Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Базы данных

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

на тему

РЕКЛАМНЫЕ ОБЪЯВЛЕНИЯ

БГУИР КР 1-40 01 01 418 ПЗ

Студент: гр. 051004 Рыхлицкий А.В.

Руководитель:

Асс. Фадеева Е.Е.

Минск 2013

**ОГЛАВЛЕНИЕ**

[ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ 4](#_Toc375150268)

[ВВЕДЕНИЕ 5](#_Toc375150269)

[1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 7](#_Toc375150270)

[1.1 Анализ литературных источников 7](#_Toc375150271)

[1.2 Анализ аналогов 8](#_Toc375150272)

[1.3 Спецификация требований 10](#_Toc375150273)

[1.4 Постановка задачи 12](#_Toc375150274)

[2 МАТЕМАТИЧЕСКИЕ МОДЕЛИ И МЕТОДЫ 13](#_Toc375150275)

[2.1 Система защиты данных OpenSSL 13](#_Toc375150276)

[3 АНАЛИЗ ТРЕБОВАНИЙ К ПРОГАММНОМУ СРЕДСТВУ И РАЗРАБОТКА ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ 14](#_Toc375150277)

[3.1 Анализ средств разработки и программных продуктов в сфере интернет 14](#_Toc375150278)

[3.2 Общие сведения о RoR MVC 15](#_Toc375150279)

[*Шаблон проектирования MVC* 15](#_Toc375150280)

[3.3 Обоснование выбора средств разработки приложения 16](#_Toc375150281)

[*Причины для создания приложения RoR MVC* 16](#_Toc375150282)

[*Выбор СУБД* 16](#_Toc375150283)

[*Преимущества языка Ruby* 17](#_Toc375150284)

[*Выбор языка программирования и платформы разработки* 18](#_Toc375150285)

[3.4 Спецификация функциональных требований 18](#_Toc375150286)

[*Аутентификация и авторизация пользователей* 18](#_Toc375150287)

[*Создание рекламных объявлений* 19](#_Toc375150288)

[3.5 Пользователи системы и их роли 20](#_Toc375150289)

[3.6 Взаимодействие между клиентом и сервером 20](#_Toc375150290)

[*Протокол HTTP* 20](#_Toc375150291)

[*Сервер приложения WEBrick* 21](#_Toc375150292)

[3.7 Диаграмма классов сущностей проекта 21](#_Toc375150293)

[4 ИНФОЛОГИЧЕСКАЯ МОДЕЛЬ ПРЕДМЕТНОЙ ОБЛАСТИ 24](#_Toc375150294)

[5 ОПИСАНИЕ БИЗНЕС-ЛОГИКИ ПРИЛОЖЕНИЯ 30](#_Toc375150295)

[5.1 Схема данных приложения «Рекламные объявления» 30](#_Toc375150296)

[6 ТЕСТИРОВАНИЕ ПРИЛОЖЕНИЯ 31](#_Toc375150297)

[6.1 Функциональное тестирование 31](#_Toc375150298)

[7 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 34](#_Toc375150299)

[ЗАКЛЮЧЕНИЕ 37](#_Toc375150300)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 38](#_Toc375150301)

[ПРИЛОЖЕНИЕ А 39](#_Toc375150302)

[Программный код создания базы данных приложения 39](#_Toc375150303)

[Исходный код модели объявлений 44](#_Toc375150304)

[Исходный код модели пользователей 46](#_Toc375150305)

[Исходный код некоторых контроллеров 48](#_Toc375150306)

[Исходный код javascript на сайте 51](#_Toc375150307)

# ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

ЛКМ – левая кнопка мыши

ПКМ – правая кнопка мыши

ПС – программное средство

ЯП – язык программирования

ОС – операционная система

MVC – Model-View-Controller

БД – база данных

TDD – Test Driven Development

# ВВЕДЕНИЕ

Во все времена в обществе существовали рыночные отношения, и всегда они являлись двигателем прогресса и процветания. Появление новых технологий (деньги, бумага, радио, телевидение) сразу же использовалось на рынке и, зачастую, было одним из решающих переломных моментов в его развитии. Последние десятилетия исследователи однозначно сходятся во мнении, что наиболее значимым достижением и открытием этих лет является глобальная сеть Интернет. И, конечно же, такое ключевое явление не могло не повлиять на рынок в целом и на рыночные отношения между людьми в частности. Проявляется это в огромном количестве аспектов. Интернет быстро стал, пожалуй, самым значимым средством коммуникации людей, а так как в основе рынка лежат отношения между людьми, вполне естественно то, что эти отношения перешли в Интернет. Уже в первые годы развития сетей, подобных Интернет, появилось огромное количество платформ для купли-продажи. Одной из ключевых составляющих частей этих сервисов является предоставление им возможности размещать на сайте и своих персональных страницах цифровые объявления, которые могут комментировать другие пользователи. Это способствует еще большему социальному объединению людей и, следовательно, развитию рынка.

Одной из первых таких платформ стала BBS ([англ.](http://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Bulletin Board System — электронная [доска объявлений](http://ru.wikipedia.org/wiki/%D0%94%D0%BE%D1%81%D0%BA%D0%B0_%D0%BE%D0%B1%D1%8A%D1%8F%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B9)). Широко используемый во времена редкости кабельных [компьютерных сетей](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C) способ общения пользователей компьютеров через коммутируемые телефонные сети. Изначально BBS содержали разнообразные фирмы и доступ к ним был платный. Впоследствии же, в качестве хобби, их стали открывать и частные лица. В основной своей массе доступ к частным BBS был бесплатным [1].

В основе подобных ресурсов лежит понятие рекламы. Развитие рекламы привело к тому, что рекламная деятельность трансформировалась в особый социальный институт, который обеспечивает общественную потребность в рекламных услугах. Производственную основу этого института составляет комплекс деятельностей, который принято определять понятием «индустрия рекламы». Понятие «индустрия рекламы» стало формироваться в современной экономике с приобретением рекламной деятельностью массового характера. Планомерное ведение рекламной деятельности, системное взаимодействие субъектов рынка рекламы с участниками различных секторов экономики, наличие предприятий, производящих рекламные продукты и оказывающих рекламные услуги, позволяет предположить, что рекламная деятельность приобрела черты индустрии [2]. И эта индустрия испытывает очень сильное влияние интернета, в то же время изменяя его обличие, ведь очень многие сайты и сервисы получают большую часть своего дохода из рекламы.

Результатом выполнения данной работы станет высокопроизводительное, надежное и безопасное программное средство с эргономичным интерфейсом, которое позволит пользователю создавать учетные записи, объявление, загружать изображения на сервер, комментировать объявления. Будет также предусмотрена возможность быстрой аутентификации через социальные сети ВКонтакте, Facebook, Google Plus без необходимости ввода своих данных.

# 1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

## **Анализ литературных источников**

Сильнейшим орудием неценовой конкуренции всегда была реклама, сегодня же роль ее возросла многократно. С помощью рекламы фирмы и обычные люди не только доносят до покупателей информацию о потребительных свойствах своих товаров, но и формируют доверие к своей товарной, ценовой и сбытовой политике, стремясь создать образ фирмы как «хорошего гражданина» той страны, на рынке которой предприниматель выступает во внешней торговле. Но рекламу нужно правильно преподнести, чтобы она не оттолкнула покупателя, а выполнила изначально задуманную ей роль. А в настоящее время, с ростом скорости обмена информацией, необходимость использования специализированных средств, которые бы представили данные в более удобном и легком к пониманию виде, растет с каждым днем. Создание электронной доски объявлений не является исключением. Большое внимание требуется уделить пользовательскому интерфейсу, с которым постоянно будет взаимодействовать пользователь. Обеспечение максимально удобного взаимодействия с информацией, поступающие из цифровых источников является очень важным моментом при проектировании программного средства.

Для упрощения и ускорения понимания окружающих процессов, люди начали использовать всевозможные средства автоматизации действий. К ним мы относим различные языки программирования, решающие за нас поставленные проблемы.

Безусловно, использование средств программирования и веб-технологий вместе, позволяло данным переходить из одного представления в другое, давая возможность конечному пользователю выбор в том, как он хочет видеть конечный результат.

Для того, чтобы увидеть веб-страницу, используются специальные программы, которые называются браузерами или, говоря простым языком, обозреватели. Они созданы, чтобы распознавать код HTML-документа и представлять его в обычном для нас виде.

Все приложения браузеров были разработаны с учетом всех мировых стандартов языка гипертекстовой разметки. Так веб-дизайнеры добились того, что все сайты стали выглядеть одинаково, не зависимо от того, каким браузером они были открыты и просмотрены.

Хотя можно сказать, что это касается не всех моментов. Например, если вы пишете страницу редким шрифтом, который поддерживает ваш браузер, но не знаете, поддерживают ли его другие обозреватели, то лучше в коде указывать несколько возможных вариантов шрифта, не искажающих внешний вид страницы [3].

В современном Интернете все виртуальные доски объявлений можно разделить на несколько типов и подтипов. Самым главным критерием доски является возможность платного или бесплатного размещения объявления. Большинство популярных крупных высоко посещаемых досок используют платный тип размещения объявлений, то есть пользователь размещает объявление, а потом вносит фиксированную сумму для того, чтобы объявление было опубликовано. При этом более мелким сайтам, для того чтобы выдержать конкуренцию за посетителя, приходится публиковать объявления бесплатно.

Немаловажной проблемой является также работа базы данных с веб-приложением. Для ее решения будет использоваться ORM – технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных» [4]. Другими словами, если администрация сайта решит сменить СУБД на другую, более производительную, то это нисколько не повлияет на производительность веб-приложения, поскольку использование данной технологии позволяет абстрагироваться от понятий реляционных баз данных.

## **Анализ аналогов**

В качестве первого аналога было решено выбрать онлайн площадку для размещения объявлений «Kufar.by» и более детально рассмотреть ее преимущества и недостатки, т.к. составной частью данного интернет ресурса является возможность создавать и просматривать объявления. На данный ресурс будет осуществляться основная ориентация приложения и преодоление его недостатков. К преимуществам сервиса можно отнести:

1. Возможность создания аккаунтов на сайте.
2. Тесная интеграция с различными использующимися на территории Беларуси социальными сетями (Вконтакте, Одноклассники, Фейсбук). Она заключается в возможности вступления в группы, возможности поделиться объявлением на своей странице, установкой лайков к понравившемся объявлениям.
3. Широкие возможности использования сервиса без регистрации. Для этого нужно ввести пароль после добавления объявления, и в следующий раз, когда нужно будет его отредактировать или провести другую операцию, требующую исключительных прав, необходимо будет только ввести пароль.
4. Бесплатность сервиса.

Однако данному интернет-порталу присущи следующие недостатки:

1. Недостаточно гибкий поиск.
2. Отсутствие в самом сервисе возможностей для социального взаимодействия.
3. Для всех пользователей присутствуют ссылки для доступа к персональным ресурсам, которые в конечном итоге никуда не ведут.
4. Отсутствует возможность аутентификации через социальные сети.

Вторым веб-ресурсом для размещения объявлений является Дорус.ру. Здесь размещаются объявления от организаций и частных лиц со всей России. На сайте присутствует удобная функция сортировки и поиска объявлений. Подача объявления на сайте достаточно проста: заполняете небольшую форму с описанием вашего предложение и прикрепляете фото, если нужно. Все объявления разбиты по категориям разнообразной тематики: от обуви и одежды до недвижимости, что облегчает поиск нужного вам предложения. Данный портал обладает следующими преимуществами:

1. Объявления принимаются как от частных лиц, так и от юридических
2. VIP-выделения для объявлений (есть возможность выделить объявление бесплатно)
3. Размещение объявления не потребует лишних регистраций на сайте.
4. Интерфейс сайта достаточно простой и понятный.
5. На сайте присутствует небольшой раздел помощи с советами как составить объявление, и советами по правильной продаже.

Однако помимо наличия явных преимуществ, ресурсу присущи следующие недостатки:

1. Большое количество сторонней рекламы, которая не только отвлекает пользователя, но также негативно влияет на восприятие интересующей информации.
2. Поиск средствами поисковика Яндекс, который невозможно настроить, что не дает необходимой гибкости при специализированных настройках пользователя.

Большой популярностью в Беларуси пользуется еще один ресурс, который, несмотря на это, спроектирован и сделан не на самом высоком уровне. Данному ПС присущи следующие достоинства:

1. Редактированный ввод контента объявления.
2. Интеграция с социальными сетями.
3. Широкая распространенность основного ресурса, к которому примыкает данный, что дает ему преимущество по сравнению с остальными.

При более детальном рассмотрении были обнаружены следующие существенные недостатки:

1. Неудобный поиск по категориям.
2. Неэргономичный интерфейс.
3. Только одна валюта, что вносит неразбериху при поиске необходимого объявления, так как пользователь может указать цену, в валюте, в которой он захочет, несмотря на программные ограничения.

В качестве аналога также было решено рассмотреть интернет-сервис irr.by. Данное программное средство обладает широким спектром преимуществ, некоторые из которых перечислены ниже:

1. Возможность подачи объявлений в печатные издания напрямую
2. Платные и бесплатные объявления
3. Премиум-объявления
4. Сервис для компаний — «Интернет-партнер»
5. Мобильное приложение для iPhone и iPod Touch
6. Раздел объявлений по поиску работы

Тем не менее, данный сервис характеризуется следующим набором отрицательных черт:

1. Большое количество навязчивой рекламы.
2. Низкая скорость работы сайта.
3. Большое количество багов на сайте.

Каждый из вышеперечисленных аналогов обладает как существенными преимуществами, так и недостатками, которые способствуют снижению их популярности среди пользователей не только ввиду неудобства использования, а также морального вреда, который данные ресурсы могут нанести другим людям. Разрабатываемое ПС призвано взять самое лучшее из рассмотренных ПС и внести ряд новых черт, которые увеличат удобство использования и функциональные возможности веб-приложения.

## **1.3 Спецификация требований**

**1.3.1** Расширяемость. Одним из основных требований, которому будет удовлетворять данное программное средство, является возможность написания дополнительных модулей. Выполнение рассматриваемого требования поможет в будущем значительно расширить функциональность программного продукта за счет добавления новых функциональных особенностей.

**1.3.2** Надежность. Крайне важным фактором при проектировании данного программного продукта является требование к надежности и отказоустойчивость.

**1.3.3** Скорость работы. Выполнение заданных пользователем операций должна осуществляться с максимально возможной скоростью ввиду того, данное ПС служит помощником для пользователя. Поэтому оптимизация работы алгоритмов должна являться одним из ключевых требований к разрабатываемому продукту.

**1.3.3.1** Многопоточность. Большинство современных мобильных процессоров имеет более одного ядра или, по крайней мере, поддерживает так называемую одновременную многопоточность.

Создание дополнительного потока для работы приложения может значительно повысить производительность ПС на многопроцессорных/многопоточных устройствах.

**1.3.4** Практичность. Интерфейс программного продукта должен быть дружелюбен и интуитивно понятен пользователям. Данное требование предполагает грамотную организацию пользовательских элементов, правильно подобранную цветовую схему и отсутствие ненужных элементов, препятствующих доступу к ключевым функциональным возможностям ПС.

**1.3.5** Безопасность. Ввод пользовательской информации должен осуществляться таким образом, чтобы третья сторона ни при каких условиях не могла её узнать.

**1.3.6** Варианты использования ПС. Разрабатываемый продукт предназначен для всеобщего использования при размещение продукта в сети интернет. Программная разработка должна предоставлять пользователям возможность по аутентификации и авторизации, возможности создания своего аккаунта и создание своих объявлений. Помимо этого, ПП должен обеспечивать возможность оставления комментариев к объявлениям и поиск объявлений по ключевым словам, разделам, подразделам, содержанию объявления и даже по имени автора а также подписку на создание альбомов и размещение фотографий.

**1.3.7** Входные данные. В качестве входных данных будет использоваться учетная информация о пользователе – его логин, представляющий собой почтовый ящик, и пароль. Помимо этого, входными данными являются фотографии, которые будут размещаться на сервере пользователем, и комментарии, оставляемые к объявлениям.

**1.3.8** Выходные данные. В качестве выходных данных будут выступать веб-страницы, отображающие пользовательские данные – его данные и часть публичных учетных данных. Кроме этого, на страницах будет содержаться информация о созданных объявлениях, оповещениях и загруженных фотографиях, оставленных комментариях. Все вышеперечисленное будет отражать основную функциональную составляющую программы.

**1.3.9** Требования к составу и параметрам технических и программных средств. Программное средство должно представлять собой набор программных модулей, объединенных в рабочее приложение, также включается база данных. Для разворачивания приложения на стороне сервера необходимо наличия сервера и сервера базы данных с настроенными пулами приложений, строкой подключения и пр.

**1.3.10** Требования к информационной и программной совместимости.

1. Работа веб-приложения должна осуществляться под управлением операционных систем семейства Linux Ubuntu/Debian/Fedora, MacOS.
2. Требования к серверу (желательно): CPU Xenon 2.2GHz (4 ядра), RAM 16 Gb, HDD 16 Tb с установленной ОС Linux и Webrick.

## **1.4 Постановка задачи**

В ходе выполнения данного курсового проекта должны быть реализованы следующие технические требования и пользовательские функции:

1. Веб-приложение для просмотра рекламных объявлений.
2. Авторизация пользователей на сайте. В основу аутентификации и авторизации положить аутентификацию на основе форм, модифицированную для собственных нужд.
3. Возможность создания и удаления объявлений на сайте авторизованным пользователям.
4. Предусмотреть возможность разделения ролей на сайте на администратора, зарегистрированного пользователя и гостя.
5. Возможность закачивания на сервер фотографий и их удаление.
6. Возможность размещения комментариев к собственным объявлениям и объявлениям других пользователей.
7. Возможность получения пользователем оповещений на комментирование его объявлений, а также на изменение статуса объявления (опубликовано, принято админом и т. д.).
8. Осуществление поиска альбомов по ключевым словам.
9. Использование в качестве хранилища данных – базу данных, обеспечивающую надежность и целостность работы с пользовательской информацией.

# МАТЕМАТИЧЕСКИЕ МОДЕЛИ И МЕТОДЫ

## **2.1 Система защиты данных OpenSSL**

В базе данных проекта есть таблица с хэш-ключами, необходимая для проверки корректности ввода с карт оплаты. Но в такой ситуации возможна подделка ключей. Чтобы этого не допустить, используются криптографически надежные генераторы случайных чисел. В стандартной библиотеке Ruby используется OpenSSL.

OpenSSL — это система защиты и сертификации данных, название SSL переводится, как система безопасных сокетов (secure socket layer). OpenSSL используется практически всеми сетевыми серверами для защиты передаваемой информацией. Существует программное API SSL (SSLEAY), позволяющее создавать безопасные сокеты с шифрацией передаваемых данных в собственных разработках. Но в данной статье я бы хотел рассказать о самой системе OpenSSL, вызываемой через командную строку.

Т.к. OpenSSL поддерживает очень много различных стандартов сертификации, шифрования, хеширования, то использование данной команды достаточно сложно. Внутри OpenSSL существуют отдельные компоненты, отвечающие за то или иное действие, для получения списка доступных компонентов можно вызвать openssl с параметрами list-standart-commands. Можно также получить список доступных алгоритмов хеширования (list-message-digest-commands) и алгоритмов шифрования (list-cipher-commands).

Доступна для большинства [UNIX](http://ru.wikipedia.org/wiki/UNIX)-подобных операционных систем (включая [Solaris](http://ru.wikipedia.org/wiki/Solaris)/[OpenSolaris](http://ru.wikipedia.org/wiki/OpenSolaris), [Linux](http://ru.wikipedia.org/wiki/Linux), [Mac OS X](http://ru.wikipedia.org/wiki/Mac_OS_X), [QNX4](http://ru.wikipedia.org/wiki/QNX), [QNX6](http://ru.wikipedia.org/wiki/QNX) и четырех операционных систем [BSD](http://ru.wikipedia.org/wiki/BSD) с открытым исходным кодом), а также для [OpenVMS](http://ru.wikipedia.org/wiki/OpenVMS) и [Microsoft Windows](http://ru.wikipedia.org/wiki/Microsoft_Windows) [5].

# 3 АНАЛИЗ ТРЕБОВАНИЙ К ПРОГАММНОМУ СРЕДСТВУ И РАЗРАБОТКА ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ

Разрабатываемый программный проект будет представлять собой web-приложение для создания и просмотра рекламных объявлений.

## **3.1 Анализ средств разработки и программных продуктов в сфере интернет**

Для правильной работы сайтов на всех браузерах была введена строгая стандартизация во всех разработки веб приложений (HTTP, HTML, CSS, JavaScript). Создание современного сайта невозможно без скриптового языка JavaScript. JavaScript используется в клиентской части веб-приложений: клиент-серверных программ, в котором клиентом выступает браузер, а сервером — веб-сервер, имеющих распределённую между сервером и клиентом логику. Обмен информацией в веб-приложениях происходит по сети. Одним из преимуществ такого подхода является тот факт, что клиенты не зависят от конкретной операционной системы пользователя, поэтому веб-приложения являются кроссплатформенными сервисами [6].

Для того, чтобы работа с сайтом была максимально удобной, чтобы на странице не было лишних загрузок и обновлений, JavaScript используется в [AJAX](http://ru.wikipedia.org/wiki/AJAX), популярном подходе к построению интерактивных пользовательских интерфейсов веб-приложений, заключающемся в «фоновом» асинхронном обмене данными браузера с веб-сервером. В результате, при обновлении данных веб-страница не перезагружается полностью и интерфейс веб-приложения становится быстрее, чем это происходит при традиционном подходе (без применения AJAX).

Изначально различные [браузеры](http://ru.wikipedia.org/wiki/%D0%91%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80) имели собственные модели документов (DOM), несовместимые с остальными. Для обеспечения взаимной и обратной совместимости, специалисты международного консорциума [W3C](http://ru.wikipedia.org/wiki/W3C) классифицировали эту модель по уровням, для каждого из которых была создана своя спецификация. Все эти спецификации объединены в общую группу, носящую название W3C DOM.

Модель DOM не накладывает ограничений на структуру документа. Любой документ известной структуры с помощью DOM может быть представлен в виде дерева узлов, каждый узел которого представляет собой элемент, атрибут, текстовый, графический или любой другой объект. Узлы связаны между собой отношениями "родительский-дочерний".

Все эти технологии в связке дают почти неограниченные возможности в построении веб-приложений.

## **3.2 Общие сведения о RoR MVC**

### *Шаблон проектирования MVC*

Model-view-controller (MVC, «модель-представление-поведение», «модель-представление-контроллер», «модель-вид-контроллер») — схема использования нескольких [шаблонов проектирования](http://ru.wikipedia.org/wiki/%D0%A8%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), с помощью которых [модель данных](http://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) приложения, [пользовательский интерфейс](http://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D0%BA%D0%B8%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81) и взаимодействие с пользователем разделены на три отдельных компонента так, что модификация одного из компонентов оказывает минимальное воздействие на остальные. Данная схема проектирования часто используется для построения [архитектурного каркаса](http://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D1%80%D0%BA%D0%B0%D1%81_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), когда переходят от теории к реализации в конкретной предметной области [7].



Рисунок 3.2.1 Шаблон проектирования MVC

Основная цель применения этой концепции состоит в разделении бизнес-логики (модели) от её визуализации (представления, вида). За счет такого разделения повышается возможность [повторного использования](http://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%B2%D1%82%D0%BE%D1%80%D0%BD%D0%BE%D0%B5_%D0%B8%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5). Наиболее полезно применение данной концепции в тех случаях, когда пользователь должен видеть те же самые данные одновременно в различных контекстах и/или с различных точек зрения. В частности, выполняются следующие задачи:

1. К одной модели можно присоединить несколько видов, при этом не затрагивая реализацию модели. Например, некоторые данные могут быть одновременно представлены в виде электронной таблицы, гистограммы и круговой диаграммы.
2. Не затрагивая реализацию видов, можно изменить реакции на действия пользователя (нажатие мышью на кнопке, ввод данных), для этого достаточно использовать другой контроллер.
3. Ряд разработчиков специализируется только в одной из областей: либо разрабатывают графический интерфейс, либо разрабатывают бизнес-логику. Поэтому возможно добиться того, что программисты, занимающиеся разработкой бизнес-логики (модели), вообще не будут осведомлены о том, какое представление будет использоваться.

## **3.3 Обоснование выбора средств разработки приложения**

### *Причины для создания приложения RoR MVC*

Для разработки программного продукта была выбрана технология RoR MVC и язык Ruby, являющийся единственным языком для написания бизнес логики на платформе RoR.

Rails – это платформа, которая полностью поддерживает MVC (модель, просмотр, контроллер). Это очень продуктивная и эффективная Ruby-платформа, распространяющаяся с открытым исходным кодом, предназначенная для разработки веб-приложений, подкрепленных базами данных. Главное, что следует выделить в Rails, заключается в возможности разработки очень продуктивных приложений при написании меньшего по объемам кода. Также многие разработчики выделяют возможность интеграции кода с базой данных.  
 Ранее уже говорилось о том, что Ruby on Rails позиционируется как самая тщательно продуманная платформа для разработки. Полная поддержка MVC является основным преимуществом, которая выделяет Ruby on Rails среди других платформ для веб-разработки. Несмотря на то, что сегодня многие разработчики заявляют о том, что Ruby on Rails работает медленнее, чем PHP или Python, важно помнить о том, что главное – продуктивность, а не производительность.

### *Выбор СУБД*

Так как в настоящее время на платформе Ruby on Rails используются и активно поддерживаются только две СУБД с реляционными базами данных (MySQL и PostgreSQL), то выбор осуществлялся среди этих двух решений. На окончательный выбор в сторону PostgreSQL повлияли следующие факторы:

1. Поддержка транзакций в MySQL пока что не настолько хорошо проверена, как в системе PostgreSQL.
2. Так как MySQL основан на использовании потоков (threads), пока что еще не безошибочно работающих в некоторых ОС, для обеспечения стабильной работы приходится либо использовать один из откомпилированных пакетов, либо точно выполнять инструкции, что не всегда удобно.
3. Блокировка таблиц, применяющаяся в нетранзакционных таблицах MyISAM, во многих случаях работает быстрее, нежели блокировки на уровне страниц, строк или контроль версий. Недостаток этого подхода в том, что если не учитывать механизм работы блокирования таблиц, один длительный запрос может надолго заблокировать таблицу. Обычно этого эффекта можно избежать, приняв соответствующие меры при разработке приложения. Если это не удастся, всегда можно изменить тип таблицы и сделать ее транзакционной.
4. При помощи UDF (user-defined functions, определяемые пользователем функции) возможности MySQL можно расширить и дополнить обычными SQL-функциями или их объединениями. Но это сделать не так просто, да и система не настолько гибка в этом отношении, как PostgreSQL.
5. В MySQL сложнее организовывались обновления, затрагивающие несколько таблиц сразу. Впрочем, это было исправлено в MySQL 4.0.2 реализацией многотабличного UPDATE и в MySQL 4.1 - с помощью подзапросов. В MySQL 4.0 можно одновременно удалять данные из нескольких таблиц [8].

### *Преимущества языка Ruby*

1. Ruby очень гибкий язык, так как он позволяет его пользователям свободно менять его части. Основные части Ruby могут быть удалены или переопределены по желанию. А существующие части можно модифицировать. Ruby старается ни в чём не ограничивать пользователя.
2. В Ruby представлен настоящий mark-and-sweep (пометь и отчисти) сборщик мусора для всех Ruby объектов. Не нужно вручную отслеживать количество ссылок в сторонних библиотеках.
3. Писать расширения на C в Ruby проще чем в Perl или Python при помощи очень элегантного API для вызова Ruby из C. Он включает в себя вызовы для встраивания Ruby в программное обеспечение, чтобы использовать его как скриптовый язык. Также доступен интерфейс SWIG.
4. В Ruby реализованы независимые от операционной системы потоки. Таким образом, на любых платформах, где вы запускаете Ruby, вы также имеете возможность использовать многопоточность, не зависимо от того, поддерживает ли данная система потоки или нет.
5. Ruby отличается высокой переносимостью: он был разработан большей частью на GNU/Linux, но работает на многих типах UNIX, Mac OS X, Windows 95/98/Me/NT/2000/XP/Vista/8, DOS, BeOS, OS/2, и так далее [9].

### *Выбор языка программирования и платформы разработки*

Разработку программного продукта было решено осуществлять на платформе Ruby on Rails с использование языка Ruby. В качестве альтернатив рассматривался язык C# и платформа ASP.NET.

ASP.NET MVC превратился в своего рода образец, с которым следует сравнивать другие веб-платформы. Суровая реальность состоит в том, что разработчики и компании, работающие в мире Microsoft .NET, сочтут более простой в адаптации и изучении платформу ASP.NET MVC, в то время как компании, работающие на Phython или Ruby в Linux и Mac OS X, отдадут предпочтение  Rails. Маловероятно, что понадобится выполнять переход от Rails на ASP.NET MVC или наоборот. Между этими двумя технологиями существуют значительные различия в области применения.

Rails - это полностью целостная платформа разработки, в том смысле, что она охватывает весь стек - от управления исходными базами данных (миграциями) до объектно-реляционного отображения (ORM), обработки запросов с помощью контроллеров и действий, а также построения автоматизированных тестов. В общем, Rails представляет собой самодостаточную систему быстрой разработки приложений, ориентированных на данные.

В противоположность этому, платформа ASP.NET MVC сосредоточена исключительно на задаче обработки веб-запросов в стиле MVC с помощью контроллеров и действий. Она не имеет ни встроенного инструмента ORM, ни встроенного инструмента модульного тестирования, ни системы управления миграциями баз данных - все это, а также многое другое предлагает платформа .NET, и вам останется только выбор.

Например, в качестве инструмента ORM можно использовать NHibernate, Microsoft LINQ to SQL, Subsonic или любое из других зрелых решений. В этом и состоит роскошь платформы .NET, хотя это также означает, что упомянутые компоненты не могут быть настолько тесно интегрированы с ASP.NET MVC, как их эквивалент в Rails [10].

## **3.4 Спецификация функциональных требований**

В разрабатываемом программном средстве должны быть реализованы следующие функции.

### *Аутентификация и авторизация пользователей*

1. Аутентификация пользователя осуществляется на основании ранее созданного аккаунта. В качестве учетных данных используется имя почтового ящика и пароль к нему.
2. Для создания аккаунта необходимо указать логин, представляющий собой уникальный почтовый ящик пользователя, имя, фамилию и пароль.
3. Реализация собственной системы проверки учетных данных и вывода результатов авторизации пользователю на экран.
4. Модифицированная для нужд данного приложения аутентификация и авторизация на основе форм.
5. Осуществлять шифрование куки, хранящихся на клиентской стороне, и проверять их наличия и состав при каждом запросе к серверу.
6. Администратор имеет доступ к дополнительной части сайта, в которой имеет возможность изменять статус объявлений, удалять пользователей, комментарии и объявления. Также администратор может установить роль пользователя в значение администратор.
7. Аутентификация через наиболее популярные социальные сети (Вконтакте, Facebook, Google Plus).

### *Создание рекламных объявлений*

1. Предусмотреть возможность создания пользователями рекламных объявлений, представляющих собой текст объявления, дополнительные параметры (место размещения, тип объявления, раздел и подраздел, в котором будет располагаться объявление).
2. Если пользователь не указывает дополнительные параметры, они устанавливаются по умолчанию.
3. Если в объявлении присутствуют фотографии, они должны отображаться под его содержимым, а при нажатии на фотографию левой кнопкой мыши должна открываться анимированная галерея всех изображений на странице.
4. Возможность добавления ключевых слов, которые помогут не только при поиске объявления, но также и при уточнении деталей при его просмотре.
5. Предусмотреть возможность редактирования у удаления рекламных объявлений владельцем этого объявления и пользователями с правами администратора.
6. Автоподсказки для создания ключевых слов, для введения раздела объявления.
7. Поиск объявлений организовать, использую AJAX для их подгрузки.

## **3.5 Пользователи системы и их роли**

Практически каждое программное средство характеризуется наличием определенного уровня доступа, представляемого пользователями системы. Разрабатываемое программное средство не является исключением. В интернет-фотоальбоме предусмотрены следующие пользователи:

1. Незарегистрированный пользователь. Имеет возможность просматривать объявления, искать их по ключевым словам или заданным параметрам. Не имеет доступа к аккаунтам пользователей.
2. Зарегистрированный пользователь. Является основной разновидность единицы аутентификации и авторизации на сайте. Для использования всех возможностей сервиса необходимо зарегистрироваться. После этого пользователь получит возможность создания объявлений и их редактирования, загрузки изображений на сервер и изменения их описания. Помимо этого, пользователь будет обладать возможностью оставлять комментарии к объявлениям других пользователей и осуществлять поиск по интернет-порталу.
3. Администратор. Роль, представляющая наибольшие функциональные возможности посетителям сайта. Администратор обладает возможностью удалять комментарии и объявления других пользователей, если они не соответствуют моральным и этическим нормам, а также оскорбляют, прямым образом или косвенно, интересы и чувства посетителей ресурса. Также администратор может и должен принимать на публикацию объявления, может их удалять, устанавливать роль пользователей в значение администратор.

## **3.6 Взаимодействие между клиентом и сервером**

### *Протокол HTTP*

Ключевым протокол взаимодействия в глобальной сети Интернет является HTTP. Именно поэтому в курсовом проекте было решено использовать именного его. HTTP — протокол прикладного уровня передачи данных (изначально — в виде гипертекстовых документов в формате HTML) [1]. Основой HTTP является технология «клиент-сервер», то есть предполагается существование потребителей (клиентов), которые инициируют соединение и посылают запрос, и поставщиков (серверов), которые ожидают соединения для получения запроса, производят необходимые действия и возвращают обратно сообщение с результатом. Основным объектом манипуляции в HTTP является ресурс, на который указывает URI в запросе клиента. Обычно такими ресурсами являются хранящиеся на сервере файлы, но ими могут быть логические объекты или что-то абстрактное.

HTTP присущи следующие преимущества:

1. Простота. Протокол прост в реализации, что позволяет легко создавать клиентские приложения.
2. Расширяемость. Возможности протокола легко расширяются благодаря внедрению своих собственных заголовков, с помощью которых можно получить необходимую функциональность при решении специфической задачи. При этом сохраняется совместимость с другими клиентами и серверами: они будут просто игнорировать неизвестные им заголовки.
3. Распространенность. При выборе протокола HTTP для решения конкретных задач немаловажным фактором является его распространённость. Как следствие, это обилие различной документации по протоколу на многих языках мира, включение удобных в использовании средств разработки в популярные IDE, поддержка протокола в качестве клиента многими программами и обширный выбор среди хостинговых компаний с серверами HTTP [11].

### *Сервер приложения WEBrick*

WEBrick – это библиотека, написанная на языке программирования [Ruby](http://ru.wikipedia.org/wiki/Ruby) и предоставляющая набор базовых [HTTP](http://ru.wikipedia.org/wiki/HTTP)-сервисов. WEBrick была написана Takahashi Masayoshi в соавторстве с Gotou Yuuzou, а также при участии других разработчиков со всего мира, в соответствии с моделью [открытого программного обеспечения](http://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BA%D1%80%D1%8B%D1%82%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5). Библиотека также содержит код для создания базовых сервисов, работающих и по другим протоколам, нежели по [HTTP](http://ru.wikipedia.org/wiki/HTTP).

Библиотека используется [программным каркасом](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D1%8B%D0%B9_%D0%BA%D0%B0%D1%80%D0%BA%D0%B0%D1%81) [Ruby on Rails](http://ru.wikipedia.org/wiki/Ruby_on_Rails) для тестирования приложений в процессе их разработки. Несмотря на свою популярность, WEBrick заслуживает критики за полное отсутствие документации внутри своего кода.

## **3.7 Диаграмма классов сущностей проекта**

**3.7.1** Класс HashKey. Объекты этого класса хранят информацию о доступных для покупки хэш-ключах, которые необходимо вводить для пополнения счета, который в дальнейшем будет использоваться для осуществления платных услуг.

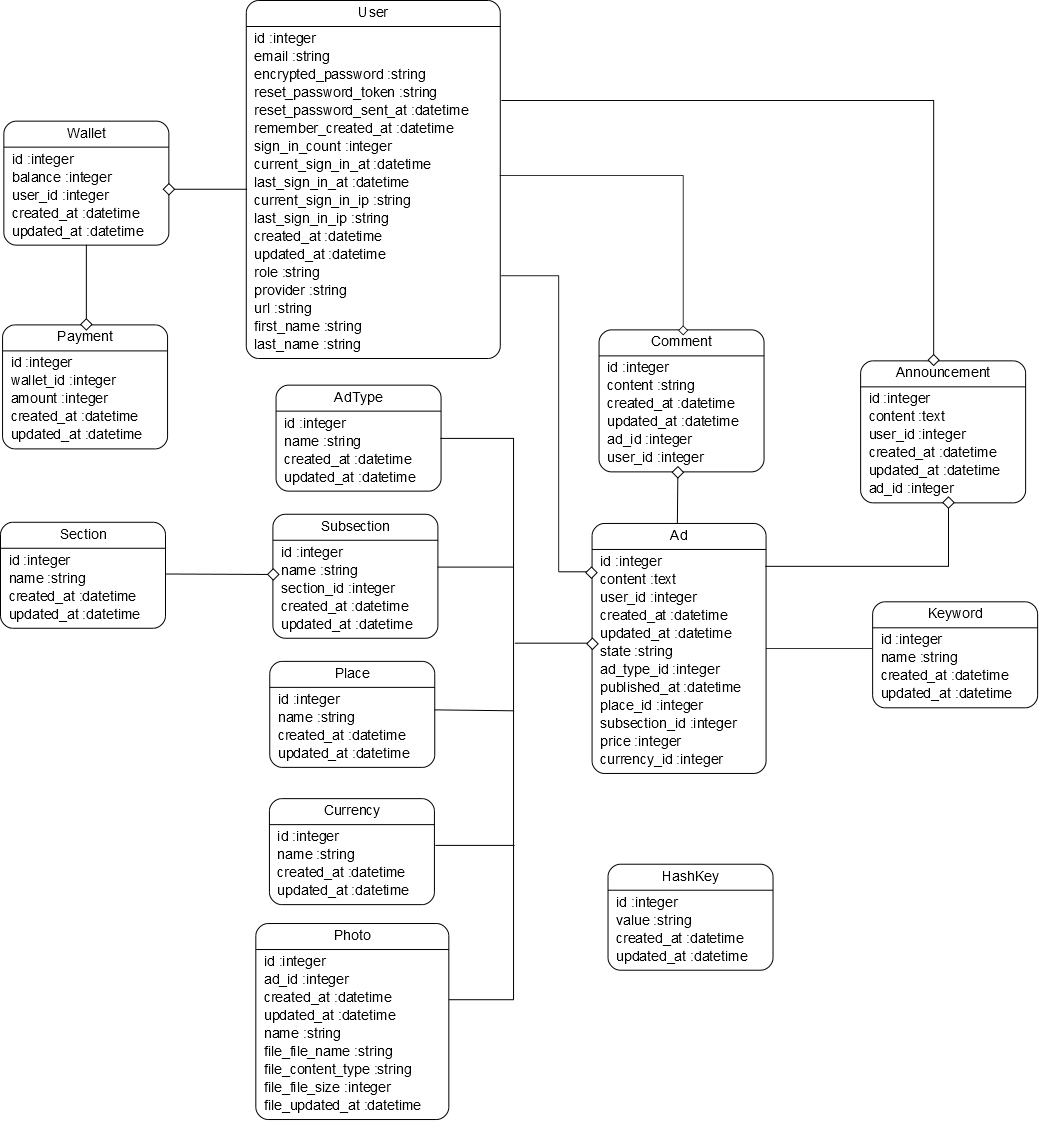


Рисунок 3.8.1 UML-диаграмма классов проекта

**3.7.2** Класс Keyword. Объекты этого класса хранят ключевые слова, по которым будет осуществляться поиск, связанные с объявлениями.

**3.7.3** Класс Currency. Служит для определения объекта, который будет хранить валюту.

**3.7.4** Класс Comment. Содержит в себе всю необходимую информацию, описывающую комментарий пользователя к определенному объявлению. Существует возможность оставлять комментарии к объявлениям зарегистрированными пользователями. Каждое объявление содержит в себе список всех комментариев, которые были оставлены пользователями для этого объявления.

**3.7.5** Класс Place. Необходим для хранения местоположения товара или услуги.

**3.7.6** Класс User. Служит описанием пользователя для данного веб-проекта. Единица авторизации и аутентификации. Каждому из пользователей при успешной регистрации выделяется свое полностью функционирующее хранилище фотографий.

**3.7.7** Класс AdType. Тип объявления. Предполагается, что там будут храниться объекты вида «покупка», «продажа», «обмен».

**3.7.8** Класс Ad. Объявление пользователя. Представляет собой такие атрибуты как место, тип, раздел и т.д., а также контейнер для фотографий, которые закачивает на сервер пользователь. Количество создаваемых объявлений ограничено только техническими характеристиками сервера.

**3.7.9** Класс Photo. Используется для хранения информации о фотографии, загружаемой пользователем на сервер и последующем ее отображении на странице веб-сайта.

**3.7.10.** Класс Section.Класс, использующийся для хранения раздела объявлений.

**3.7.11** Класс Subsection. Объекты этого класса хранят информацию о подразделах объявлений.

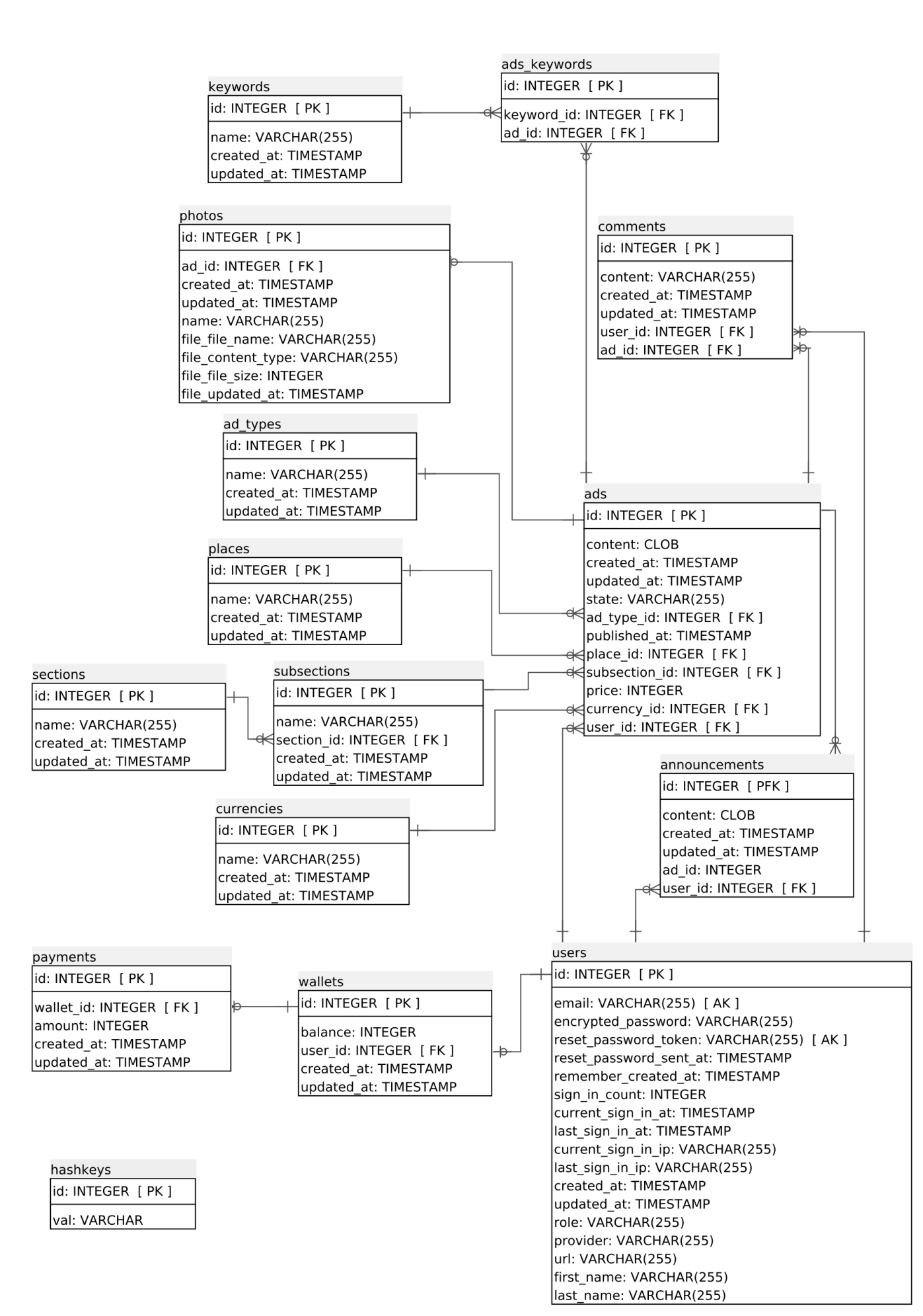
**3.7.12** Класс Wallet. Класс, характеризующий электронный кошелек пользователя, предназначенный для проведения операций по оплате услуг на сайте.

**3.7.13** Класс Payment. Содержит в себе базовую информацию проведенном платеже, будь то снятие средств с кошелька или перечисление средств на него.

**3.7.14** Класс Announcement. Класс, объекты которого хранят информацию об оповещениях пользователя. Это может быть извещение о комментировании его объявления, или извещение об изменении его состояния.

# 4 ИНФОЛОГИЧЕСКАЯ МОДЕЛЬ ПРЕДМЕТНОЙ ОБЛАСТИ

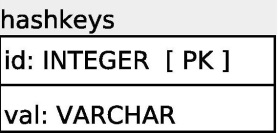
Рисунок 4.1 Схема базы данных проекта

****

В качестве хранилища пользовательской информации решено использовать базу данных СУБД PostgreSQL. На рисунке 4.1 представлена ее схема. Дамб структуры БД приведен в приложении A.

Сейчас будет рассмотрена и описана каждая из таблиц в отдельности.

**4.1** Таблица hashkeys. В этой таблице, как и во всех последующих, есть поле под названием id, имеющее тип integer. Также в каждой таблице, кроме последней, есть два поля типа timestamps, в которых хранится информация о времени создания и последнего обновления записи.



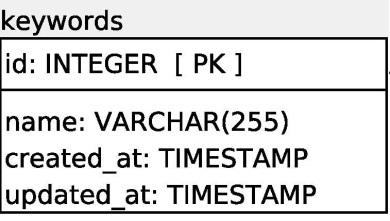
Поле value. Хранит хеш, сформированный надежным криптографическим алгоритмом.

В этой таблице, как и во всех последующих, есть поля:

Поле created\_at. Хранит время и дату создания объекта.

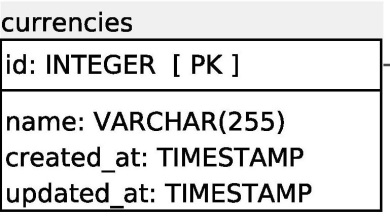
Поле updated\_at. Хранит время и дату редактирования объекта.

**4.2** Таблица keywords. Здесь присутствует поле name, которое будет встречаться еще в нескольких таблицах.



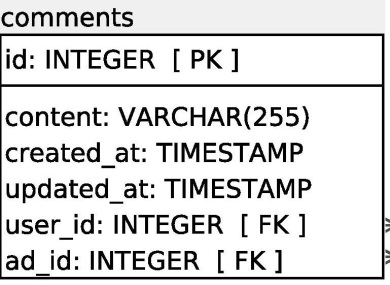
Поле name. Хранит название ключевого слова. Может содержать набор символов без пробела.

**4.3** Таблица currencies. Таблица связана отношением один ко многим с таблицей ads.



Поле name. Определяет имя или символьное представление валюты.

**4.4** Таблица comments. Хранит поля с идентификаторами объявления, которому принадлежит комментарий, и пользователя, который этот комментарий оставил.

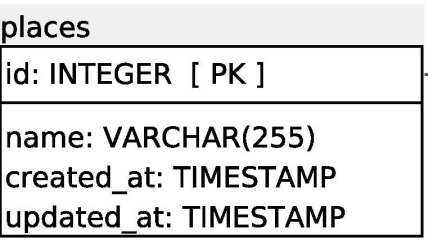


Поле content. Содержание комментария.

Поле ad\_id. Содержит в себе уникальный идентификатор объявления.

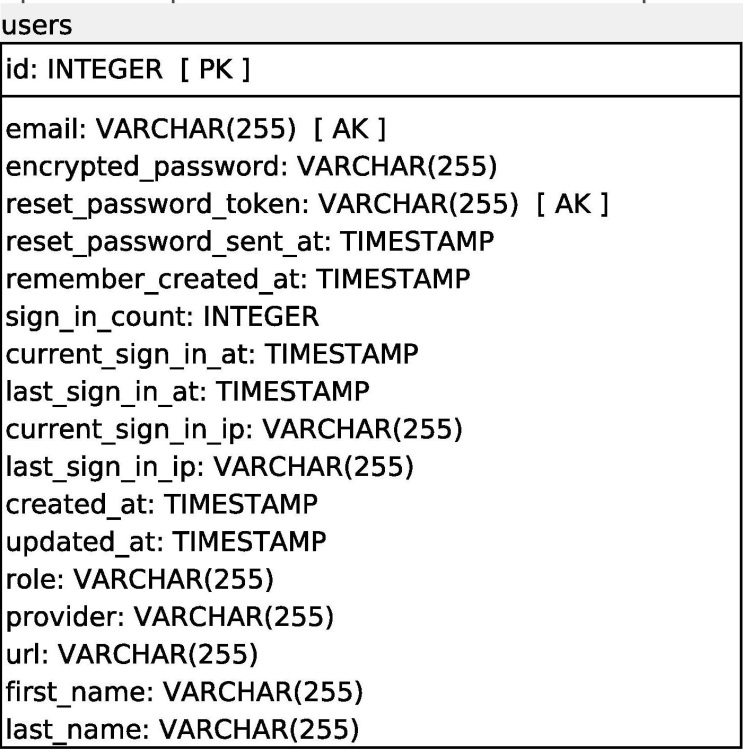
Поле user\_id. Содержит идентификатор пользователя, который оставил комментарий.

**4.5** Таблица places. Таблица связана отношением один ко многим с таблицей ads.



Поле name. Содержит непосредственно название местности.

**4.6** Таблица users. Хранит большую часть информации о пользователе, включая его роль, имя, пароль и т.д.



Поле first\_name. Содержит имя пользователя.

Поле last\_name. Содержит фамилию пользователя.

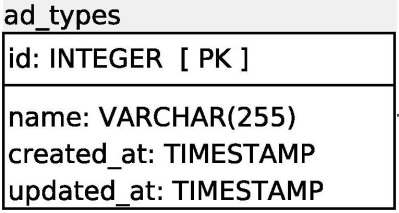
Поле email. Уникальный адрес электронной почты. Служит в качестве логина при аутентификации пользователя на сайте.

Поле encrypted\_password. Хеш-значение пароля пользователя вместе с солью. Необходимо для более тщательной защиты каждого из пользователей.

Поле role. Хранит роль пользователя.

Поле provider. Содержит название социальной сети, через которую был зарегистрирован пользователь.

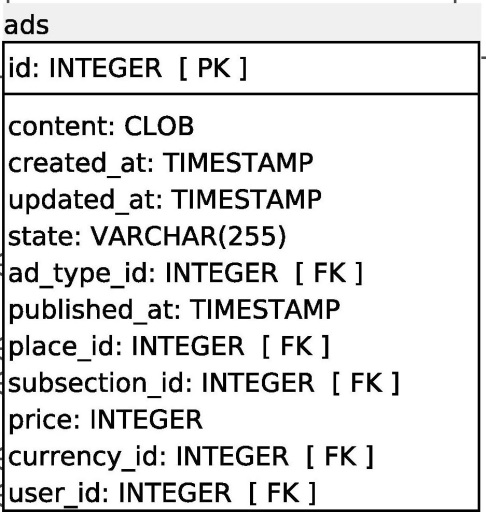
**4.7** Таблица ad\_types. Таблица связана отношением один ко многим с таблицей ads.



Поле name. Представляет собой имя типа объявления.

Поле lastName. Представляет собой фамилию пользователя.

**4.8** Таблица ads. Хранит записи объявлений.



Поле content. Содержание объявления.

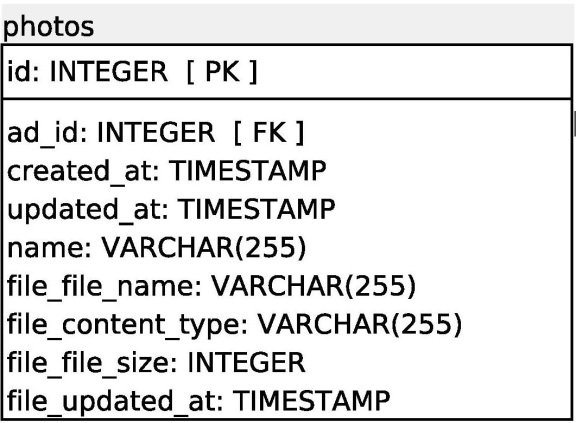
Поле user\_id. Идентификатор пользователя, создавшего объявление.

Поле state. Состояние объявления (черновик, на проверку, подтверждено и т.д.).

Поля ad\_type\_id, place\_id, subsection\_id, currency\_id. Идентификаторы соответствующих атрибутов объявления.

Поле price. Цена объявления.

**4.9** Таблица photos. Таблица связана отношением многие к одному с таблицей ads.



Поле ad\_id. Содержит в себе уникальный идентификатор объявления.

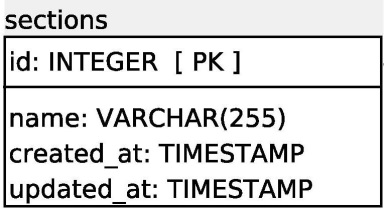
Поле name. Дает пользователю возможность ввести имя фотографии.

Поле file\_file\_name. Имя файла, загруженного пользователем.

Поле file\_content\_type. Тип файла.

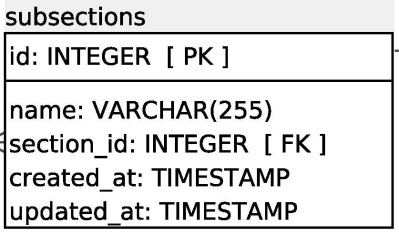
Поле file\_size. Размер файла.

**4.10.** Таблица sections. Таблица связана отношением один ко многим с таблицей subsections.



Поле name. Данное поле хранит имя раздела.

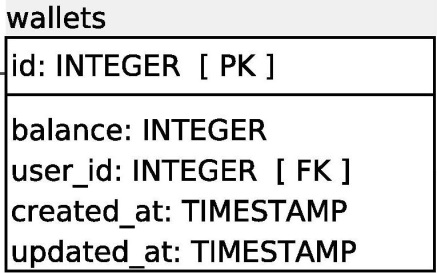
**4.11** Таблица subsections. В ней присутствуют идентификаторы из таблиц sections и ads, при помощи которых строятся отношения один ко многим.



Поле name. Содержится имя подраздела объявлений.

Поле section\_id. Минимизированная версия изображения.

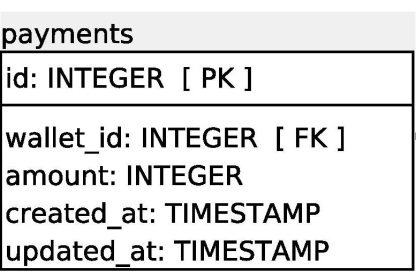
**4.12** Таблица wallets. Связана отношением один ко многим с таблицей users, а также с таблицей payments.



Поле balance. Текущий баланс на кошельке.

Поле user\_id. Уникальный идентификатор пользователя, которому принадлежит кошелек.

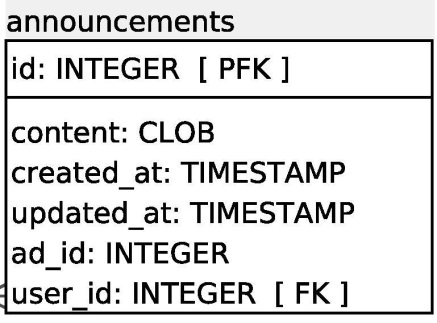
**4.13** Таблица payments.



Поле amount. Представляет собой размер платежа.

Поле wallet\_id. Хранит идентификатор кошелька пользователя.

**4.14** Таблица announcements. Хранит поля с идентификаторами объявления, на которое ссылается оповещение, и пользователя, которому оповещение адресовано.

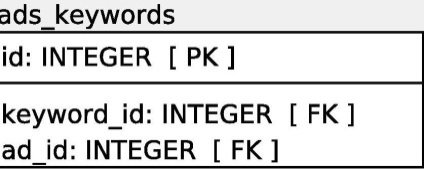


Поле content. Текстовое содержание оповещения.

Поле user\_id. Идентификатор пользователя, которому предназначается оповещение.

Поле ad\_id. Идентификатор объявления, с которым связано оповещение.

**4.15** Таблица ads\_keywords. Таблица, которой не соответствует модель, и, следовательно, которая не нуждается в уникальном идентификаторе. При помощи нее реализуется связь многие ко многим между таблицами ads и keywords.



# 5 ОПИСАНИЕ БИЗНЕС-ЛОГИКИ ПРИЛОЖЕНИЯ

## **5.1 Схема данных приложения «Рекламные объявления»**

Схема данных отражает путь данных при решении задач и отображает этапы обработки, также различные применяемые носители данных. В приложении Б представлена схема данных разработанного программного средства. Как видно из представленной схемы, обращение к различным источникам данных, будь то жесткий диск или база данных, происходит достаточно интенсивно.

Далее будет представлено описание каждого из блоков, отображающих обращение к устройствам для работы с данными.

1. Блок 2. Обращение к жесткому диску с цель получения статических изображений, отображаемых на главной странице.
2. Блок 11. Аналогичен блоку 2 – получение статических изображений, отображаемых на странице.
3. Блок 28. Осуществление запросов к базе данных на основании сформированных поисковых запросов.
4. Блок 31. Обращение к базе данных с целью проверки подлинности и наличия учетных данных пользователя. Если проверка завершается успешно, то в базе обновляются данные о времени последней авторизации.
5. Блок 32. Сохранение учетных данных пользователя в базе данных.
6. Блок 34. Получение из базы полной информации о выбранном объявлении.
7. Блок 47. Выборка из базы данных созданных пользователем объявлений для отображения на странице.
8. Блок 52. Сохранение созданного комментария в базе данных.
9. Блок 65. Обращение к базе данных для выбранного объявления.
10. Блок 67. Удаление объявления из базы данных.
11. Блок 76. Сохранение новых данных редактируемого объявления в базе данных.

# 6 ТЕСТИРОВАНИЕ ПРИЛОЖЕНИЯ

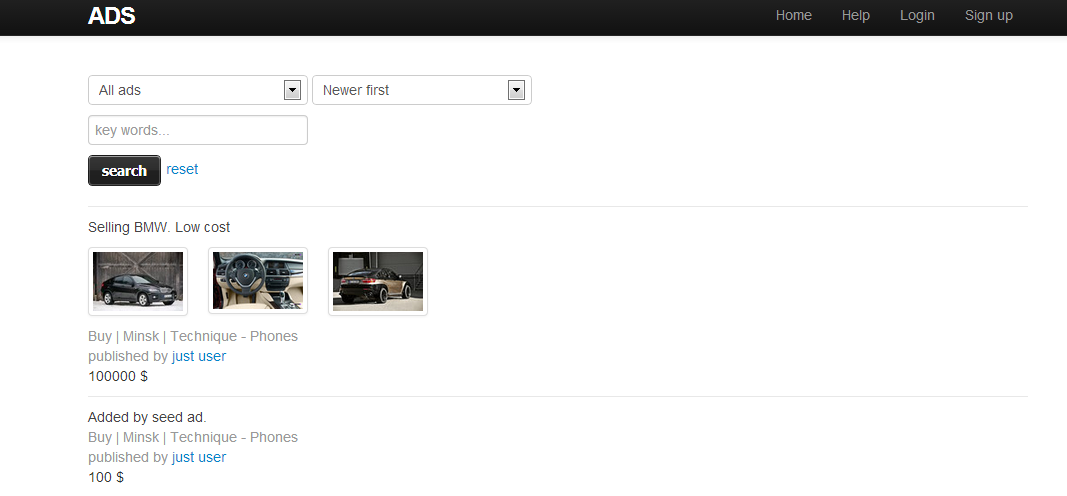
Важнейшим требованием, предъявляемым к системе, является ее надежность. Его выполнение не может быть осуществлено без наличия различного рода тестов, проверяющих все стороны функционирования приложения, начиная от сохранения информации в базе данных и заканчивая корректным отображением страниц в браузере. В данном разделе будет рассмотрено тестирование сохранений в базе данных и корректное отображение верстки страниц в браузере.

## **6.1 Функциональное тестирование**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Функция программного средства | Входные данные | Ожидаемый результат | Фактический результат |
| 1 | Создание пользователя | Email: [123qwe@gmail.com](mailto:123qwe@gmail.com)  First Name: Bil  Last Name: Martin  Password: mmmmrrrr  Confirm Password: mmmmrrrr | Success | Success |
| Email: [123qwe@gmail.com](mailto:123qwe@gmail.com)  First Name: Bil  Last Name: Martin  Password: 123  Confirm Password: 123 | Failure. Password too short. | Failure. Password too short |
| Email: [123qwe@gmail.com](mailto:123qwe@gmail.com)  First Name: Bil  Last Name: Martin  Password: pppppp  Confirm Password: pppppd | Failure. Passwords are not equal | Failure. Passwords are not equal |
| 2 | Вход на сайт | Email: [123qwe@gmail.com](mailto:123qwe@gmail.com)  Password: pppppp32 | Success | Success |
| Email: [dickinson@gmail.com](mailto:dickinson@gmail.com)  Password: pppppp12 | Success | Success |
| Email: [pesch@gmail.com](mailto:pesch@gmail.com)  Password: passwordd | Success | Success |
| 3 | Создание объявления | Content: game of thrones | Success. Отображается на странице пользователя | Success. Отображается на странице пользователя |
| Content: kingdom  Keywords: word | Success. Отображается на странице пользователя | Success. Отображается на странице пользователя |
| Content: Still weather  Place: Grodno  Subsection: laptop | Success. Отображается на странице пользователя | Success. Отображается на странице пользователя |
| 4 | Редактирование объявления | Old:  Content: game of thrones  New:  Content: Feeling sad | Success.  Изменения успешно произведены. | Success.  Изменения успешно произведены |
| Old:  Content: game of thrones  Keywords: book  New:  Content: game of thrones  Description: magazine | Success.  Изменения успешно произведены. | Success.  Изменения успешно произведены |
| Old:  Content: Thunder  Keywords: foo  New:  Content: Thunder  Keywords: bar | Success.  Изменения успешно произведены. | Success.  Изменения успешно произведены |
| 5 | Загрузка фотографий на сервер | Файл: my\_photo.jpg | Success | Success |
| Файл: graphic.bmp | Success | Success |
| Файл: schema.jpeg | Success | Success |
| 6 | Редактирование типов объявлений | Old:  Name: one\_type  New:  Name: another\_type | Success.  Изменения успешно произведены. | Success.  Изменения успешно произведены. |
|  |  | Old:  Name: name  New:  Name: second\_name | Success.  Изменения успешно произведены. | Success.  Изменения успешно произведены. |
| 7 | Комментирование | Content: content\_1 | Chicken Burger  content\_1 | Chicken Burger content\_1 |
| Content: content\_2 | Mac Duck  content\_2 | Mac Duck  content\_2 |
| Content: one another content | Garry Potter  one another content | Garry Potter  one another content |
| 8 | Поиск по сайту | Key: cool | 01:  BMW  02:  Added by seed ad. | 01:  BMW  02:  Added by seed ad. |
| Key: bmw | 01: BMW | 01: BMW |
| Key: minsk | 01: BMW place: minsk | 01: BMW place: minsk |

Функциональное тестирование было проведено в автоматическом режиме, использую такие предметно-ориентированные языки как RSpec, Capybara, а также технологии Selenium, DatabaseCleaner.

# 7 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

При входе на сайт, пользователь попадает на главную страницу, изображенную на рисунке 7.1.

Доступ к функциональной составляющей ресурса, хотя и не полной, может быть осуществлен даже без регистрации, просто зайдя на сайт. После этого пользователь может просматривать объявления, искать их по ключевым словам.

Рисунок 7.1 Главная страница приложения

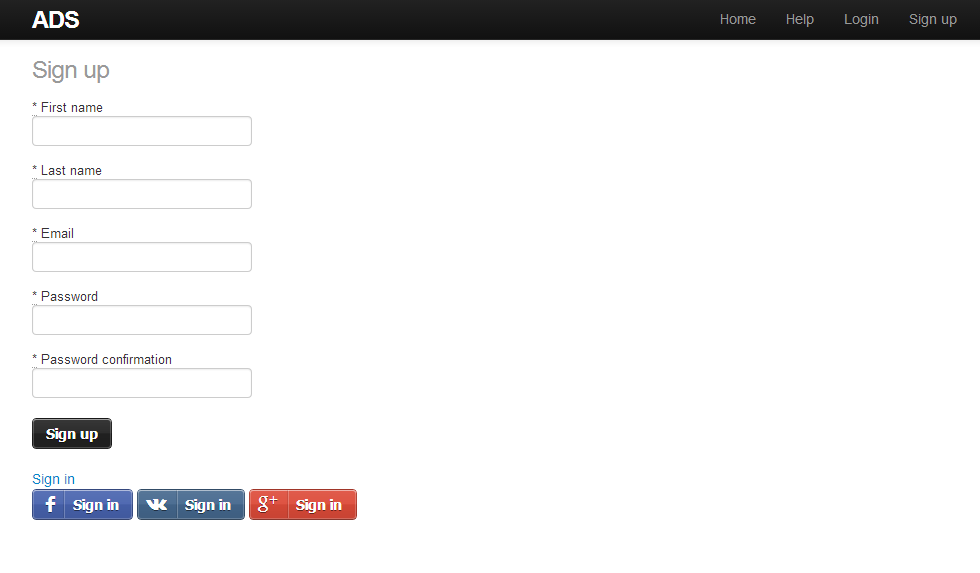
Для доступа к полному функционалу сайта пользователю необходимо зарегистрироваться. Итак, окно регистрации представлено на рисунке 7.2. 

Рисунок 7.2 Страница регистрации пользователя

Как видно из рисунка 7.2, для регистрации необходимо указать имя почтового ящика (должно быть уникальным), имени, фамилии, пароля и его подтверждения. При успешном заполнении указанных полей пользователь будет зарегистрирован на сайте. Также присутствуют кнопки для аутентификации через социальные сети.

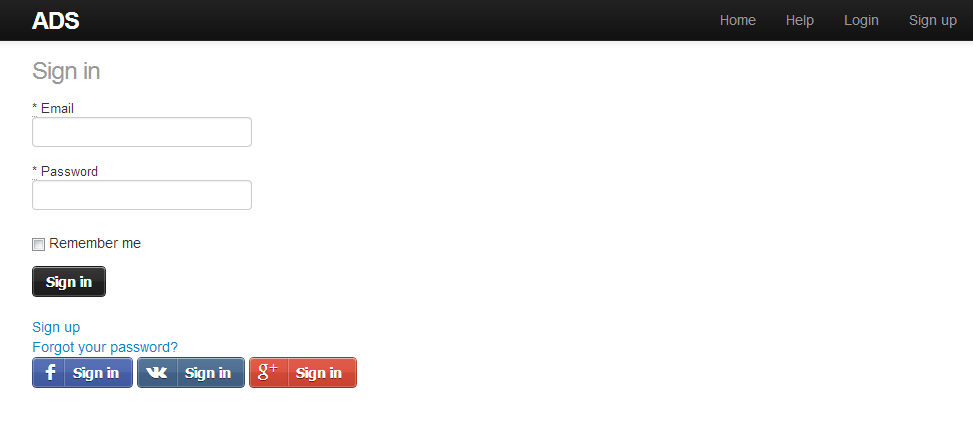


Рисунок 7.3 Страница ввода учетных данных

Если у пользователя уже создан аккаунт, то он может нажать на кнопку Login и ввести свое имя почтового ящика и пароль. В результате успешного выполнения данной операции будет получен доступ к полной функциональности ресурса. Страница входа показа на рисунке 7.3.

После успешного прохождения процедуры авторизации и аутентификации пользователь может зайти на страницу его аккаунта, представленную на рисунке 7.4. На данной странице он может осуществить создание новых объявлений и редактирование уже существующих, а также может отредактировать свой профиль, просмотреть созданные им объявления.

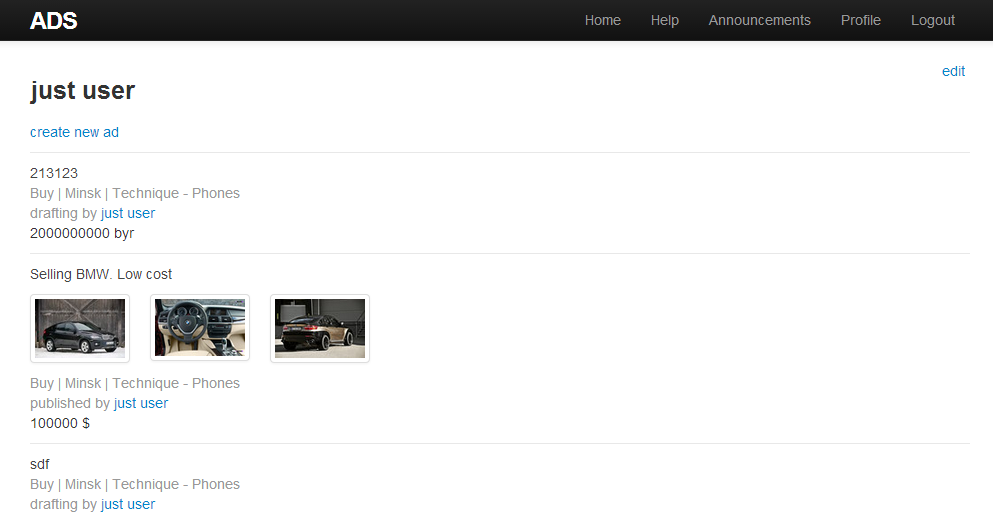


Рисунок 7.4 Персональная страница пользователя

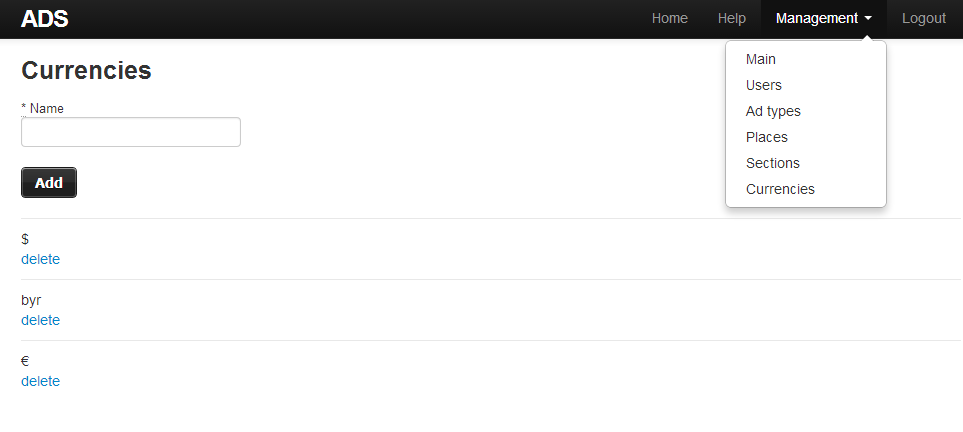


Рисунок 7.5 Страница результата поисковой выдачи

Помимо всего вышеперечисленного, на сайте существует страница администратора, представленная на рисунке 7.5. На ней может осуществляться просмотр информации о пользователях, а также просмотр, добавление и удаление различных атрибутов объявлений.

В данном руководстве были рассмотрены основные функциональные возможности, которые предоставляет разработанное программное средство.

# ЗАКЛЮЧЕНИЕ

В результате выполнения данного курсового проекта было разработано безопасное, надежное программное средство с эргономичным пользовательским интерфейсом. Разработанное программное средство позволяет пользователям создавать рекламные объявления, публиковать их, загружать фотографии на сервер, осуществлять поиск по сайту. Помимо этого, программный продукт предоставляет возможность зарегистрированным пользователям редактировать свои объявления, а также оставлять комментарии к объявлениям других посетителей ресурса. Пользователям, находящимся в роли администратора, доступны дополнительные возможности по удалению комментариев, объявлений, пользователей, атрибутов объявлений, а также по изменению их состояний (отправка на публикацию и прочее).

Разработанное ПС удовлетворяет требованиям надежности за счет использования при аутентификации и авторизации проверенных многолетним опытом сторонних библиотек, которые позволяют защитить данные пользователей от утечек, а сайт – от хакерских атак.

Важной особенностью программного продукта является его проверенная безотказная работа, что было достигнуто путем написания и выполнения автоматических тестов, которые на каждом этапе разработки проверяют корректность работы программы. Кроме того, требование надежности невозможно было бы выполнить без использования тщательно спроектированной СУБД, отвечающей за хранение пользовательских данных.

Архитектурная составляющая проекта была спроектирована максимально гибко. В результате этого она предоставляет широкие возможности по расширению функциональной составляющей веб-сайта.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Wikipedia [Электронный ресурс] – Электронные данные – Режим доступа: <http://ru.wikipedia.org/wiki/BBS>.
2. Wikipedia [Электронный ресурс] – Электронные данные – Режим доступа: <http://ru.wikipedia.org/wiki/Реклама>.
3. At-www [Электронный ресурс] – Электронные данные – Режим доступа: <http://www.at-www.ru/rol-html-v-sozdanii-web-stranic.html>
4. Matsumoto, M. Mersenne twister: A 623-dimensionally equidistributed uniform pseudorandom number generator / M. Matsumoto, T. Nishimura – ACM Trans. on Modeling and Computer Simulations 8 (1): 3-30.
5. Nixp [Электронный ресурс] – Электронные данные – Режим доступа: <http://www.nixp.ru/articles/Основы-работы-с-OpenSSL.html>
6. Wikipedia [Электронный ресурс] – Электронные данные – Режим доступа: <http://ru.wikipedia.org/wiki/JavaScript>.
7. Wikipedia [Электронный ресурс] – Электронные данные – Режим доступа: <http://ru.wikipedia.org/wiki/Model-View-Controller>
8. Mysql.ru [Электронный ресурс] – Электронные данные – Режим доступа: <http://www.mysql.ru/docs/man/MySQL-PostgreSQL_features.html>
9. Ruby-lang [Электронный ресурс] – Электронные данные – Режим доступа: <https://www.ruby-lang.org/ru/about/>
10. Aspnet [Электронный ресурс] – Электронные данные – Режим доступа: [http://www.aspnet.com.ua/BlogAll/sravnenie-Asp-Net-mvc.aspx/203#.UrA4aNJdVKY](http://www.aspnet.com.ua/BlogAll/sravnenie-Asp-Net-mvc.aspx/203#.UrA4aNJdVKY )
11. Wikipedia [Электронный ресурс] – Электронные данные – Режим доступа: <http://ru.wikipedia.org/wiki/HTTP>

# ПРИЛОЖЕНИЕ А

**(обязательное)**

## **Программный код создания базы данных приложения**

SET statement\_timeout = 0;

SET client\_encoding = 'UTF8';

SET standard\_conforming\_strings = on;

SET check\_function\_bodies = false;

SET client\_min\_messages = warning;

CREATE EXTENSION IF NOT EXISTS plpgsql WITH SCHEMA pg\_catalog;

COMMENT ON EXTENSION plpgsql IS 'PL/pgSQL procedural language';

SET search\_path = public, pg\_catalog;

SET default\_tablespace = '';

SET default\_with\_oids = false;

CREATE TABLE ad\_types (

id integer NOT NULL,

name character varying(255),

created\_at timestamp without time zone NOT NULL,

updated\_at timestamp without time zone NOT NULL

);

CREATE SEQUENCE ad\_types\_id\_seq

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

ALTER SEQUENCE ad\_types\_id\_seq OWNED BY ad\_types.id;

CREATE TABLE ads (

id integer NOT NULL,

content text,

user\_id integer,

created\_at timestamp without time zone NOT NULL,

updated\_at timestamp without time zone NOT NULL,

state character varying(255),

ad\_type\_id integer,

published\_at timestamp without time zone,

place\_id integer,

subsection\_id integer,

price integer,

currency\_id integer

);

CREATE SEQUENCE ads\_id\_seq

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

ALTER SEQUENCE ads\_id\_seq OWNED BY ads.id;

CREATE TABLE ads\_keywords (

ad\_id integer,

keyword\_id integer

);

CREATE TABLE announcements (

id integer NOT NULL,

content text,

user\_id integer,

created\_at timestamp without time zone NOT NULL,

updated\_at timestamp without time zone NOT NULL,

ad\_id integer

);

CREATE SEQUENCE announcements\_id\_seq

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

ALTER SEQUENCE announcements\_id\_seq OWNED BY announcements.id;

CREATE TABLE comments (

id integer NOT NULL,

content character varying(255),

created\_at timestamp without time zone NOT NULL,

updated\_at timestamp without time zone NOT NULL,

ad\_id integer,

user\_id integer

);

CREATE SEQUENCE comments\_id\_seq

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

ALTER SEQUENCE comments\_id\_seq OWNED BY comments.id;

CREATE TABLE currencies (

id integer NOT NULL,

name character varying(255),

created\_at timestamp without time zone NOT NULL,

updated\_at timestamp without time zone NOT NULL

);

CREATE SEQUENCE currencies\_id\_seq

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

ALTER SEQUENCE currencies\_id\_seq OWNED BY currencies.id;

CREATE TABLE keywords (

id integer NOT NULL,

name character varying(255),

created\_at timestamp without time zone NOT NULL,

updated\_at timestamp without time zone NOT NULL

);

CREATE SEQUENCE keywords\_id\_seq

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

ALTER SEQUENCE keywords\_id\_seq OWNED BY keywords.id;

CREATE TABLE payments (

id integer NOT NULL,

wallet\_id integer,

amount integer,

created\_at timestamp without time zone NOT NULL,

updated\_at timestamp without time zone NOT NULL

);

CREATE SEQUENCE payments\_id\_seq

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

ALTER SEQUENCE payments\_id\_seq OWNED BY payments.id;

CREATE TABLE photos (

id integer NOT NULL,

ad\_id integer,

created\_at timestamp without time zone NOT NULL,

updated\_at timestamp without time zone NOT NULL,

name character varying(255),

file\_file\_name character varying(255),

file\_content\_type character varying(255),

file\_file\_size integer,

file\_updated\_at timestamp without time zone

);

CREATE SEQUENCE photos\_id\_seq

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

ALTER SEQUENCE photos\_id\_seq OWNED BY photos.id;

CREATE TABLE places (

id integer NOT NULL,

name character varying(255),

created\_at timestamp without time zone NOT NULL,

updated\_at timestamp without time zone NOT NULL

);

CREATE SEQUENCE places\_id\_seq

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

ALTER SEQUENCE places\_id\_seq OWNED BY places.id;

CREATE TABLE schema\_migrations (

version character varying(255) NOT NULL

);

CREATE TABLE sections (

id integer NOT NULL,

name character varying(255),

created\_at timestamp without time zone NOT NULL,

updated\_at timestamp without time zone NOT NULL

);

CREATE SEQUENCE sections\_id\_seq

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

ALTER SEQUENCE sections\_id\_seq OWNED BY sections.id;

CREATE TABLE subsections (

id integer NOT NULL,

name character varying(255),

section\_id integer,

created\_at timestamp without time zone NOT NULL,

updated\_at timestamp without time zone NOT NULL

);

CREATE SEQUENCE subsections\_id\_seq

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

ALTER SEQUENCE subsections\_id\_seq OWNED BY subsections.id;

CREATE TABLE users (

id integer NOT NULL,

email character varying(255) DEFAULT ''::character varying NOT NULL,

encrypted\_password character varying(255) DEFAULT ''::character varying NOT NULL,

reset\_password\_token character varying(255),

reset\_password\_sent\_at timestamp without time zone,

remember\_created\_at timestamp without time zone,

sign\_in\_count integer DEFAULT 0 NOT NULL,

current\_sign\_in\_at timestamp without time zone,

last\_sign\_in\_at timestamp without time zone,

current\_sign\_in\_ip character varying(255),

last\_sign\_in\_ip character varying(255),

created\_at timestamp without time zone NOT NULL,

updated\_at timestamp without time zone NOT NULL,

role character varying(255),

provider character varying(255),

url character varying(255),

first\_name character varying(255),

last\_name character varying(255)

);

CREATE SEQUENCE users\_id\_seq

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

ALTER SEQUENCE users\_id\_seq OWNED BY users.id;

CREATE TABLE wallets (

id integer NOT NULL,

balance integer,

user\_id integer,

created\_at timestamp without time zone NOT NULL,

updated\_at timestamp without time zone NOT NULL

);

CREATE SEQUENCE wallets\_id\_seq

START WITH 1

INCREMENT BY 1

NO MINVALUE

NO MAXVALUE

CACHE 1;

ALTER SEQUENCE wallets\_id\_seq OWNED BY wallets.id;

ALTER TABLE ONLY ad\_types ALTER COLUMN id SET DEFAULT nextval('ad\_types\_id\_seq'::regclass);

ALTER TABLE ONLY ads ALTER COLUMN id SET DEFAULT nextval('ads\_id\_seq'::regclass);

ALTER TABLE ONLY announcements ALTER COLUMN id SET DEFAULT nextval('announcements\_id\_seq'::regclass);

ALTER TABLE ONLY comments ALTER COLUMN id SET DEFAULT nextval('comments\_id\_seq'::regclass);

ALTER TABLE ONLY currencies ALTER COLUMN id SET DEFAULT nextval('currencies\_id\_seq'::regclass);

ALTER TABLE ONLY keywords ALTER COLUMN id SET DEFAULT nextval('keywords\_id\_seq'::regclass);

ALTER TABLE ONLY payments ALTER COLUMN id SET DEFAULT nextval('payments\_id\_seq'::regclass);

ALTER TABLE ONLY photos ALTER COLUMN id SET DEFAULT nextval('photos\_id\_seq'::regclass);

ALTER TABLE ONLY places ALTER COLUMN id SET DEFAULT nextval('places\_id\_seq'::regclass);

ALTER TABLE ONLY sections ALTER COLUMN id SET DEFAULT nextval('sections\_id\_seq'::regclass);

ALTER TABLE ONLY subsections ALTER COLUMN id SET DEFAULT nextval('subsections\_id\_seq'::regclass);

ALTER TABLE ONLY users ALTER COLUMN id SET DEFAULT nextval('users\_id\_seq'::regclass);

ALTER TABLE ONLY wallets ALTER COLUMN id SET DEFAULT nextval('wallets\_id\_seq'::regclass);

ALTER TABLE ONLY ad\_types

ADD CONSTRAINT ad\_types\_pkey PRIMARY KEY (id);

ALTER TABLE ONLY ads

ADD CONSTRAINT ads\_pkey PRIMARY KEY (id);

ALTER TABLE ONLY announcements

ADD CONSTRAINT announcements\_pkey PRIMARY KEY (id);

ALTER TABLE ONLY comments

ADD CONSTRAINT comments\_pkey PRIMARY KEY (id);

ALTER TABLE ONLY currencies

ADD CONSTRAINT currencies\_pkey PRIMARY KEY (id);

ALTER TABLE ONLY keywords

ADD CONSTRAINT keywords\_pkey PRIMARY KEY (id);

ALTER TABLE ONLY payments

ADD CONSTRAINT payments\_pkey PRIMARY KEY (id);

ALTER TABLE ONLY photos

ADD CONSTRAINT photos\_pkey PRIMARY KEY (id);

ALTER TABLE ONLY places

ADD CONSTRAINT places\_pkey PRIMARY KEY (id);

ALTER TABLE ONLY sections

ADD CONSTRAINT sections\_pkey PRIMARY KEY (id);

ALTER TABLE ONLY subsections

ADD CONSTRAINT subsections\_pkey PRIMARY KEY (id);

ALTER TABLE ONLY users

ADD CONSTRAINT users\_pkey PRIMARY KEY (id);

ALTER TABLE ONLY wallets

ADD CONSTRAINT wallets\_pkey PRIMARY KEY (id);

CREATE INDEX index\_ads\_on\_user\_id\_and\_created\_at ON ads USING btree (user\_id, created\_at);

CREATE UNIQUE INDEX index\_users\_on\_email ON users USING btree (email);

CREATE UNIQUE INDEX index\_users\_on\_reset\_password\_token ON users USING btree (reset\_password\_token);

CREATE UNIQUE INDEX unique\_schema\_migrations ON schema\_migrations USING btree (version);

INSERT INTO schema\_migrations (version) VALUES ('20131001125838');

INSERT INTO schema\_migrations (version) VALUES ('20131002163513');

INSERT INTO schema\_migrations (version) VALUES ('20131003201908');

INSERT INTO schema\_migrations (version) VALUES ('20131010110159');

INSERT INTO schema\_migrations (version) VALUES ('20131014163800');

INSERT INTO schema\_migrations (version) VALUES ('20131015102451');

INSERT INTO schema\_migrations (version) VALUES ('20131020114528');

INSERT INTO schema\_migrations (version) VALUES ('20131022155909');

INSERT INTO schema\_migrations (version) VALUES ('20131022163545');

INSERT INTO schema\_migrations (version) VALUES ('20131030115017');

INSERT INTO schema\_migrations (version) VALUES ('20131105094310');

INSERT INTO schema\_migrations (version) VALUES ('20131105114048');

INSERT INTO schema\_migrations (version) VALUES ('20131106112218');

INSERT INTO schema\_migrations (version) VALUES ('20131118162045');

INSERT INTO schema\_migrations (version) VALUES ('20131118162801');

INSERT INTO schema\_migrations (version) VALUES ('20131120184535');

INSERT INTO schema\_migrations (version) VALUES ('20131121152832');

INSERT INTO schema\_migrations (version) VALUES ('20131121153234');

INSERT INTO schema\_migrations (version) VALUES ('20131121183124');

INSERT INTO schema\_migrations (version) VALUES ('20131121192243');

INSERT INTO schema\_migrations (version) VALUES ('20131121192452');

INSERT INTO schema\_migrations (version) VALUES ('20131125162625');

INSERT INTO schema\_migrations (version) VALUES ('20131125164527');

INSERT INTO schema\_migrations (version) VALUES ('20131126174218');

INSERT INTO schema\_migrations (version) VALUES ('20131127083907');

INSERT INTO schema\_migrations (version) VALUES ('20131127110327');

INSERT INTO schema\_migrations (version) VALUES ('20131127184022');

INSERT INTO schema\_migrations (version) VALUES ('20131205093752');

INSERT INTO schema\_migrations (version) VALUES ('20131205100023');

INSERT INTO schema\_migrations (version) VALUES ('20131210143538');

INSERT INTO schema\_migrations (version) VALUES ('20131210150148');

INSERT INTO schema\_migrations (version) VALUES ('20131210153823');

## **Исходный код модели объявлений**

**class** **Ad** **<** ActiveRecord**::**Base

  attr\_accessible :content, :photos\_attributes, :ad\_type\_id, :place\_id, :subsection\_id, :currency\_id, :price

  belongs\_to :user

  belongs\_to :ad\_type

  belongs\_to :place

  belongs\_to :subsection

  belongs\_to :currency

  has\_many :photos, :dependent **=>** :destroy

  has\_many :comments, :dependent **=>** :destroy

  has\_many :announcements, :dependent **=>** :destroy

  has\_and\_belongs\_to\_many :keywords

  validates :user\_id, :ad\_type\_id, :place\_id, :subsection\_id,

    :currency\_id, presence: **true**

  validates :content, presence: **true**, length: {maximum: 200}, format: { with: GlobalConstants**::**Content\_regexp }

  accepts\_nested\_attributes\_for :photos, :allow\_destroy **=>** **true**

  default\_scope order: 'ads.created\_at DESC'

  scope :published, **->** { where(state: 'published') }

  scope :posting, **->** { where(state: 'posting') }

  @@Per\_page\_small **=** 5

  @@Per\_page\_big **=** 12

  @@max\_price **=** 2 **\*\*** (1**.**size **\*** 8 **-** 1)

**def** **self.max\_price**

    @@max\_price

**end**

**def** **self.per\_page\_small**

    @@Per\_page\_small

**end**

**def** **self.per\_page\_big**

    @@Per\_page\_big

**end**

  validates\_numericality\_of :price, greater\_than\_or\_equal\_to: 0,

    less\_than: max\_price, :allow\_nil **=>** **true**,

    :message **=>** "can only be ranged number"

  state\_machine :state, :initial **=>** :drafting **do**

    after\_transition :approved **=>** :published **do** **|**ad, transition**|**

      ad**.**published\_at **=** Time**.**now

      ad**.**save

**end**

    after\_transition :published **=>** :archived **do** **|**ad, transition**|**

      ad**.**published\_at **=** **nil**

      ad**.**save

      new\_announcement(ad, transition)

**end**

    after\_transition :posting **=>** any **do** **|**ad, transition**|**

      new\_announcement(ad, transition)

**end**

    after\_transition any **=>** **[**:published, :archived**]** **do** **|**ad, transition**|**

      new\_announcement(ad, transition)

**end**

    event :post **do**

      transition :drafting **=>** :posting

**end**

    event :reject **do**

      transition :posting **=>** :rejected

**end**

    event :approve **do**

      transition :posting **=>** :approved

**end**

    event :publish **do**

      transition :approved **=>** :published

**end**

    event :archive **do**

      transition :published **=>** :archived

**end**

    event :draft **do**

      transition **[**:rejected, :archived**]** **=>** :drafting

**end**

**def** **new\_announcement**(ad, transition)

      an **=** ad**.**user**.**announcements**.**build(ad\_id: ad**.**id,

        content: I18n**.**translate(:was, scope: **[**:ads**]**) **+** transition**.**to);

      an**.**save

**end**

**end**

**end**

**class** **Section** **<** ActiveRecord**::**Base

**include** Option

  attr\_accessible :name

  has\_many :subsections

  validates :name, presence: **true**,

    format: { with: GlobalConstants**::**Content\_regexp}, length: {maximum: 20}

  before\_destroy :before\_destroy

**def** **before\_destroy**

**if** **!**self**.**subsections**.**blank?

      errors**.**add(:subsections, 'must be blank')

**false**

**end**

**end**

**end**

## **Исходный код модели пользователей**

**class** **User** **<** ActiveRecord**::**Base

**extend** Enumerize

  has\_many :ads, dependent: :destroy

  has\_many :comments, dependent: :destroy

  has\_many :announcements, dependent: :destroy

  devise :database\_authenticatable, :registerable,

         :recoverable, :rememberable, :trackable, :validatable, :omniauthable

  enumerize :role, **in**: **[**:guest, :user, :admin**]**, default: :guest

  attr\_accessible :first\_name, :last\_name, :email, :password,

    :password\_confirmation, :remember\_me, :provider, :url

  attr\_accessible :first\_name, :last\_name, :email, :password,

    :password\_confirmation, :remember\_me, :role, :as **=>** :admin

  validates :first\_name, :last\_name, :email, presence: **true**

  default\_scope order: 'users.created\_at DESC'

  after\_create :user\_role

**def** **name**

    first\_name **+** ' ' **+** last\_name

**end**

**def** **profile\_ads**(current\_user)

**if** self**.**role**.**admin?

      Ad**.**posting

**elsif** self **==** current\_user

      self**.**ads

**else**

      self**.**ads**.**published

**end**

**end**

**def** **self.find\_for\_oauth** access\_token

    info **=** access\_token**.**info

**if** access\_token**.**provider **==** 'vkontakte'

      email **=** access\_token**.**extra**.**raw\_info**.**screen\_name **+** '@vk.com'

**else**

      email **=** info**.**email

**end**

**if** user **=** User**.**where(email: email)**.**first

      user

**else**

      friendly\_token **=** Devise**.**friendly\_token**[**0,20**]**

      User**.**create!(first\_name: info**.**first\_name, last\_name: info**.**last\_name,

        email: email, :provider **=>** access\_token**.**provider,

        :url **=>** access\_token**.**info**.**urls**[**access\_token**.**provider**.**capitalize**]**,

        password: friendly\_token, password\_confirmation: friendly\_token)

**end**

**end**

**private**

**def** **user\_role**

      self**.**role **=** 'user' **if** self**.**role **==** 'guest'

      self**.**save

**end**

**end**

## **Исходный код некоторых контроллеров**

**class** **AdsController** **<** ApplicationController

  load\_and\_authorize\_resource

**def** **create**

    @ad**.**keywords **=** get\_keywords\_from\_params

    current\_user**.**ads **<<** @ad

    @ad**.**save

    load\_dropdown\_collections

    load\_keywords\_and\_hints

    respond\_with(@ad, :location **=>** @ad)

**end**

**def** **destroy**

    @ad**.**destroy

    respond\_with @ad

**end**

**def** **index**

    ad\_type\_id **=** params**[**'ad\_type\_id'**].**blank? ? **nil** : params**[**'ad\_type\_id'**]**

    order **=** params**[**'created\_at'**]** **||** 'created\_at DESC'

    @ads **=** Ad**.**search(params**[**'key\_word'**]**,

      :sql **=>** {:include **=>** **[**:user, :ad\_type, :photos**]** },

      :conditions **=>** {:ad\_type\_id **=>** ad\_type\_id, :state **=>** :published},

      :order **=>** order, :page **=>** params**[**:page**]**, :per\_page **=>** Ad**.**per\_page\_small)

    @ad\_types **=** AdType**.**all

**end**

**def** **update**

    @ad**.**keywords **=** get\_keywords\_from\_params

    @ad**.**attributes **=** params**[**:ad**]**

    @ad**.**save

    load\_dropdown\_collections

    load\_keywords\_and\_hints

    respond\_with(@ad, :location **=>** @ad)

**end**

**def** **show**

    load\_keywords

    session**[**:return\_to**]** **||=** request**.**referer

    @ad\_types **=** AdType**.**all

**if** can? :create, Comment

      @comment **=** current\_user**.**comments**.**build(params**[**:comment**]**)

      @comment**.**ad\_id **=** params**[**:id**]**

**end**

**end**

**def** **edit**

    load\_dropdown\_collections

    load\_keywords\_and\_hints

**end**

**def** **new**

    load\_dropdown\_collections

    load\_keywords\_hint

**end**

**def** **change\_state**

    @ad**.**public\_send(params**[**:state\_event**]**)

    @ad**.**save

    flash**[**:notice**]** **=** t(:status\_changed, scope: **[**:ads**]**)

    redirect\_to(:back)

**end**

**private**

**def** **load\_keywords\_and\_hints**

    load\_keywords

    load\_keywords\_hint

**end**

**def** **load\_keywords**

    gon**.**watch**.**keywords **=** @ad**.**keywords

**end**

**def** **load\_keywords\_hint**

    gon**.**keywords\_hint **=** Keyword**.**all

**end**

**def** **get\_keywords\_from\_params**

**if** params**[**'tags'**]**

      params**[**'tags'**].**split(',')**.**

        map { **|**keyword**|** Keyword**.**find\_or\_create\_by\_name(keyword) }

**else**

**return** **[]**

**end**

**end**

**def** **load\_dropdown\_collections**

    @ad\_types **=** AdType**.**all

    @places **=** Place**.**all

    @sections **=** Section**.**all

    @currencies **=** Currency**.**all

**end**

**end**

**class** **UsersController** **<** ApplicationController

  load\_and\_authorize\_resource

**def** **index**

    @users **=** User**.**paginate(:page **=>** params**[**:page**]**, :per\_page **=>** Ad**.**per\_page\_big)

**end**

**def** **destroy**

    @user**.**destroy

    respond\_with(@user)

**end**

**def** **show**

    @ads **=** @user**.**profile\_ads(current\_user)**.**

      paginate(:page **=>** params**[**:page**]**, :per\_page **=>** Ad**.**per\_page\_small)

**end**

**def** **make\_admin**

    @user**.**role **=** :admin

    @user**.**save

    respond\_with(@user, :location **=>** @user)

**end**

**end**

**class** **PlacesController** **<** ApplicationController

  load\_and\_authorize\_resource

**def** **create**

    @place**.**save

    @places **=** Place**.**all

    respond\_with(@place, :location **=>** :new\_place)

**end**

**def** **destroy**

    @place**.**destroy

    respond\_with(@place, :location **=>** :new\_place)

**end**

**def** **new**

    @places **=** Place**.**all

**end**

**end**

## **Исходный код javascript на сайте**

$( document ).ajaxComplete(**function**() {

  hermitage.init();

});

$( document ).ready(**function**(){

  $('.selectpicker').selectpicker();

  setTagit();

  setInputValueBeforeSubmit();

});

**function** setTagit () {

**if** (gon.keywords\_hint) {

**var** keywordsHint **=** gon.keywords\_hint.map(**function**(g){ **return** g.name });

  }

**var** settings **=** {

    availableTags**:** keywordsHint,

    autocomplete**:** {delay**:** 0, minLength**:** 1},

    removeConfirmation**:** **true**

  };

  $(".writable#keywords").tagit(settings);

  settings.readOnly **=** **true**;

  $(".readOnly#keywords").tagit(settings);

**if** (gon.keywords) {

**var** keywords **=** gon.keywords.map(**function**(g){ **return** g.name });

  }

**for**(key **in** keywords) {

    $("#keywords").tagit("createTag", keywords[key]);

  }

}

**function** setInputValueBeforeSubmit () {

  $(':submit').click(**function**(){

**var** keywords **=** $("#keywords").tagit("assignedTags").join(',');

    $('.tagit-hidden-field').last().val(keywords);

  })

}