (View) отвечает за отображение данных модели пользователю, реагируя на изменения модели.

Контроллер (Controller) интерпретирует действия пользователя, оповещая модель о необходимости изменений.

Основной целью лабораторной работы является изучения принципа работы паттерна MVC для доступа к базе данных пользователей социальной сети.

При выполнении работы студент должен решить следующие задачи: разработать базу данных пользователей социальной сети; сформировать запросы к базе данных; оформить часть пользовательского интерфейса; сформировать защищенное соединение машины пользователя с сервером базы данных.

Для решения задач используются следующие инструменты:

– HTML, CSS, JavaScript;
– библиотека JQUERY;
– фреймворк Laravel;
– View Engine Blade;
– база данных MySQL;
– менеджер пакетов Composer.

Защищенное соединение разрабатывается следующим образом.

Работа программы начинается с момента перехода по ссылку socialnet.loc. При этом браузер отправляет запрос на сервер, после чего возвращается файл, описывающий начальную страницу с заранее подключенными файлами стилизации.

Далее для авторизации необходимо ввести e-mail, пароль и нажать кнопку «Войти», при этом сервер обрабатывает запрос через специальные скрипты, написанные на языке php, берет данные из базы данных и генерирует ответ. В зависимости от валидности введенных данных (пароль должен быть введен латинскими буквами в количестве не менее 8 символов, e-mail должен иметь соответствующую структуру), ответ может быть в виде перехода на защищенную ссылку (туда можно попасть только авторизировавшись), либо в виде перезагрузки данной страницы. Если введенная информация валидна, то сервер возвращает файл, описывающий страницу личного кабинета данного пользователя, с подключенными скриптами и файлами стилизации. Информация об авторизованности данного пользователя записывается в локальную память в виде «cockie». По такому же принципу работают другие защищенные запросы на сервер.

При создании модели данных, в представленном случае это будет пользователь, необходимо создать для нее класс и установить связи с другими моделями данных (другими пользователями). В разрабатываемом приложении используются связи «один ко многим», «многие ко многим», «один к одному». Рассмотрим пример, когда пользователь имеет несколько постов. На программном уровне это инициализируется следующим образом:

```
class User extends Model implements Authenticatable {
public function posts()
   {
      return $this->hasMany('App\Post');//Relations!!
   } }
```

Реализация связи «многие к одному» приводит к тому, что пользователь может иметь несколько постов, в свою очередь пост имеет одного пользователя:

```
class Post extends Model {
public function user()
   {
      return $this->belongsTo('App\User');
   } }
```

Согласно приведенному коду, при компиляции автоматически создаются таблицы в базе данных с требуемыми связями. Таким образом, разработчику не нужно знать все нюансы синтаксиса SQL для создания связей между отдельными таблицами в базе данных. Пример диаграммы связей между созданными в базе данных таблицами представлен на рисунке 1.

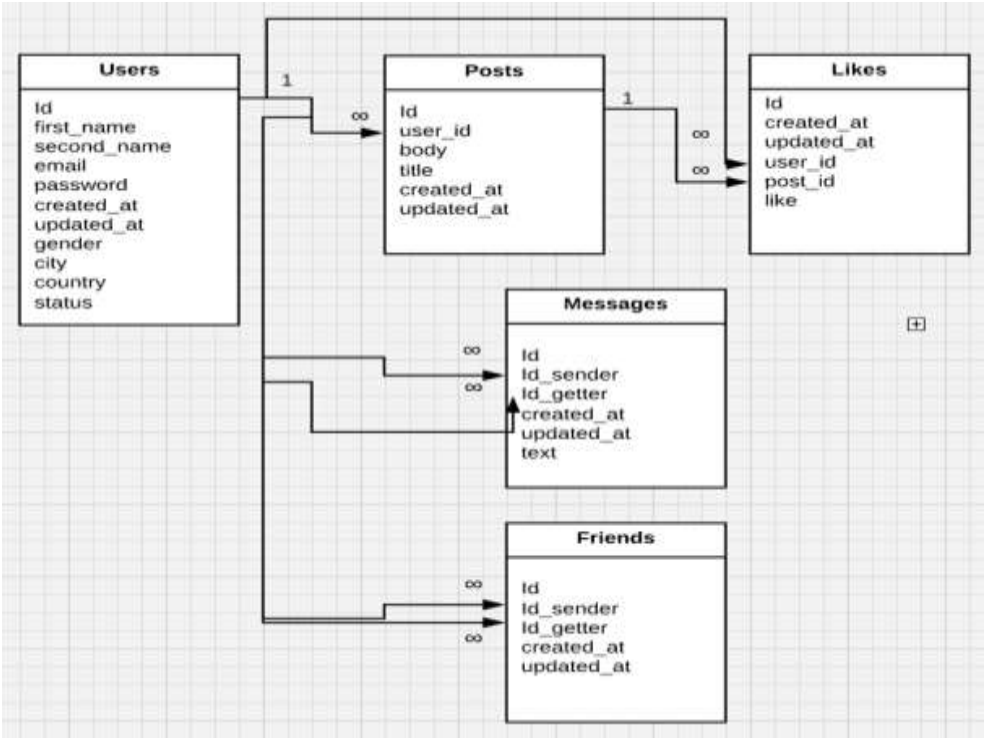

Рисунок 1 – Диаграмма связей таблиц в базе данных

Запросы происходят по следующему сценарию:

1) Пользователь входит в приложение, основной контроллер принимает запрос от приложения «клиента», расположенного на машине пользователя

2) Контроллер возвращает приложению «клиента» файл с гипертекстовой разметкой (view) и скрипты.

3) Браузер отображает файл с гипертекстовой разметкой и загружает скрипты.

4) При загрузке скриптов в приложение «клиента» оформляется остальная часть пользовательского интерфейса и добавляются элементы интерактивности на страницу в браузере.

Пример схемы работы запросов представлен на рисунке 2.

Для манипуляций со страницей студентами могут быть выбраны следующие средства: CSS, JavaScript.

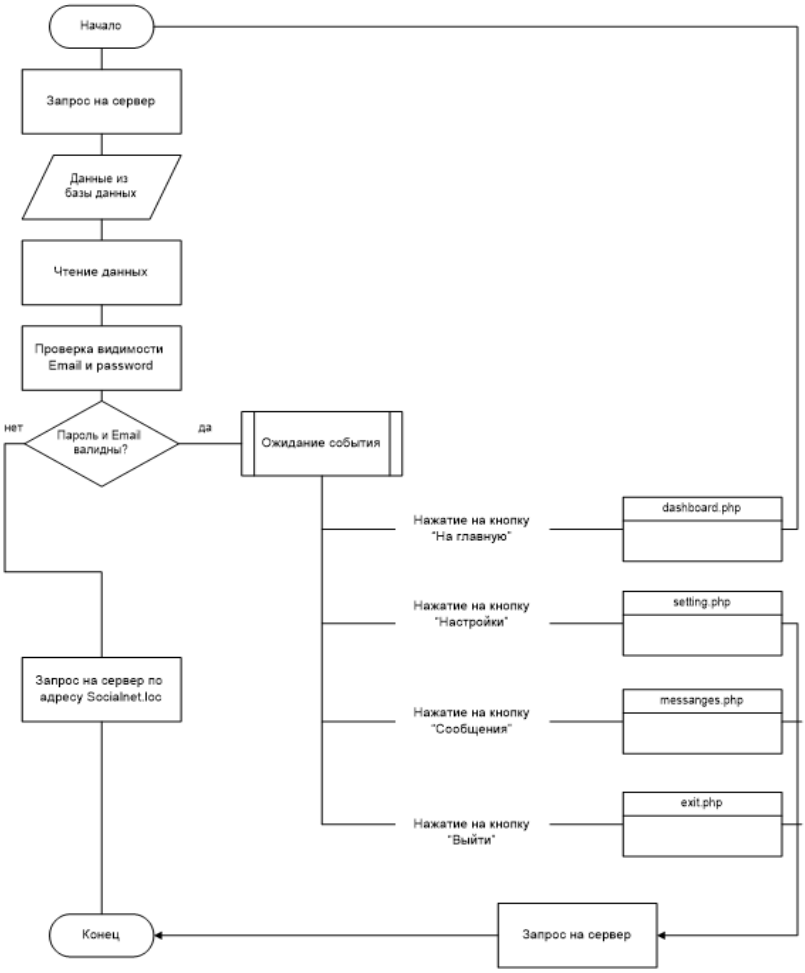

Рисунок 2 – Схема работы запроса по адресу socialnet.loc/dashboard

Применение данной разработки в учебном процессе позволило сформировать у студентов навык создания сетевых приложений на архитектурном паттерне MVC со следующими возможностями: валидацией данных на стороне «сервера»; валидацией данных на стороне «клиента»; применение современных UI-стандартов со стороны «клиента»; защищенное соединение как с административной панелью, так и с сервером базы данных, расширенные возможности работы с интерфейсом страницы пользователя.

В заключение можно добавить, что MVC паттерн дает возможность сделать проекты, разработанные студентами, легко расширяемыми под большие сетевые приложения.

Список литературы

1. Стивенс, У.Р. UNIX: разработка сетевых приложений / У.Р. Стивенс, Б. Феннер, Э.М. Рудофф. – СПб.: Питер, 2007. – 1039 с.

2. Котеров, Д. PHP 7 / Д. Котеров, И. Симдянов. – СПб.: БХВ-Петербург, 2016. – 1088 с.