Министерство образования Республики Беларусь

Учреждение образования

БелорусскиЙ государственный университет

информатики и радиоэлектроники

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

|  |
| --- |
| *К защите допустить*: |
| Заведующая кафедрой ПОИТ |
| \_\_\_\_\_\_\_\_\_\_\_\_Н. В. Лапицкая |

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к дипломному проекту

на тему

**Клиент-серверное приложение управления скидками на товары и услуги**

БГУИР ДП 1-40 01 01 03 083 ПЗ

|  |  |  |
| --- | --- | --- |
| Студент |  | А. Р. Рыхлицкий |
| Руководитель |  | Л. А. Глухова |
| Консультанты: |  |  |
| *от кафедры ПОИТ* |  | Л. А. Глухова |
| *по экономической части* |  | К. Р. Литвинович |
| *по охране труда* |  | Е.А. Криштопова |
| Нормоконтролер |  | П. Ю. Бранцевич |
|  |  |  |
| Рецензент |  |  |

Минск 2015

**РЕФЕРАТ**

Пояснительная записка 110 с., 20 рис., 12 табл., 22 источника

КЛИЕНТ-СЕРВЕРНОЕ ПРИЛОЖЕНИЕ УПРАВЛЕНИЯ СКИДКАМИ НА ТОВАРЫ И УСЛУГИ

Предметной областью разработки является сфера торговли на пользовательском уровне. Объект разработки – мобильный клиент для пользователя и веб-приложение для управления контентом

Целью разработки является создание удобного в использовании мобильного приложения с широкими функциональными возможностями, а также создание в высокой степени автоматизированного веб-приложения управления контентом для администраторов.

При разработке проекта использовалась среда разработки Sublime Text 3 с различными расширениями, такими как CodeIntel, PackageControl и т.д. Язык программирования серверной части – Ruby, программная платформа – Ruby on Rails, расширенная при помощи многочисленных плагинов; язык программирования и программная платформа мобильного приложения – Java.

Результатом разработки стали простое в использовании мобильное приложение, а также функциональная и удобная администраторская панель. Разработаны диаграмма компонентов, диаграмма вариантов использования клиент-серверного приложения, а также различные схемы алгоритмов.

Предполагается использование приложения пользователями торговых сетей, а также администрацией данных сетей для привлечения новых клиентов.

Разработанное клиент-серверное приложение является экономически эффективным, он полностью оправдывает средства, вложенные в его разработку. Результаты эргономической экспертизы показали, что при использовании рекомендуемых параметров оборудования и рабочего места работа с данным приложением является комфортной для администратора.

**СОДЕРЖАНИЕ**

[Введение 8](#_Toc420451167)

[1 Анализ предметной области 10](#_Toc420451168)

[1.1 Основные принципы и методы организации клиент-серверного приложения 10](#_Toc420451169)

[1.1.1 Клиент-серверное приложение 10](#_Toc420451170)

[1.1.2 Особенности использования баз данных в клиент-серверных приложениях 11](#_Toc420451171)

[1.1.3 Реляционные базы данных 11](#_Toc420451172)

[1.1.4 Применение СУБД 12](#_Toc420451173)

[1.2 Анализ существующих аналогов 13](#_Toc420451174)

[1.2.1 Gazetki.by 13](#_Toc420451175)

[1.2.2 skidki.by 14](#_Toc420451176)

[1.2.3 Минскскидки 15](#_Toc420451177)

[1.2.4 shoppingplus 18](#_Toc420451178)

[1.3 Обоснование выбора технологий разработки 18](#_Toc420451179)

[1.3.1 Выбор архитектуры 18](#_Toc420451180)

[1.3.2 Выбор СУБД 20](#_Toc420451181)

[1.3.3 Выбор языка программирования и платформы разработки 21](#_Toc420451182)

[1.4 Укрупненная спецификация требований 24](#_Toc420451183)

[2 Моделирование предметной области 26](#_Toc420451184)

[2.1 Разработка функциональной модели предметной области 26](#_Toc420451185)

[2.2 Разработка информационной модели 28](#_Toc420451186)

[2.3 Разработка спецификации функциональных требований 30](#_Toc420451187)

[3 Проектирование клиент-серверного приложения 34](#_Toc420451188)

[3.1 Разработка архитектуры 34](#_Toc420451189)

[3.2 Разработка схемы программы клиент-серверного приложения 37](#_Toc420451190)

[3.3 Разработка схемы алгоритма работы администратора приложения 39](#_Toc420451191)

[3.4 Разработка схемы алгоритма добавления скидок администратором 42](#_Toc420451192)

[4 Тестирование клиент-серверного приложения 44](#_Toc420451193)

[5 Методика использования разработанного клиент-серверного приложения 50](#_Toc420451194)

[5.1 Работа со средой для управления скидками администратором 50](#_Toc420451195)

[5.2 Работа с мобильным приложенем для просмотра скидок 57](#_Toc420451196)

[6 Технико-экономическое обоснование разработки и использования клиент-серверного приложения управления скидками 60](#_Toc420451197)

[6.1 Характеристика программного продукта 60](#_Toc420451198)

[6.2 Экономический эффект у разработчика 60](#_Toc420451199)

[6.2.1 Определение объема и трудоемкости ПО 60](#_Toc420451200)

[6.2.2 Расчет сметы затрат и цены заказного ПО 63](#_Toc420451201)

[6.3 Расчет экономического эффекта от применения программного средства у пользователя 69](#_Toc420451202)

[7 Эргономическая экспертиза клиент-серверного приложения управления скидками 74](#_Toc420451203)

[Заключение 86](#_Toc420451204)

[Список использованных источников 88](#_Toc420451205)

[Приложение А. Текст программы 90](#_Toc420451206)

**ОПРЕДЕЛЕНИЯ И СОКРАЩЕНИЯ**

В пояснительной записке применяются следующие определения и сокращения.

ОС – операционная система.

БД – база данных.

СУБД – система управления базами данных.

HTTP – hypertext transfer protocol (протокол передачи гипертекста).

REST – representational state transfer (передача репрезентативного состояния).

SQL – structured query language (язык структурированных запросов).

HTML – hypertext markup language (язык гипертекстовой разметки).

XML – extensible markup language (расширяемый язык разметки).

JSON – JavaScript object notation (формат объектов JavaScript).

MIME – multipurpose internet mail extensions (многоцелевые расширения интернет-почты).

ACID – atomicity, consistency, isolation, durability (атомарность, согласованность, изолированность, надежность).

GOMS – goals, objects, methods and selection rules (цели, объекты, методы и выделения).

# Введение

Последние десятилетия интенсивно развиваются компьютерные технологии, все глубже проникая во все сферы деятельности человека. Едва ли в современном мире осталась область, где не используются компьютеры, будь то медицина, образование, и даже армия.

Много лет компьютеры используются и в сфере торговли. Сейчас у многих торговых компаний, вне зависимости от размера, есть свои веб-сайты, на которых доступна постоянно актуальная информация по контактам, адресам, персоналу, обслуживанию, товарам и услугам организации.

Посредством повсеместной компьютеризации растет глобализация рынков сбыта, когда покупатели, благодаря всевозможным каталогам на товары и услуги, размещенным во всеобщем доступе в сети интернет, осведомлены о ценах у различных поставщиков намного лучше, чем до этого. В связи с этим растет и конкуренция между торговыми сетями, когда они вынуждены бороться за покупателей, предоставляя различные скидки. Списки скидок можно получить, посетив магазин торговой сети, что не всегда удобно при планировании покупок, или зайдя на сайт сети и пролистав онлайн каталоги. Однако у последнего способа, хоть и вполне удобного, тоже есть свои недостатки. Одним из них является то, что информация из различных сетей разрознена по многочисленным сайтам. В данной ситуации тяжело осуществлять поиск (приходится по несколько раз искать одно и то же на каждом сайте), сравнение товаров (приходится переходить с одного сайта на другой, меняя вкладки браузера). Также зачастую в данных каталогах отсутствует какой бы то ни было поиск, категоризация, фильтрация товаров. Все это привело к тому, что начали появляться специализированные сайты, предоставляющие доступ к каталогам товаров и скидок одновременно с нескольких сайтов торговых организаций. Такие сайты нередко лишены приведенных ниже недостатков, однако обладают собственными.

В последние годы, на фоне значительного роста всех сфер информационной деятельности человека, все большее место в жизни людей занимают мобильные технологии. В большей степени это касается простых пользователей, которым данные технологии позволяют не отрываться от привычных дел на улице, в общественных местах и т.д. В то время как компании, связанные с информационными технологиями, осознали и приняли на использование эти потребности пользователей, многие, в том числе и торговые организации, не поспевают за современными трендами и не удовлетворяют в полной мере потребности пользователей мобильных устройств. В данной ситуации проблему решают сторонние разработчики, предоставляющие программное обеспечение для всех категорий пользователей.

Поэтому тема дипломного проекта, посвященная разработке клиент-серверного приложения для работы со скидками, является актуальной.

# 1 Анализ предметной области

## 1.1 Основные принципы и методы организации клиент-серверного приложения

### **1.1.1** Клиент-серверное приложение

Разрабатываемое программное средство – клиент-серверное. Клиент-сервер – вычислительная или сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг, называемыми серверами, и заказчиками услуг, называемыми клиентами. Физически клиент и сервер – это [программное обеспечение](https://ru.wikipedia.org/wiki/%D0%9F%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). Обычно они взаимодействуют через компьютерную сеть посредством [сетевых протоколов](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%BE%D0%B9_%D0%BF%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB) и находятся на разных вычислительных машинах, но могут выполняться также и на одной машине. Программы-сервера ожидают от клиентских программ запросы и предоставляют им свои ресурсы в виде данных (например, [загрузка файлов](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%B3%D1%80%D1%83%D0%B7%D0%BA%D0%B0_%D1%84%D0%B0%D0%B9%D0%BB%D0%BE%D0%B2) посредством [HTTP](https://ru.wikipedia.org/wiki/HTTP), [FTP](https://ru.wikipedia.org/wiki/FTP) , [потоковое мультимедиа](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%B2%D0%BE%D0%B5_%D0%BC%D1%83%D0%BB%D1%8C%D1%82%D0%B8%D0%BC%D0%B5%D0%B4%D0%B8%D0%B0) или работа с [базами данных](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85)) или сервисных функций (например, работа с [электронной почтой](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%BF%D0%BE%D1%87%D1%82%D0%B0), общение посредством [систем мгновенного обмена сообщениями](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%BC%D0%B3%D0%BD%D0%BE%D0%B2%D0%B5%D0%BD%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%BC%D0%B5%D0%BD%D0%B0_%D1%81%D0%BE%D0%BE%D0%B1%D1%89%D0%B5%D0%BD%D0%B8%D1%8F%D0%BC%D0%B8), просмотр [веб-страниц](https://ru.wikipedia.org/wiki/Web-%D1%81%D1%82%D1%80%D0%B0%D0%BD%D0%B8%D1%86%D0%B0) во [всемирной паутине](https://ru.wikipedia.org/wiki/%D0%92%D1%81%D0%B5%D0%BC%D0%B8%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D0%B0%D1%83%D1%82%D0%B8%D0%BD%D0%B0)) [1].

REST – метод взаимодействия компонентов распределённого приложения в сети [Интернет](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82), при котором вызов удаленной процедуры представляет собой обычный HTTP-запрос (обычно GET или POST; такой запрос называют REST-запрос), а необходимые данные передаются в качестве параметров запроса. Этот способ является альтернативой более сложным методам, таким как [SOAP](https://ru.wikipedia.org/wiki/SOAP), [CORBA](https://ru.wikipedia.org/wiki/CORBA) и [RPC](https://ru.wikipedia.org/wiki/Remote_Procedure_Call) [2].

В качестве необходимых условий для построения распределенных REST-приложений можно перечислить следующие:

1. клиент-серверная архитектура;
2. сервер не обязан сохранять информацию о состоянии клиента;
3. в каждом запросе клиента должно явно содержаться указание о возможности кэширования ответа и получения ответа из существующего [кэша](https://ru.wikipedia.org/wiki/%D0%9A%D1%8D%D1%88);
4. клиент может взаимодействовать не напрямую с сервером, а с произвольным количеством промежуточных узлов. При этом клиент может не знать о существовании промежуточных узлов, за исключением случаев передачи конфиденциальной информации;
5. унифицированный программный интерфейс сервера. Филдинг приводил [URI](https://ru.wikipedia.org/wiki/URI) в качестве примера формата запросов к серверу, а в качестве примера ответа сервера форматы [HTML](https://ru.wikipedia.org/wiki/HTML), [XML](https://ru.wikipedia.org/wiki/XML) и [JSON](https://ru.wikipedia.org/wiki/JSON), различаемые с использованием идентификаторов [MIME](https://ru.wikipedia.org/wiki/MIME).

Если все условия соблюдены, то, приложение получит следующие преимущества:

1. надёжность (за счет отсутствия необходимости сохранять информацию о состоянии клиента, которая может быть утеряна);
2. производительность (за счет использования кэша);
3. масштабируемость;
4. прозрачность системы взаимодействия, особенно необходимая для приложений обслуживания сети;
5. простота интерфейсов;
6. портативность компонентов;
7. легкость внесения изменений;
8. способность эволюционировать, приспосабливаясь к новым требованиям (на примере Всемирной паутины) [2].

### **1.1.2** Особенности использования баз данных в клиент-серверных приложениях

Для клиент-серверных приложений характерно глубокое использование баз данных, хоть и с некоторыми специфичными для сферы веб особенностями:

1. Количество пользователей может не только существенно варьироваться, но и изменяться без предупреждения. В обычных клиент/серверных приложениях администраторы могут контролировать число пользователей и возможность их доступа к данным.
2. Пользователей физически невозможно определить, они могут, не закрыв одного приложения, подключиться к другому, что не гарантирует правильного отключения от приложения.
3. В клиент-серверных приложениях при изменении данных несколькими пользователями используются блокировки на уровне записи (пессимистический параллелизм). В веб-приложениях применяется принцип оптимистического параллелизма, предполагающего фактически монопольный доступ для каждого пользователя.
4. Посредником, реализующим доступ от клиента к БД, выступает специальная программа – веб-сервер, обеспечивающая функционирование пользовательского приложения, выполняющего непосредственный запрос к данным и возврат ответа в виде HTML-документа [2].

### **1.1.3** Реляционные базы данных

При разработке клиент-серверных приложений используются как реляционные, так и нереляционные базы. Так как реляционные базы в большей степени соответствуют специфике и требованиям данного программного средства, были выбраны именно они.

Реляционная модель данных по сравнению с нереляционной имеет следующие достоинства:

1. отображает информацию в наиболее простой для пользователя форме;
2. основана на развитом математическом аппарате, который позволяет достаточно лаконично описать основные операции над данными;
3. позволяет создавать языки манипулирования данными не процедурного типа;
4. манипулирование данными на уровне выходной БД и возможность изменения.

Реляционная база данных представляет собой множество взаимосвязанных таблиц, каждая из которых содержит информацию об объектах определенного вида. Каждая строка таблицы содержит данные об одном объекте (например, автомобиле, компьютере, клиенте), а столбцы таблицы содержат различные характеристики этих объектов – атрибуты (например, номер двигателя, марка процессора, телефоны фирм или клиентов) [3].

Строки таблицы называются записями. Все записи таблицы имеют одинаковую структуру - они состоят из полей (элементов данных), в которых хранятся атрибуты объекта. Каждое поле записи содержит одну характеристику объекта и представляет собой заданный тип данных (например, текстовая строка, число, дата). Для идентификации записей используется первичный ключ. Первичным ключом называется набор полей таблицы, комбинация значений которых однозначно определяет каждую запись в таблице.

### **1.1.4** Применение СУБД

Для работы с данными используются системы управления базами данных (СУБД). Основные функции СУБД:

1. определение данных (описание структуры баз данных);
2. обработка данных;
3. управление данными.

Разработка структуры БД - важнейшая задача, решаемая при проектировании БД. Структура БД (набор, форма и связи ее таблиц) - это одно из основных проектных решений при создании приложений с использованием БД. Созданная разработчиком структура БД описывается на языке определения данных СУБД.

Любая СУБД позволяет выполнять следующие операции с данными:

1. добавление записей в таблицы;
2. удаление записей из таблицы;
3. обновление значений некоторых полей в одной или нескольких записях в таблицах БД;
4. поиск одной или нескольких записей, удовлетворяющих заданному условию.

Для выполнения этих операций применяется механизм запросов. Результатом выполнения запросов является либо отобранное по определенным критериям множество записей, либо изменения в таблицах. Запросы к базе формируются на специально созданном для этого языке, который так и называется «язык структурированных запросов» (SQL - Structured Query Language) [3].

Под управлением данными обычно понимают защиту данных от несанкционированного доступа, поддержку многопользовательского режима работы с данными и обеспечение целостности и согласованности данных.

## 1.2 Анализ существующих аналогов

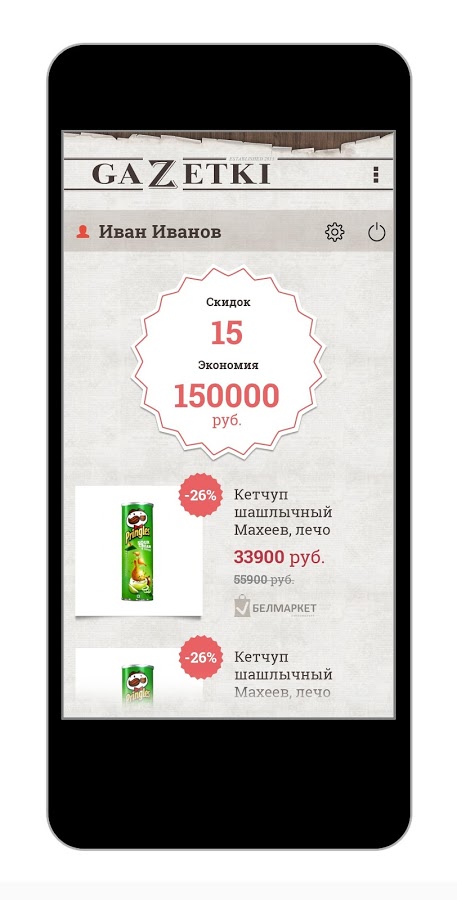
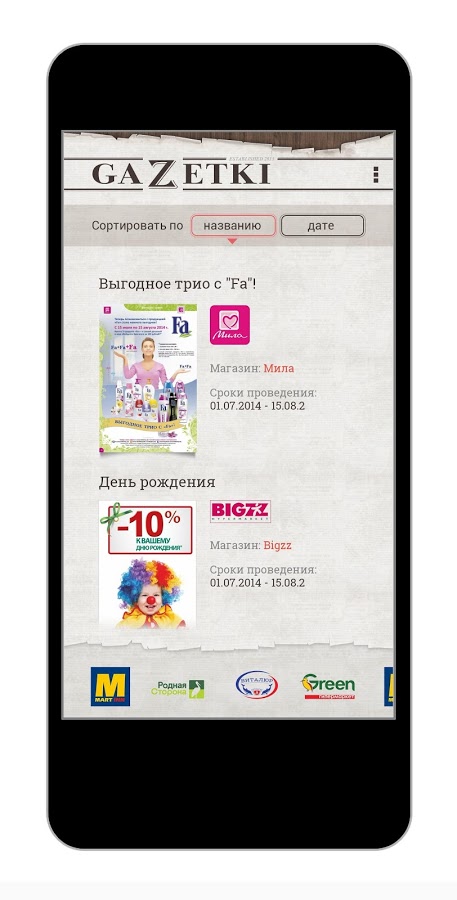
### **1.2.1** Gazetki.by

Gazetki.by – это приложение, посвященное последним (актуальным) скидкам и акциям в коммерческих сетях, работающих на белорусском рынке [4].

На этом сайте можно найти как все скидки, действующие на данный момент в большинстве торговых сетей Беларуси, так и все акции проводимые торговыми сетями. В настоящее время на данном сайте представлено большинство крупных торговых сетей (магазинов) Минска и остальных регионов. На рисунке 1.1 представлен интерфейс приложения.

Достоинства ПС:

1. постоянные и достаточно своевременные обновления списка скидок;
2. концентрация на белорусском рынке;
3. представлено большое количество торговых сетей;
4. простой интерфейс;
5. низкая потребность в ресурсах сети при работе приложения;
6. большая распространенность программного средства;
7. наличие как мобильного, так и веб приложения.
8. Стоит отметить следующие недостатки:
9. нефункциональный интерфейс;
10. высокая потребность в ресурсах сети при старте приложения;
11. отсутствие многих полезных функциональных возможностей.

Рисунок 1.1 – Интерфейс приложения Gazetki.by

### **1.2.2** skidki.by

Skidki.by – приложение, предоставляющее дисконтную систему, которая состоит из покупателей и продавцов. Продавцы – партнеры системы предоставляют скидку на товар или услугу владельцам дисконтной карточки, а покупатели пользуются этими скидками и экономят свои деньги [5].

На сайте размещена подробная информация о партнерах и их товарах и услугах. А также информация о проводящихся специальных акциях и распродажах, как у партнеров системы, так и в прочих организациях. Данное приложение позволяет найти заданный товар по выгодной цене, не затрачивая на это много времени, пользуясь оптимизированным для этих целей интерфейсом. На рисунке 1.2 представлена главная страница приложения.

Достоинства:

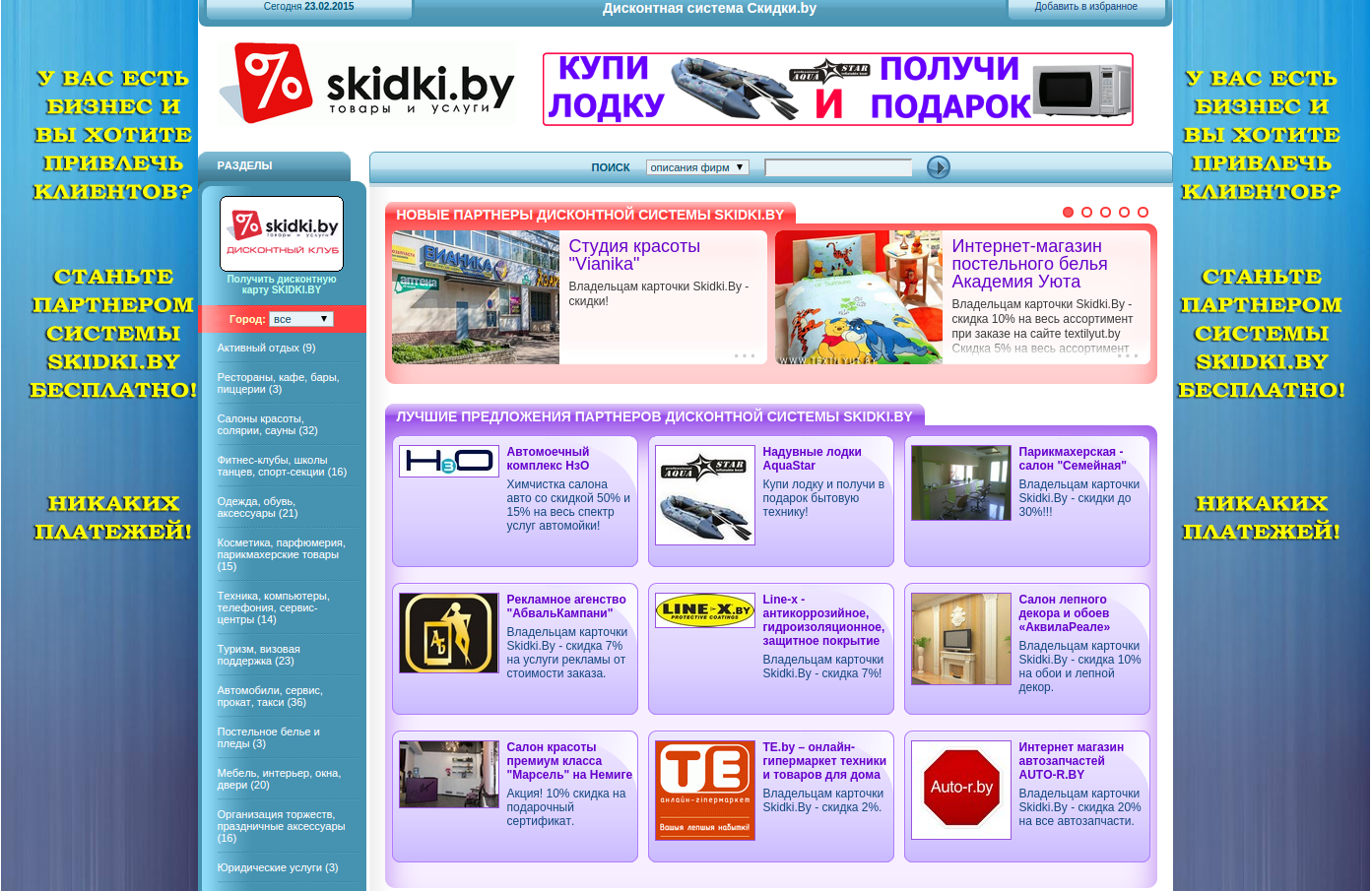
1. предоставление некоторых специальных бонусов, доступных только пользователям приложения;
2. большой спектр категорий товаров на скидках;
3. наличие специальных партнеров приложения;
4. концентрация на белорусском рынке.
5. Недостатки:
6. жесткая ценовая политика;
7. необходимость вложения финансовых средств для пользования сервисом;
8. отсутствие мобильного приложения;
9. неудобный интерфейс веб приложения;
10. низкая отзывчивость интерфейса ввиду использования устаревших технологий.

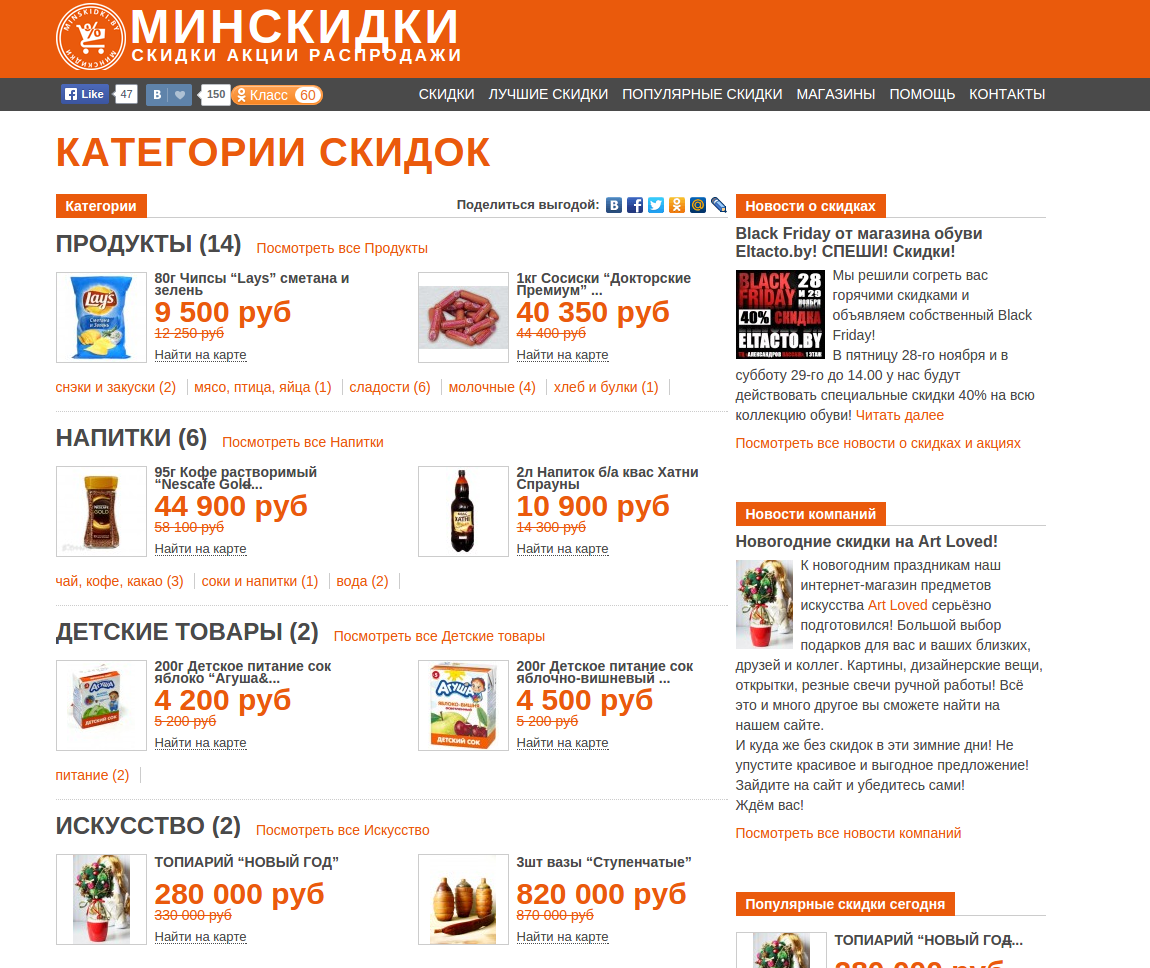
### **1.2.3** Минскскидки

Минскскидки – это бесплатный сайт, где можно просмотреть скидки, акции и распродажи в Минске. Минскидки — не купонный сайт, здесь не нужно платить, чтобы пользоваться скидками, акциями и распродажами, опубликованными на сайте [6]. Все скидки и акции предоставляются магазинами напрямую, но здесь не отдается предпочтение ни одному из них. С помощью сайта можно экономить, найдя лучшие скидки в Минске или скидки рядом с пользователем. На рисунке 1.3 представлена страница с категориями товаров приложения Минскскидки.

Достоинства:

1. бесплатность;
2. большой список представленных торговых сетей;
3. концентрация на минском рынке.
4. Недостатки:
5. отсутствие мобильного приложения;
6. отсутствие поддержки рынка регионов;
7. отсутствие продвинутых функциональных возможностей (поиск, категоризация).

Рисунок 1.2 – Главная страница приложения skidki.by

Рисунок 1.3 – Страница с категориями товаров

### **1.2.4** shoppingplus

Shoppingplus – это бесплатный способ отслеживать акции, скидки, распродажи и просто каталоги большого количества различных магазинов и компаний, а также получать и скидки и бонусы, не теряя времени на поиск и обработку информации – всё уже сделано за пользователя [7]. На рисунке 1.4 представлена страница с доступными категориями и картой.

Достоинства:

1. бесплатность;
2. большой список представленных торговых сетей;
3. автоматическое определение местоположения и предоставление скидок в зависимости от него.

Недостатки:

1. отсутствие мобильного приложения;
2. не доступен поиск по параметрам и фильтрация товаров и скидок.

## 1.3 Обоснование выбора технологий разработки

### **1.3.1** Выбор архитектуры

Для разработки программного продукта 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, представление, 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.

Ruby on Rails позиционируется как очень тщательно продуманная платформа для разработки. Полная поддержка MVC является основным преимуществом, которое 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.

Рисунок 1.4 – Категории товаров и карта в приложении shoppingplus

### **1.3.2** 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].

На мобильном клиенте выбор доступных СУБД еще более ограничен. В данном случае наилучшим решением является SQLite. SQLite – компактная встраиваемая реляционная база данных.

Слово «встраиваемый» означает, что SQLite не использует парадигму клиент-сервер, то есть движок SQLite не является отдельно работающим процессом, с которым взаимодействует программа, а предоставляет библиотеку, с которой программа компонуется и движок становится составной частью программы. Таким образом, в качестве протокола обмена используются вызовы функций (API) библиотеки SQLite. Такой подход уменьшает накладные расходы, время отклика и упрощает программу. SQLite хранит всю базу данных (включая определения, таблицы, индексы и данные) в единственном стандартном файле на том компьютере, на котором исполняется программа. Простота реализации достигается за счёт того, что перед началом исполнения транзакции записи весь файл, хранящий базу данных, блокируется; ACID-функции достигаются в том числе за счёт создания файла журнала [9].

Несколько процессов или потоков могут одновременно без каких-либо проблем читать данные из одной базы. Запись в базу можно осуществить только в том случае, если никаких других запросов в данный момент не обслуживается; в противном случае попытка записи оканчивается неудачей, и в программу возвращается код ошибки. Другим вариантом развития событий является автоматическое повторение попыток записи в течение заданного интервала времени.

В комплекте поставки идёт также функциональная клиентская часть в виде исполняемого файла sqlite3, с помощью которого демонстрируется реализация функций основной библиотеки. Клиентская часть работает из командной строки, позволяет обращаться к файлу БД на основе типовых функций ОС.

Благодаря описанным выше особенностям, SQLite идеально подходит для использования в небольших по мощности клиентских устройствах. В Android существует встроенная поддержка базы данных SQLite. Поддерживаются все функции SQLite, предоставляется API оболочки с совместимым интерфейсом. API Android SQLite является типичным; разработчику следует реализовать всю обработку базы данных, включая создание, управление версиями, обновления базы данных и прочие настройки. Если нужно использовать заранее заполненную базу данных SQLite, требуется дополнительная настройка.

Прямое использование API Android SQLite может привести к написанию большого количества шаблонного кода. Существует несколько библиотек Android, помогающих упростить этот процесс. Они обладают и рядом других возможностей в дополнение к удобному и эффективному использованию баз данных SQLite в приложениях Android.

### **1.3.3** Выбор языка программирования и платформы разработки

Для разработки выбран язык программирования Ruby, который имеет следующие достоинства:

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].

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 [11].

В качестве клиента приложения выбраны устройства на базе операционной системы Android. Android проявляет себя лучше своих конкурентов Apple iOS и Windows Phone, в ряде особенностей, таких как веб-сёрфинг, интеграция с сервисами Google Inc. и прочих. Также Android является открытой платформой, что позволяет реализовать на ней больше функций. Большую роль при выборе платформы сыграла ее популярность. По последним данным, мобильными устройствами на базе операционной системы Android пользуется в несколько раз больше людей, чем остальными платформами вместе взятыми, что позволит получить большую клиентскую базу в кратчайшие сроки.

Обмен данными между приложением клиента и сервером осуществляется посредствам протокола HTTP. HTTP – протокол прикладного уровня передачи данных (изначально – в виде гипертекстовых документов в формате HTML, в настоящий момент используется для передачи произвольных данных) [12]. 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. В данном проекте посредствам протокола HTTP будут передаваться данные в формате JSON или XML.

При разработке приложений под операционную систему Android преимущественно используется платформа Java. Это обусловлено многими факторами. Технология Java протестирована, усовершенствована, расширена и проверена участниками сообщества разработчиков Java, архитекторов и энтузиастов. Java позволяет разрабатывать высокопроизводительные портативные приложения практически на всех компьютерных платформах [13]. Доступность приложений в разнородных средах позволяет предоставлять более широкий спектр услуг, способствует повышению производительности, уровня взаимодействия и совместной работы конечных пользователей и существенному снижению стоимости совместного владения корпоративными и потребительскими приложениями. Java является очень удобным инструментом для разработчиков и открывает для них следующие возможности:

1. написание программного обеспечения на одной платформе и его запуск практически на любой другой платформе;
2. создание программ, работающих в веб-браузере и имеющих доступ к веб-службам;
3. разработка приложений на стороне клиента для глубокого взаимодействия с сервером;
4. объединение приложений или служб с использованием языка Java для создания высокоспециализированных приложений или служб;
5. создание многофункциональных и эффективных приложений для мобильных телефонов, удаленных процессоров, микроконтроллеров, беспроводных модулей, датчиков, шлюзов, потребительских продуктов и практически любых других категорий электронных устройств.

## 1.4 Укрупненная спецификация требований

Как показал анализ представленных выше существующих аналогов, они имеют одни и те же, причем характерные для многих веб-приложений, недостатки:

1. Недостаточно плавный и быстрый интерфейс. Эта проблема возникает в связи с использованием устаревших технологий при проектировании и разработке приложения. В некоторых случаях, когда можно перезагрузить только часть страницы используя технологии Ajax, страница перезагружается полностью, что, безусловно, сказывается как на скорости перезагрузки, так и на визуальном восприятии информации со страницы, так как при перезагрузке отдельной части страницы пользователь концентрируется именно на этой части, что позволяет ему не отвлекаться и с комфортом использовать приложение.
2. Также существенным недостатком систем с агрегацией данных с различных сайтов, какими и являются приложения по отображению скидок, ведь данные поступают на них с сайтов торговых сетей, является недостаточно своевременное обновление данных. Это возникает из-за недостаточной автоматизации процессов, а также использования ручных операций вместо автоматических.
3. Большинство рассмотренных аналогов не имеют мобильных приложений, что ограничивает их использование.
4. Многие из рассмотренных аналогов не имеют продвинутых средств по поиску (фильтрация скидок на товары и услуги по различным доступным параметрам, полнотекстовый поиск с учетом морфологии и ошибок и опечаток) и категоризации (разбиение скидок на товары и услуги на различные категории по всевозможным признакам, будь то размер скидки на товар или тип или цена товара).

Исходя из вышеперечисленных недостатков, разрабатываемое программное средство должно обладать следующими особенностями:

1. Использование современных технологий как на клиенте, так и на сервере, что позволит пользоваться интерфейсом приложения без задержек и с максимальным комфортом.
2. По возможности автоматический, а где это невозможно – максимально автоматизированный процесс получения данных о товарах и скидках на них. Для выполнения ручных операций предполагается создание максимально удобной панели администратора, включающей в себя все необходимое.
3. Наличие удобного и современного мобильного приложения.
4. Поиск и категоризация, включающие в себя широкий круг функциональных возможностей, причем как в серверном, так и в клиентском приложении. Данная возможность становится особенно удобной при использовании современных средств хранения и кеширования данных, когда при каждом действии пользователя не нужно будет отправлять запрос на сервер или обращаться к базе данных. Все необходимые данные будут доступны в оперативной памяти устройства, что сделает поиск чрезвычайно быстрым и приятным для пользователя.

К нефункциональным требованиям клиент-серверного приложения управления скидками на товары и услуги относятся:

1. версия браузера для работы веб-приложения:

* Microsoft Internet Explorer 10+;
* Mozilla Firefox 31+;
* Google Chrome 31+;
* Safari 7+;
* Google Chrome for Android 42+.

1. версия ОС Android выше 4.0.

# 2 Моделирование предметной области

## 2.1 Разработка функциональной модели предметной области

Для функционального моделирования применена диаграмма вариантов использования, являющаяся частью унифицированного языка моделирования (UML), а для ее построения – среда «Enterprise Architect» компании Sparx Systems [14]. Данная среда содержит мощный набор UML инструментов для анализа и дизайна на всех стадиях разработки программного обеспечения и обладает достаточной функциональностью для генерации документов для разрабатываемой системы. Среда «Enterprise Architect» предоставляет мощный механизм для генерации высококачественных настраиваемых документов прямо из UML модели.

UML (англ. Unified Modeling Language – унифицированный язык моделирования) – язык графического описания для объектного моделирования в области разработки программного обеспечения. UML является языком широкого профиля, это – открытый стандарт, использующий графические обозначения для создания абстрактной модели системы, называемой UML-моделью. UML был создан для определения, визуализации, проектирования и документирования, в основном, программных систем [15].

Для функционального моделирования предметной области в языке UML имеется диаграмма вариантов использования, относящаяся к диаграммам поведения. Диаграмма вариантов использования (англ. Use case diagram, диаграмма прецедентов) — диаграмма, на которой отражены отношения, существующие между актёрами и вариантами использования. Вместе они определяют сферу применения создаваемой системы. При этом первые описывают все то, что происходит внутри системы, а вторые – то, что происходит снаружи. Именно поэтому диаграмма вариантов использования применена при функциональном моделировании предметной области.

Общий вид укрупненной диаграммы вариантов использования программного средства для управления скидками представлен на рисунке 2.1.

Как видно из диаграммы на рисунке 2.1, в использовании программного средства будут участвовать 3 актера:

1. незарегистрированный пользователь (недоступны некоторые возможности приложения);
2. зарегистрированный пользователь (доступно больше возможностей);
3. администратор, осуществляющий управление данными приложения (пользователи, категории товаров и скидок, скидки).



Рисунок 2.1 – Диаграмма ролей и вариантов использования клиент-серверного приложения

## 2.2 Разработка информационной модели

На рисунке 2.2 представлена схема базы данных. Данная схема разработана в среде моделирования RailRoad [16].

База данных должна содержать следующие таблицы:

User – пользователь системы, причем как с администраторскими правами, так и без них. Данная таблица включает следующие основные поля:

1. First\_name. Содержит имя пользователя.
2. Last\_name. Содержит фамилию пользователя.
3. Email. Уникальный адрес электронной почты. Служит в качестве логина при аутентификации пользователя на сайте.
4. Encrypted\_password. Хеш-значение пароля пользователя вместе с солью. Необходимо для более тщательной защиты каждого из пользователей.
5. Role. Хранит роль пользователя.
6. Provider. Содержит название социальной сети, через которую был зарегистрирован пользователь.
7. Created\_at. Содержит дату создания пользователя.
8. Updated\_at. Содержит дату последнего обновления пользователя.

Store – магазин, предоставляющий скидки. Данная таблица включает следующие основные поля:

1. Name. Данное поле хранит название магазина.
2. Created\_at. Содержит дату создания магазина.

Discount – скидки, аггрегируемые с различных сайтов и отображаемые пользователю. Данная таблица включает следующие основные поля:

1. Content. Содержание объявления.
2. User\_id. Идентификатор пользователя, создавшего объявление.
3. State. Состояние объявления (черновик, на проверку, подтверждено и т.д.).
4. Price. Цена объявления.

Category – категория скидок. Данная таблица включает следующие основные поля:

1. Name. Данное поле хранит имя раздела.
2. Created\_at. Содержит дату создания раздела.

Bucket – корзина, в которую пользователь может добавлять скидки. Данная таблица включает следующие основные поля:

1. Name. Название корзины.

Report – отчет о новых скидках, подлежащих редактированию администратором. Данная таблица включает следующие основные поля:

1. Content – содержание отчета.

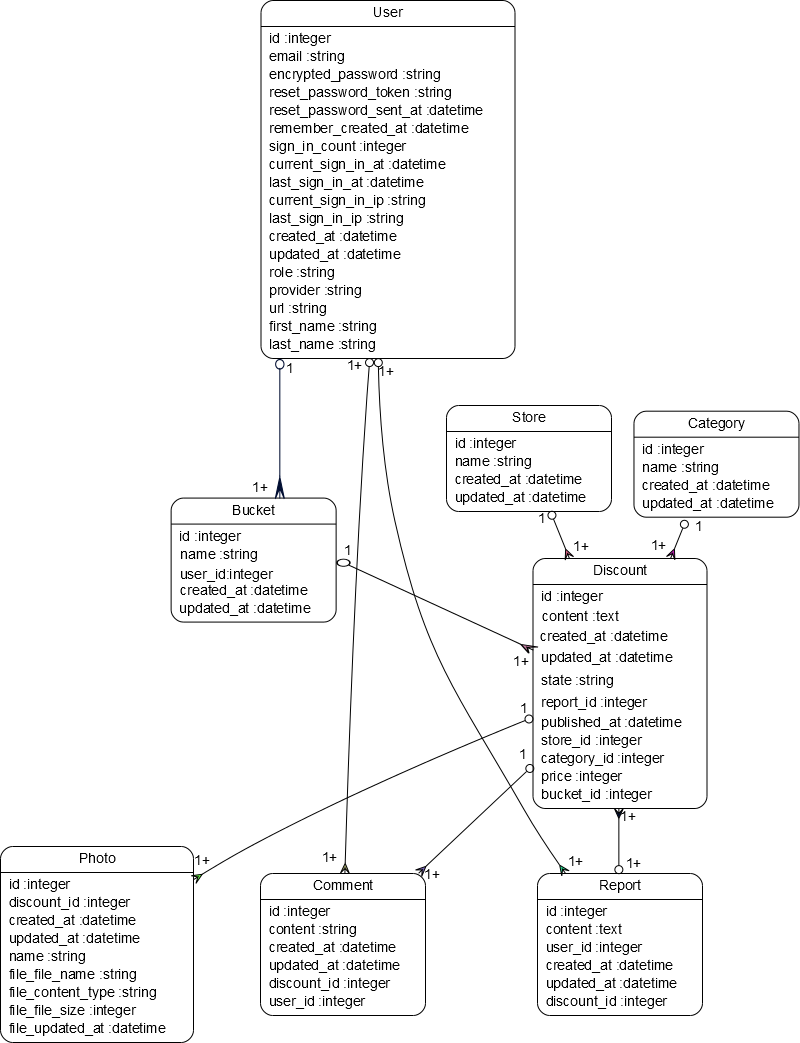


Рисунок 2.2 – Схема базы данных

Comment – комментарий пользователя к скидке. Данная таблица включает следующие основные поля:

1. Content. Содержание комментария.
2. Discount\_id. Содержит в себе уникальный идентификатор скидки.
3. User\_id. Содержит идентификатор пользователя, который оставил комментарий.

Photo – данные о файле с изображением продукта со скидкой. Данная таблица включает следующие основные поля:

1. Discount\_id. Содержит в себе уникальный идентификатор скидки.
2. Name. Дает пользователю возможность ввести имя фотографии.
3. File\_file\_name. Имя файла, загруженного пользователем.
4. File\_content\_type. Тип файла.
5. File\_size. Размер файла.

## 2.3 Разработка спецификации функциональных требований

Данная спецификация разработана на основе рисунка 2.1. Предполагается предоставление программным средством трех режимов работы, непосредственно связанными с ролями пользователей:

1. Режим гостя. В данном режиме ПС используют незарегистрированные пользователи. В нем доступна лишь часть функций, а именно:
   1. Просмотр товаров и скидок. Пользователь может просматривать все доступные в системе скидки на товары и услуги всех торговых предприятий. При просмотре доступно перелистывание страниц и просмотр скидок отдельных, указанных пользователем, предприятий. Данная функция доступна всем пользователям системы.
   2. Регистрация. Пользователь регистрирует себя в системе посредством ввода адреса своей электронной почты и пароля. Также доступна регистрация через наиболее популярные зарубежные и отечественные социальные сети. Данная функция недоступна другим типам пользователей.
2. Режим зарегистрированного пользователя. В данном режиме ПС используют пользователи, прошедшие регистрацию. В нем доступен максимально возможный для пользователей без прав администратора функционал ПС, а именно:
   1. Поиск скидок. Осуществляется с использованием различных присущих скидкам параметров, таких как цена, величина скидки в абсолютном и относительном выражении, категория скидки, название продукта (причем поиск будет осуществлен корректно даже с введенной с опечаткой частью названия).
   2. Авторизация. Пользователь должен ввести свои авторизационные данные для доступа к расширенному функционалу приложения.
   3. Создание корзины скидок. Пользователь может создать свою корзину скидок с уникальным для данного пользователя именем.
   4. Добавление скидок в корзину скидок. При просмотре скидок пользователь может добавлять их к себе в корзину.
3. Режим администратора. В данном режиме ПС могут использовать только пользователи с предоставленными правами администратора. В нем доступны абсолютно все функции ПС, в том числе:
   1. Управление пользователями. Администратор может редактировать данные существующих пользователей, удалять их, создавать новых, назначать пользователям роли, в том числе роль администратора.
   2. Поиск пользователей. Администратор должен иметь возможность искать и фильтровать пользователей по различным параметрам, таким как адрес электронной почты, имя, фамилия, для нахождения и последующего редактирования данных о пользователях.
   3. Управление категориями скидок. Возможность администратора создавать, редактировать, удалять просматривать категории скидок. При создании предполагается ввод данных новой категории и загрузка соответствующего ей изображения.
   4. Управление скидками. Возможность администратора создавать, редактировать, удалять просматривать скидки. При создании предполагается редактирование данных скидки, полученных аггрегатором скидок.
   5. Работа с отчетом о новых скидках за период. Предполагается получение администратором уведомления с соответствующим отчетом о новых скидках и дальнейшее редактирование и модерация администратором полученных данных, в результате которых будут сформированы списки новых скидок.

На рисунке 2.3 представлена диаграмма вариантов использования, детализирующая диаграмму на рисунке 2.1.

Управление пользователями доступно только администраторам. Оно включает в себя:

1. Просмотр детальной информации о пользователе (адрес электронной почты, имя, фамилия).
2. Создание пользователя с заданными параметрами.
3. Редактирование параметров пользователя.



Рисунок 2.3 – Детализированная диаграмма вариантов использования клиент-серверного приложения

Управление скидками также доступно только администраторам и состоит из следующих возможностей:

1. Просмотр детальной информации о скидке (изображение продукта, дата создания, редактирования, цена, величина скидки).
2. Создание скидки с желаемыми параметрами.
3. Редактирование параметров скидки.

Поиск скидок доступен как пользователям мобильного клиента приложения, так и администраторам, работающим с веб-приложением. Поиск включает в себя следующие возможности:

1. Фильтрация скидок по их характеристикам (цена, название продукта, величина скидки, дата создания и редактирования).
2. Отображение скидок только заданной категории или категорий.
3. Сортировка скидок по их характеристикам (цена, название продукта, величина скидки, дата создания и редактирования).

# 3 Проектирование клиент-серверного приложения

## 3.1 Разработка архитектуры

Приложение состоит из следующих компонентов (представлены на рисунке 3.1):

1. References – указатели на используемые библиотеки. Среди библиотек можно выделить наиболее важные:
   1. ActiveRecord – это модель в шаблоне MVC – модель – которая является слоем в системе, ответственным за представление бизнес-логики и данных. Active Record упрощает создание и использование бизнес-объектов, данные которых требуют постоянного хранения в базе данных. Сама по себе эта реализация паттерна Active Record является описанием системы ORM (Object Relational Mapping).
   2. Action Controller – это контроллер в шаблоне MVC. После того, как маршрутизация определит, какой контроллер использовать для обработки запроса, контроллер ответственен за обработку запроса и генерацию подходящего ответа. ActionController делает за программиста большую часть работы и использует соглашения, чтобы сделать это по возможности максимально просто. Для большинства приложений, основанных на [RESTful](http://ru.wikipedia.org/wiki/REST), контроллер получает запрос (это невидимо для разработчика), извлекает или сохраняет данные в модели и использует вью для создания результирующего HTML.
   3. Active Support – часть ядра Rails, которая предоставляет расширение языка Ruby, утилиты и другие возможности. Одна из них включает инструментарий API, который может использоваться внутри приложения, для отслеживания определенных действий, которые возникают как в коде Ruby, так и внутри приложения Rails и самой платформы. Однако, он не ограничен Rails. При необходимости его можно независимо использовать в других скриптах Ruby. Для обеспечения минимума влияния, Active Support по умолчанию ничего не загружает. Он разбит на маленькие части, поэтому можно загружать лишь то, что нужно, и имеет некоторые точки входа, которые по соглашению загружают некоторые расширения за раз, или даже все.
2. Controllers – обеспечивает связь между пользователем и системой: контролирует ввод данных пользователем и использует модель и представление для реализации необходимой реакции. Предполагается создание следующих контроллеров:

 Рисунок 3.1 – Диаграмма компонентов клиент-серверного приложения

* 1. ApplicationController – контроллер, в котором определены общие для всех контроллеров методы. От данного контроллера наследуются остальные контроллеры.
  2. MainController – контроллер, в котором определены методы для выполнения общих действий в системе, а именно: регистрация, авторизация, отображение главной страницы приложения.
  3. UsersController – контроллер, в котором определены методы для работы с пользователями.
  4. DiscountsController – контроллер, в котором определены методы для работы со скидками.
  5. BucketsController – контроллер, в котором определены методы для работы с корзинами скидок.

1. Models – предоставляет знания: данные и методы работы с этими данными, реагирует на запросы, изменяя своё состояние. Не содержит информации, как эти знания можно визуализировать. Предполагается создание файла базы данных DB.pdd и взаимодействие с ним модуля ActiveRecord::Base.
2. Views – отвечает за отображение информации (визуализацию). Часто в качестве представления выступает [форма (окно)](https://ru.wikipedia.org/wiki/%D0%9E%D0%BA%D0%BD%D0%BE_(%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)) с графическими элементами. Состоит из представлений, связанных с соответствующими контроллерами:
   1. Users. Включает в себя следующие представления:
      * + edit.html.slim – отображает поля для редактирования пользователя;
        + index.html.slim – отображает список пользователей;
        + new.html.slim –страница для создания пользователя;
        + show.html.slim – отображает информацию о пользователе;
   2. Main. Включает в себя следующие представления:
      * + index.html.slim – главная страница приложения;
        + sign\_in.html.slim – страница для ввода информации для авторизации;
        + sign\_up.html.slim – страница для ввода информации для регистрации.
   3. Discounts. Включает в себя следующие представления:
      * + edit.html.slim – отображает поля для редактирования скидок;
        + index.html.slim – отображает список скидок;
        + new.html.slim –страница для создания скидок;
        + show.html.slim – отображает информацию о скидке;
        + generate\_report.html.slim – страница для отображения отчета о новых скидках;
        + add\_category.html.slim – страница для добавления категорий скидок.
   4. Buckets. Включает в себя следующие представления:
      * + edit.html.slim – отображает поля для редактирования корзины;
        + index.html.slim – отображает список корзин;
        + new.html.slim –страница для создания корзины;
        + show.html.slim – отображает информацию о корзине;
        + add\_discount.html.slim – страница для добавления скидок в корзину.

## 3.2 Разработка схемы программы клиент-серверного приложения

Приложение является клиент-серверным, что в наибольшей степени определяет общий алгоритм его работы. Компоненты клиент-серверного приложения, клиент и сервер, взаимодействуют в данном случае через компьютерную сеть посредством протокола HTTP и находятся на разных вычислительных машинах. Программа-сервер ожидает от клиентской программы запросы, обрабатывает их и предоставляет свои ресурсы в виде данных. Инициатором установки соединения является клиентское приложение, поэтому схема работы программы, представленная на рисунке 3.2, начинается с действий именно на мобильном устройстве (блок 2).

После запуска мобильного приложения для осуществления многих действий нужны данные, хранящиеся в базе данных на сервере. После выполнения пользователем этих действий (блок 3) осуществляется его обработка и анализ в мобильном приложении. Если для отображения результата этого действия не нужны данные из серверной базы данных (блок 4), то пользователю сразу отображается интересующее его представление. Если же отображение результата действия невозможно без дополнительных данных, тогда в первую очередь проверяется, закешированы ли необходимые данные на мобильном устройстве (блок 5). Для кеширования в данном случае используется база данных SQLite. В случае, если подобное действие ранее выполнялось, данные были закешированы и сейчас могут использоваться. Если же необходимые данные не закешированы, осуществляется подготовка запроса на необходимые данные (блок 6), что включает в себя формирование запроса из данных сервера, хранящихся в программных настройках приложения, а также из данных, введенных клиентом при осуществлении действия (например, идентификатор магазина).



Рисунок 3.2 – Схема работы программы

Сформированный запрос отправляется посредствам указанного выше протокола HTTP на сервер (блок 8). На серверной машине постоянно запущен веб-сервер Nginx, который принимает все запросы, приходящие на 80 порт. Если запрос был сформирован неверно (это возможно при использовании сторонних клиентских приложений), веб-сервер посылает сообщение об ошибке, которое представляет собой структуру данных в указанном в запросе формате, которая включает в себя код ошибки и сообщение, детализирующее ее. Если используется клиентское приложение, созданное в рамках данного дипломного проекта, таких ошибок возникать не будут. После успешной обработки запроса, веб-сервер переотправляет запрос серверу приложения, работающему на платформе Ruby (блок 15), где и осуществляется основная часть всей обработки запроса и отправки ответа. Опять же, при использовании сторонних клиентских приложений на данном этапе может возникнуть ошибка, связанная с неправильным формированием запроса или указанием в нем недействительных данных. В данном случае сервер также отправляет на клиент ответ с кодом ошибки и сообщением (блок 23).

Если запрос сформирован корректно, начинается его непосредственная обработка контроллером, началу которой соответствует блок 19. Шаблон проектирования MVC предполагает использование небольших по объему классов контроллеров. Если же для формирования ответа необходимо осуществление большого количества операций, регулируемых сложно логикой, часть работы выполняется в специализированных службах. После получения необходимых данных контроллером или сервисом начинается их формирование в указанном в запросе клиентом формате (это может быть JSON, XML) и дальнейшая их отправка на клиент (блок 20).

Когда клиентское приложение получает отправленные сервером данные, оно начинает их обработку и формирует из них объекты языка, используемого в работе приложения. При необходимости полученные данные кешируются (блок 30) для дальнейшего использования без обращений к серверу, что намного увеличивает отзывчивость интерфейса и общую скорость работы приложения. Далее формируется представление с полученными данными и отображается пользователю. При получении сообщения об ошибке, на клиенте формируется и отображается окно с полученным сообщением (блок 36).

## 3.3 Разработка схемы алгоритма работы администратора приложения

Большую часть данных, которые хранятся на сервере, формирует и добавляет администратор, поэтому эта часть работы приложения детально описана в схеме на рисунке 3.3.



Рисунок 3.3 – Схема работы администратора

Для использования администраторского веб-приложения пользователю-администратору нужен аккаунт. Создание данного аккаунта, которое представлено в блоке 12), возможно только другим администратором или через базу данных на сервере. Это сделано для того, чтобы ограничить доступ к приложению сторонними лицами. После создания аккаунта администратору необходимо авторизоваться (блок 20), введя свои указанные при создании адрес почтового ящика и пароль. После авторизации администратору предоставляется доступ к приложению. В верхней части страницы располагается меню со ссылками на доступные компоненты системы. Данное меню состоит из четырех пунктов: список пользователей, настройки, список магазинов, выход.

При переходе к списку пользователей отображается таблица с добавленными в систему пользователями (блок 1). Колонками в таблице являются их параметры. На данной странице доступна фильтрация и сортировка пользователей, добавление новых пользователей, переход на страницу для редактирования существующих пользователей.

После нажатия на ссылку настроек отображается страница с настройками системы (блок 6). Администратор может просмотреть настройки и нажать кнопку для возвращения на главную страницу приложения, а также может отредактировать настройки и нажать на кнопку сохранения, в результате чего изменения будут сохранены, и отобразится главная страница приложения.

При переходе к списку магазинов (блок 26) отображается таблица с добавленными в систему магазинами. В данной таблице отображены основные параметры магазинов. Далее возможен переход на главную страницу приложения (блок 9). Также возможен выбор пункта для редактирования магазина, в результате чего отображается страница с параметрами магазина. После изменения необходимых полей и сохранения происходит перенаправление на страницу с магазинами. При нажатии на кнопку удаления магазина отображается диалоговое окно для подтверждения. При подтверждении магазин удаляется. Диалоговое окно закрывается и отображается страница со списком магазинов. После выбора пункта просмотра листов скидок отображаются добавленные для данного магазина листы. На данном этапе доступно удаление ненужных листов и переход на страницу магазинов. Для каждого магазина также доступна операция обновления листов скидок, которая осуществляется при нажатии на соответствующую кнопку. Далее отображается список магазинов. Для каждого магазина доступно отображение списка его скидок, представленных в виде таблицы. В данном режиме возможно удаление и редактирование скидок магазина, а также переход на страницу списка магазинов.

Последним пунктов меню является пункт выхода из системы (блок 37). При нажатии на данный пункт сессия приложения завершается. Для последующего входа в систему необходима авторизация.

## 3.4 Разработка схемы алгоритма добавления скидок администратором

Наиболее часто выполняемой процедурой, которая осуществляется администратором, является добавление скидок, схема чего представлена на рисунке 3.4.

Для автоматизации данного процесса необходима настройка магазинов. В рамках этого вносится расписание обновления листов скидок на официальном сайте магазина (блок 2), добавляется адрес страницы официального сайта, на которой размещена ссылка на листы скидок, а также указывается селектор для получения этого адреса.

После выполнения описанной конфигурации может быть осуществлено добавление в базу данных листов скидок при их наличии на официальном сайте. Если создание листов на официальном сайте соответствует расписанию, указанному в настройках магазина (блок 6), процесс обновления листов начинается автоматически (блок 8). Если создание внеплановое, то для инициализации обновления необходимы действия администратора (блок 7).

Далее осуществляется обновление листов и их разрезание на отдельные скидки. Если изображение обрезано неровно или параметры скидки распознаны неверно, возможно ручное редактирование администратором. После создания всех скидок они сохраняются в базу данных и кешируются.



Рисунок 3.4 – Схема добавления скидок администратором

# 4 Тестирование клиент-серверного приложения

Для оценки правильности работы клиент-серверного приложения управления скидками проведено тестирование клиента и веб-приложения.

Тест-кейсы для функционального требования «Управление магазинами», реализуемого в веб-приложении, представлены в таблице 4.1.

Таблица 4.1 – Тестирование функции «Управление магазинами»

|  |  |  |
| --- | --- | --- |
| Название тест-кейса и его описание | Ожидаемый результат | Полученный результат |
| 1 | 2 | 3 |
| Создание магазина  1. Загрузить программное средство.  2. Перейти на страницу магазинов.  3. Нажать кнопку для перехода на страницу создания магазина Create.  4. Заполнить свойства магазина.  5. Нажать клавишу Enter | 1. Программное средство загружается и отображается главная страница.  2. Отображается страница со списком магазинов.  3. Отображается страница с формой для создания магазина.  4. Необходимые поля доступны для заполнения.  5. Магазин успешно добавлен в систему и отображается вместе с остальными в списке магазинов. | 1. Программное средство загружается и отображается главная страница.  2. Отображается страница со списком магазинов.  3. Отображается страница с формой для создания магазина.  4. Необходимые поля доступны для заполнения.  5. Магазин успешно добавлен в систему и отображается вместе с остальными в списке магазинов. |
| Удаление магазина  1. Загрузить программное средство.  2. Перейти на страницу магазинов.  3. Нажать кнопку Delete для выбранного магазина. | 1. Программное средство загружается и отображается главная страница.  2. Отображается страница со списком магазинов.  3. Магазин успешно удален из системы и больше не отображается в списке магазинов. | 1. Программное средство загружается и отображается главная страница.  2. Отображается страница со списком магазинов.  3. Магазин успешно удален из системы и больше не отображается в списке магазинов. |

Продолжение таблицы 4.1

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| Редактирование магазина  1. Загрузить программное средство.  2. Перейти на страницу магазинов.  3. Нажать кнопку для перехода на страницу редактирования магазина Edit.  4. Изменить свойства магазина.  5. Нажать клавишу Enter | 1. Программное средство загружается и отображается главная страница.  2. Отображается страница со списком магазинов.  3. Отображается страница с формой для редактирования магазина.  4. Необходимые поля доступны для редактирования.  5. Магазин успешно сохранен в системе и отображается с измененными полями в списке магазинов. | 1. Программное средство загружается и отображается главная страница.  2. Отображается страница со списком магазинов.  3. Отображается страница с формой для редактирования магазина.  4. Необходимые поля доступны для редактирования.  5. Магазин успешно сохранен в системе и отображается с измененными полями в списке магазинов. |
| Просмотр скидок заданного магазина  1. Загрузить программное средство.  2. Перейти на страницу магазинов.  3. Нажать кнопку Discounts для выбранного магазина. | 1. Программное средство загружается и отображается главная страница.  2. Отображается страница со списком магазинов.  3. Отображается страница со списком скидок для данного магазина. | 1. Программное средство загружается и отображается главная страница.  2. Отображается страница со списком магазинов.  3. Отображается страница со списком скидок для данного магазина. |
| Просмотр листов заданного магазина  1. Загрузить программное средство.  2. Перейти на страницу магазинов.  3. Нажать кнопку Sheets для выбранного магазина. | 1. Программное средство загружается и отображается главная страница.  2. Отображается страница со списком магазинов.  3. Отображается страница со списком листов скидок для данного магазина. | 1. Программное средство загружается и отображается главная страница.  2. Отображается страница со списком магазинов.  3. Отображается страница со списком листов скидок для данного магазина. |

Продолжение таблицы 4.1

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| Обновление листов магазина  1. Загрузить программное средство.  2. Перейти на страницу магазинов.  3. Нажать кнопку Update sheet.  4. Перейти на страницу листов магазина. | 1. Программное средство загружается и отображается главная страница.  2. Отображается страница со списком магазинов.  3. Запрос обрабатывается, выводится сообщение об успешном результате обновления.  4. В списке листов магазина появился новый элемент. | 1. Программное средство загружается и отображается главная страница.  2. Отображается страница со списком магазинов.  3. Запрос обрабатывается, выводится сообщение об успешном результате обновления.  4. В списке листов магазина появился новый элемент. |

Тест-кейсы для функционального требования «Управление скидками», реализуемого в веб-приложении, представлены в таблице 4.2.

Таблица 4.2 – Тестирование функции «Управление скидками»

|  |  |  |
| --- | --- | --- |
| Название тест-кейса и его описание | Ожидаемый результат | Полученный результат |
| 1 | 2 | 3 |
| Создание скидки заданного магазина  1. Загрузить программное средство.  2. Перейти на страницу магазинов. 3. Нажать кнопку Discounts для выбранного магазина.  4. Нажать кнопку для перехода на страницу создания скидки Create. | 1. Программное средство загружается и отображается главная страница.  2. Отображается страница со списком магазинов.  3. Отображается страница  со списком скидок для данного магазина.  4. Отображается страница с формой для создания скидки.  5. Необходимые поля доступны для заполнения. | 1. Программное средство загружается и отображается главная страница.  2. Отображается страница со списком магазинов.  3. Отображается страница  со списком скидок для данного магазина.  4. Отображается страница с формой для создания скидки.  5. Необходимые поля доступны для заполнения. |

Продолжение таблицы 4.2

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| 5. Заполнить свойства скидки.  6. Нажать клавишу Enter | 6. Скидка успешно добавлена в систему и отображается вместе с остальными в списке скидок заданного магазина. | 6. Скидка успешно добавлена в систему и отображается вместе с остальными в списке скидок заданного магазина. |
| Редактирование скидки заданного магазина  1. Загрузить программное средство.  2. Перейти на страницу магазинов.  3. Нажать кнопку Discounts для выбранного магазина.  4. Нажать кнопку для перехода на страницу редактирования скидки Edit.  5. Изменить свойства скидки.  6. Нажать клавишу Enter | 1. Программное средство загружается и отображается главная страница.  2. Отображается страница со списком магазинов.  3. Отображается страница  со списком скидок для данного магазина.  4. Отображается страница с формой для создания скидки с заполненными  существующими полями.  5. Необходимые поля доступны для редактирования.  6. Скидка успешно отредактирована и отображается с измененными параметрами. | 1. Программное средство загружается и отображается главная страница.  2. Отображается страница со списком магазинов.  3. Отображается страница  со списком скидок для данного магазина.  4. Отображается страница с формой для создания скидки с заполненными  существующими полями.  5. Необходимые поля доступны для редактирования.  6. Скидка успешно отредактирована и отображается с измененными параметрами. |
| Удаление скидки  1. Загрузить программное средство.  2. Перейти на страницу магазинов.  3. Нажать кнопку Discounts для выбранного магазина. | 1. Программное средство загружается и отображается главная страница.  2. Отображается страница со списком магазинов.  3. Отображается страница со списком скидок для данного магазина. | 1. Программное средство загружается и отображается главная страница.  2. Отображается страница со списком магазинов.  3. Отображается страница со списком скидок для данного магазина. |

Продолжение таблицы 4.2

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| 4. Нажать кнопку для удаления скидки Delete.  5. Подтвердить удаление в диалоговом окне. | 4. Отображается диалоговое окно для подтверждения удаления скидки.  6. Скидка успешно удалена из системы и не отображается вместе с остальными в списке скидок заданного магазина. | 4. Отображается диалоговое окно для подтверждения удаления скидки.  6. Скидка успешно удалена из системы и не отображается вместе с остальными в списке скидок заданного магазина. |

Тест-кейсы для функционального требования «Просмотр товаров и скидок», реализация которого представлена на мобильном Android клиенте приложения, представлены в таблице 4.2.

Таблица 4.3 – Тестирование функции «Просмотр товаров и скидок»

|  |  |  |
| --- | --- | --- |
| Название тест-кейса и его описание | Ожидаемый результат | Полученный результат |
| Выбор магазина для просмотра скидок  1. Загрузить программное средство.  2. Нажать кнопку в верхнем левом углу экрана. | 1. Программное средство загружается и отображается главная страница.  2. Отображается список магазинов. | 1. Программное средство загружается и отображается главная страница.  2. Отображается список магазинов. |
| Просмотр товаров и скидок  1. Загрузить программное средство.  2. Нажать кнопку в верхнем левом углу экрана.  3. Нажать на один из магазинов. | 1. Программное средство загружается и отображается главная страница.  2. Отображается список магазинов.  3. Отображается список товаров и скидок, доступный в выбранном магазине. | 1. Программное средство загружается и отображается главная страница.  2. Отображается список магазинов.  3. Отображается список товаров и скидок, доступный в выбранном магазине. |

Таким образом результат тестирования подтверждает, что клиент-серверное приложение управления скидками на товары и услуги функционирует в полном соответствии со спецификацией требований.

# 5 Методика использования разработанного клиент-серверного приложения

## 5.1 Работа со средой для управления скидками администратором

Среда для управления скидками администратором представляет собой веб приложение, которое может работать в любой операционной системе с установленным современным браузером. Корректная работа приложения гарантируется в браузерах, перечисленных в разделе, описывающем анализ предметной области и укрупненную спецификацию требований.

Данная методика использования приложения составлена с использованием браузера Google Chrome 42.

После запуска данного программного средства появляется страница авторизации, форма которой представлена на рисунке 5.1.

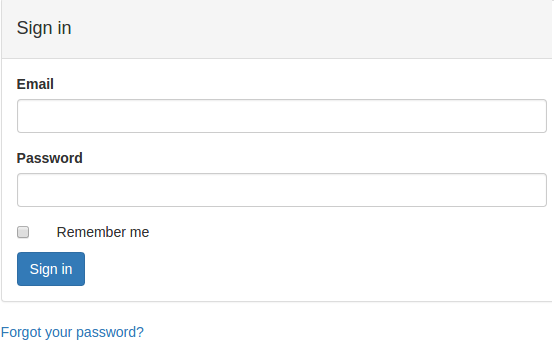


Рисунок 5.1 – Форма авторизации

Страница авторизации содержит три поля ввода:

1. email администратора;
2. пароль администратора;
3. опция сохранения авторизационных данных в куках браузера для последующего входа без повторного ввода данных.

Если пользователь забыл свой пароль, доступен его сброс и последующая переустановка через email.

После авторизации пользователь перенаправляется на страницу со списком магазинов, представленную на рисунке 5.2. На данной странице пользователь может переходить на страницу для создания нового магазина, нажав на кнопку «Create», а также просматривать некоторые данные уже существующих магазинов, а именно:

1. название магазина;
2. логотип;
3. дата добавления магазина в систему;
4. дата обновления магазина.

Также на данной странице доступно выполнение ряда действий над каждым магазином:

1. обновлять списки доступных для каждого магазина скидок;
2. удалять магазин;
3. переходить на страницу для редактирования магазина;
4. переходить на страницу со списком скидок для данного магазина;
5. переходить на страницу со списком листов магазина.

Форма для создания нового магазина представлена на рисунке 5.3. Она включает в себя следующие компоненты:

1. кнопки сохранения внесенных данных и отмены и возврата на предыдущую страницу;
2. поле для ввода названия магазина;
3. поля для ввода изображения логотипа магазина. Доступна загрузка как локального файла изображения, так и файла, размещенного по ссылке, введенной в поле «logo url»;
4. поля для настройки обрезания листов со скидками:
5. page with link url – поле, в которое необходимо вводить адрес страницы на официальном сайте магазина, на которой находится ссылка для просмотра и скачивания листов со скидками, доступными в данный момент в магазине;

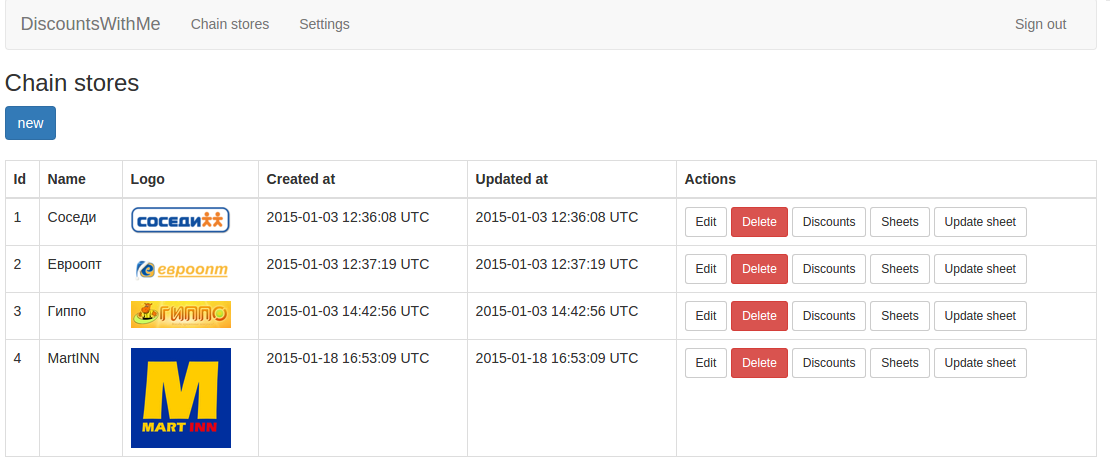


Рисунок 5.2 – Страница со списком магазинов

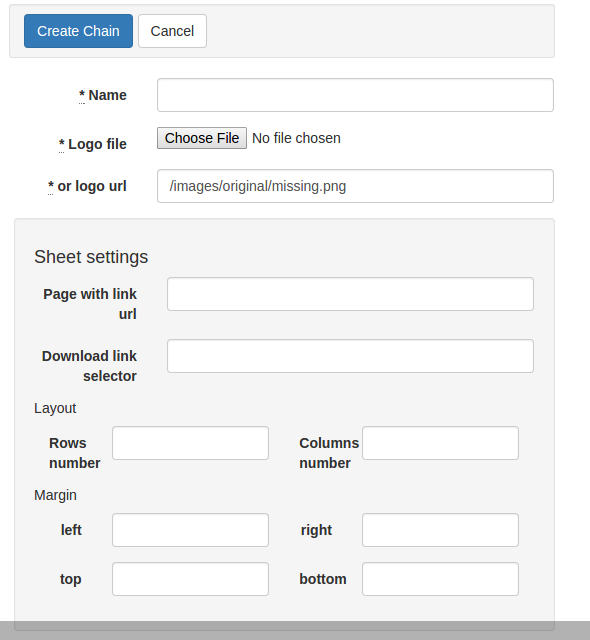


Рисунок 5.3 – Форма создания магазина

1. download link selector – css селектор, по которому располагается ссылка для просмотра и скачивания листов со скидками на официальном сайте магазина;
2. layout – настройки шаблона расположения и количества скидок на листе;
   * rows number – количество товаров со скидкой, расположенных в одном ряду на листе;
   * columns number – количество товаров со скидкой в одном столбце на листе.
3. margin – отступы от скидок до краев листа:
   * left – левый;
   * right – правый;
   * top – верхний;
   * bottom – нижний.

После нажатия на кнопку «Discounts» в списке магазинов, отображается страница, представленная на рисунке 5.4, со списком продуктов, доступных по скидке в выбранном магазине. На ней доступны:

1. кнопка для перехода на страницу создания нового продукта, на которой расположена форма для ввода полей продукта;
2. кнопка для возврата на предыдущую страницу;
3. таблица со списком продуктов.

В таблице представлены следующие колонки:

1. id – идентификатор продукта в базе данных, по которому осуществляется доступ к продукту;
2. product name – название продукта;
3. image – изображение продукта;
4. price – цена продукта;
5. economy – величина скидки;
6. created at – дата добавления продукта в систему;
7. updated at – дата последнего обновления продукта;
8. actions – список действий, доступных над данным продуктом:
   1. edit – переход на страницу редактирования продукта;
   2. delete – удаление продукта.

После нажатия на кнопку «Sheets» в списке магазинов, отображается страница, представленная на рисунке 5.5, со списком листов, загруженных для заданного магазина. В данной таблице представлены следующие колонки:

1. images таблицы со списком представлены изображения заданного листа, перечисленные в обратном порядке следования в буклете;
2. created at – дата добавления продукта в систему;
3. updated at – дата последнего обновления продукта;
4. actions – список действий, доступных над данным продуктом:
   1. edit – переход на страницу редактирования продукта;
   2. delete – удаление продукта.



Рисунок 5.4 – Список товаров заданного магазина

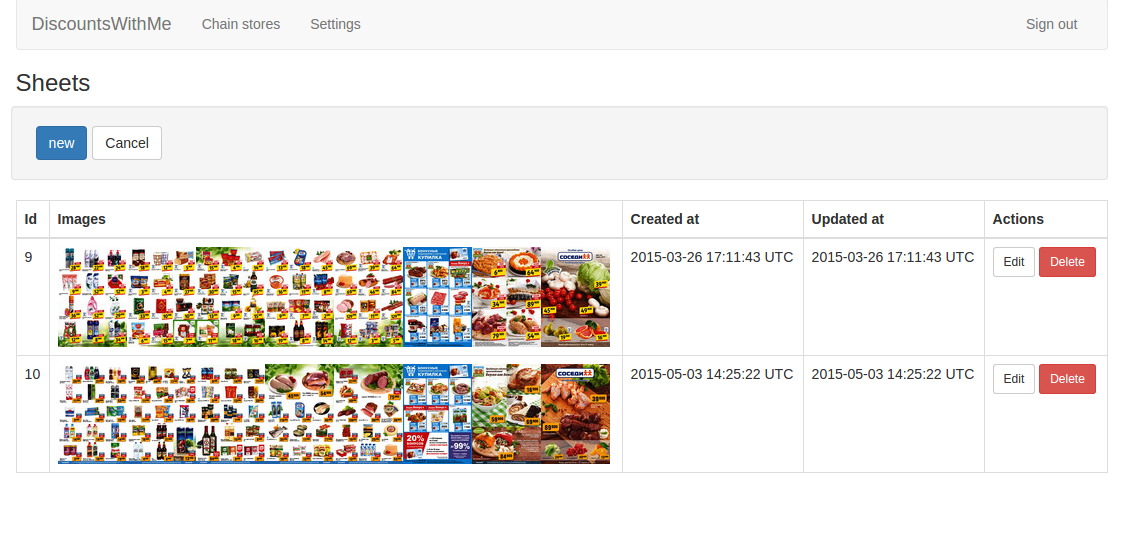


Рисунок 5.5 – Список листов скидок заданного магазина

## 5.2 Работа с мобильным приложенем для просмотра скидок

Мобильное приложение для просмотра скидок рассчитано на работу в операционной системе Android версий, указанных в укрупненной спецификации требований.

После запуска данного программного средства появляется список магазинов который представлена на рисунке 5.6.



Рисунок 5.6 – Список магазинов на мобильном приложении

Доступны элементы интерфейса для осуществления поиска необходимых пользователю скидок. Поиск инициализируется путем нажатия на значок лупы. Для его выполнения необходимо ввести данные в появившееся поле ввода. При выборе определенного магазина будет отображен список скидок, изображенный на рисунке 5.7.

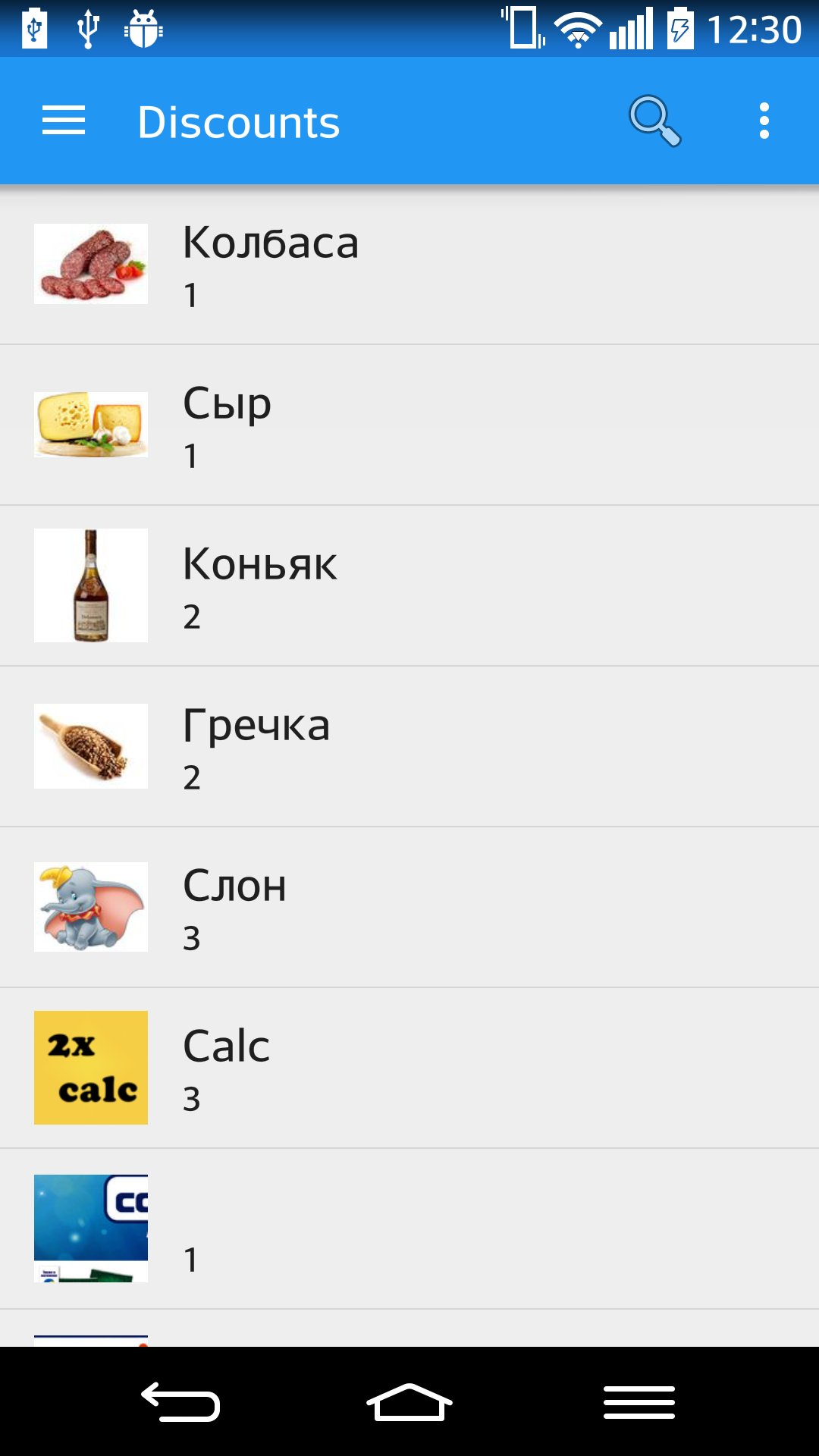


Рисунок 5.7 – Список скидок магазина на мобильном приложении

Данный список представлен в виде таблицы, в первом столбце которой размещено изображение продукта на скидке, во втором – название продукта и его цена. Возможно пролистывание и просмотр всех скидок, доступных в выбранном магазине.

В верхней части экрана расположены элементы интерфейса для осуществления поиска необходимых пользователю скидок. Поиск инициализируется путем нажатия на значок лупы. Для его выполнения необходимо ввести данные в появившееся поле ввода. При нажатии на элемент списка, будет отображена детальная информация о данной скидке, которая включает в себя цену продукта или услуги, размер скидки, период ее действия в магазине.

# 6 Технико-экономическое обоснование разработки и использования клиент-серверного приложения управления скидками

## 6.1 Характеристика программного продукта

Разрабатываемое в данном дипломе программное средство для управления скидками должно не только выполнять требуемые функции, но и быть экономически выгодным. Данный программный продукт предназначен для повышения эффективности поиска всевозможной информации, связанной со скидками, а также для повышения эффективности агрегации и сбора данной информации для последующего ее представления в удобочитаемом виде. В настоящее время программное обеспечение представляет собой товар, выпускаемый научно-техническими организациями, который поставляется покупателям по рыночным ценам. Экономический эффект зависит от объема затрат на разработку проекта, уровня цены на разработанный программный продукт и объема продаж. Выбор эффективных проектов программного обеспечения требует их экономической оценки и расчета экономического эффекта.

Для определения эффективности разработки и использования данного программного средства производится расчёт суммы расходов на разработку.

## 6.2 Экономический эффект у разработчика

Данный программный модуль относится к 1 категории сложности, так как он работает в реальном времени и может собирать информацию с нескольких удалённых объектов. При этом дополнительный коэффициент сложности равняется 1,12. По степени новизны ПС относится к категории А с коэффициентом новизны "K" \_н=1, так как принципиальных аналогов выявлено не было. Но данный проект не подразумевает в себе использование принципиально нового типа ЭВМ или ОС.

При разработке модуля используются существующие технологии и средства разработки, которые охватывают около 20 – 30% реализуемых функций, поэтому коэффициент использования стандартных модулей принимается равным 0,8.

### **6.2.1** Определение объема и трудоемкости ПО

Для того, чтобы рассчитать плановую смету затрат на разработку ПО, требуется определить общий объем ПС ("V" \_0). В качестве единицы измерения возьмем количество строк исходного кода (LinesofCode, LOC). Объем разрабатываемых функций определяется по каталогу функций (см. таблицу 6.1).

Таблица 6.1 – Каталог функций ПО

|  |  |  |  |
| --- | --- | --- | --- |
| Код функции | Наименование (содержание) функции | Объем функций по каталогу | Объем функций уточнённый |
| 1 | 2 | 3 | 4 |
| 101 | Организация ввода информации | 150 | 130 |
| 102 | Контроль, предварительная обработка и ввод информации | 450 | 420 |
| 204 | Обработка наборов и записей базы данных | 500 | 480 |
| 207 | Манипулирование данными | 1000 | 1100 |
| 208 | Организация поиска и поиска в базе данных | 500 | 550 |
| 309 | Формирование файла | 1020 | 850 |
| 506 | Обработка ошибочных и сбойных ситуаций | 410 | 490 |
| 507 | Обеспечение интерфейса между компонентами | 970 | 1250 |
| 704 | Процессор отчетов | 3200 | 2700 |
| 707 | Графический вывод результатов | 480 | 530 |
|  | Итого | 8680 | 8500 |

На основе общего объема ПС рассчитывается нормативная трудоемкость ПО (). Так как общий объем ПС = 8680, категория сложности ПО – 1, нормативная трудоемкость = 361. Тогда общая трудоемкость разработки определяется по формуле:

|  |  |
| --- | --- |
|  | (6.1) |

где – дополнительный коэффициент сложности;

– коэффициент, учитывающий использование типовых программ и модулей;

– коэффициент новизны.

Подставляя значения в формулу 6.1, получим:

(человеко-дней)

На основе общей трудоемкости определяется плановое число разработчиков по формуле:

|  |  |
| --- | --- |
|  | (6.2) |

где – численность исполнителей проекта;

– общая трудоемкость разработки проекта (чел./дн.);

– срок разработки проекта (лет);

– эффективный фонд времени работы одного работника в течение года (дн).

Срок разработки проекта составляет полгода ( = 0,5 год). Эффективный фонд времени определяется по формуле:

|  |  |
| --- | --- |
| , | (6.3) |

где – количество дней в году;

– количество праздничных дней в году;

– количество выходных дней в году;

– количество дней отпуска.

Подставляя значения в формулу 6.3, получим:

(дней).

При решении сложных задач с длительным периодом разработки ПО трудоемкость определяется по стадиям разработки: техническое задание (ТЗ), эскизный проект (ЭП), технический проект (ТП), рабочий проект (РП), внедрение (ВН). При этом трудоемкость разработки ПО отличается в зависимости от стадий. Общий вид формулы, по которой рассчитывается трудоемкость изготовления ПС:

|  |  |
| --- | --- |
| , | (6.4) |

где – трудоемкость изготовления ПО на данной стадии;

– нормативная трудоемкость;

– удельный вес трудоемкости выбранной стадии разработки ПО в общей трудоемкости разработки ПО.

При этом для стадии «Рабочий проект» полученное значение трудоемкости изготовления ПО требуется умножить на . Общая трудоемкость определяется как сумма трудоемкости изготовления ПО на каждой из стадий разработки. Результаты расчетов трудоемкости по стадиям сведены в таблицу 6.2.

Таблица 6.2 – Расчет общей трудоемкости разработки ПО

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Показатели | Стадии | | | | | Итого |
| ТЗ | ЭП | ТП | РП | ВН |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1. Коэффициенты удельных весов трудоемкости стадии разработки ПО () | 0,11 | 0,09 | 0,11 | 0,55 | 0,14 | 1,00 |
| 2. Коэффициент сложности | 1,12 | 1,12 | 1,12 | 1,12 | 1,12 |  |
| 3. Коэффициент, учитывающий использование стандартных модулей |  |  |  | 0,80 |  |  |
| 4. Коэффициент, учитывающий новизну ПО () | 1,00 | 1,00 | 1,00 | 1,00 | 1,00 |  |
| 5. Общая трудоемкость ПО (), чел./дн. | 35,53 | 29,07 | 35,53 | 177,65 | 45,22 | 323 |

На основе уточненной трудоемкости разработки ПО с использованием формулы 9.2 найдем общую численность разработчиков, которые требуются, чтобы вложиться в заданные сроки:

### **6.2.2** Расчет сметы затрат и цены заказного ПО

Основная статья расходов на создание ПО является заработная плата разработчиков проекта, то есть людей, непосредственно занимающихся разработкой.

Месячная тарифная ставка каждого специалиста () определяется по формуле:

|  |  |
| --- | --- |
| , | (6.5) |

где – месячная тарифная ставка первого разряда (тыс.руб.);

– тарифный коэффициент, соответствующий установленному тарифному разряду.

Часовая тарифная ставка :

|  |  |
| --- | --- |
|  | (6.6) |

где – часовая тарифная ставка (тыс.руб);

– среднемесячная норма рабочего времени в часах (составляет 170 часов).

Расчет месячных и часовых тарифных ставок сведен в таблицу 9.3 с учётом того, что месячная тарифная ставка первого разряда на апрель 2015 года составляет 292 (тыс.руб.).

Таблица 6.3 – Расчет месячных и почасовых тарифных ставок

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Должность | Количество ставок | Тарифный разряд | Тарифный коэффициент | Месячная тарифная ставка (руб.) | Часовая тарифная ставка (руб.) |
| Ведущий инженер-программист | 1 | 14 | 3,25 | 780000 | 4642,86 |
| Инженер-программист 1-ой категории | 4 | 12 | 2,84 | 681600 | 4057,14 |

Основная заработная плата исполнителей рассчитывается по формуле:

|  |  |
| --- | --- |
| , | (6.7) |

где *n* – количество исполнителей;

– часовая тарифная ставка i-го исполнителя (ден.ед.);

– количество часов работы в день, ч;

– коэффициент премирования;

– плановый фонд рабочего времени -го исполнителя (дн.).

(руб).

Дополнительная заработная плата () включает в себя оплаты отпусков и другие выплаты, предусмотренные законодательством, и определяется по формуле:

|  |  |
| --- | --- |
|  | (6.8) |

где – норматив дополнительной заработной платы (10-20%).

|  |  |
| --- | --- |
|  |  |

Отчисления в фонд социальной защиты определяется по формуле:

|  |  |
| --- | --- |
|  | (6.9) |

где – норматив отчислений в фонд социальной защиты наделения.

Отчисления в фонд социальной защиты – 34%, отчисления в фонд социального страхования – 0,6%. Исходя из этого, получаем:

|  |  |
| --- | --- |
|  |  |

Расходы по статье «Материалы» отражают расходы на магнитные носители, бумагу, тонер и прочие вещи, необходимые для разработки ПО. Нормы расхода материалов в суммарном выражении () определяется в расчете на 100 строк исходного кода. Сумма затрат на расходные материалы определяется по формуле:

|  |  |
| --- | --- |
|  | (6.10) |

где – норма расхода материалов в расчете на 100 строк исходного кода ПО (руб);

– общий объем ПО (строк исходного кода).

|  |  |
| --- | --- |
|  |  |

Расходы по статье «Машинное время» включает оплату машинного времени, необходимого для разработки и отладки ПО. Норматив на 100 строк исходного кода () зависит от характера решаемых задачи и типа приложений. Расходы по этой статье определяются по формуле:

|  |  |
| --- | --- |
| , | (6.11) |

где – цена одного машино-часа (руб);

– общее время работы над проектом (часов).

(руб).

Расходы по статье «Научные командировки» () определяются по формуле:

|  |  |
| --- | --- |
|  | (6.12) |

где – норматив расходов на командировки в целом по организации (10%).

|  |  |
| --- | --- |
|  |  |

Расходы по статье «Прочие затраты» () включают затраты на приобретение и подготовку специальной научно-технической информации и специальной литературы. Определяется по формуле:

|  |  |
| --- | --- |
|  | (6.13) |

где – норматив прочих затрат в целом по организации.

|  |  |
| --- | --- |
|  |  |

Затраты по статье «Накладные расходы» () связаны с необходимостью содержания аппарата управления, вспомогательных хозяйств и опытных производств.

Затраты по статье «Накладные расходы» Определяются по формуле:

|  |  |
| --- | --- |
|  | (6.14) |

где – норматив накладных расходов в целом по организации.

Общая сумма расходов по смете ():

(руб).

Затраты на сопровождение и адаптацию ПО ():

|  |  |
| --- | --- |
|  | (6.15) |

где – норматив расходов на сопровождение (%).

Затраты на производство и продвижение программы определяется по формуле:

|  |  |
| --- | --- |
| . | (6.16) |

Подставляя значения в формулу 6.16, получим:

(руб).

Прибыль от создаваемого ПО определяется по формуле:

|  |  |
| --- | --- |
|  | (6.17) |

где – уровень рентабельности ПО (%).

Прогнозируемая цена ПО без налогов () определяется по формуле:

|  |  |
| --- | --- |
|  | (6.18) |

Подставляя значения в формулу 6.18, получим:

(руб).

Таблица 6.4 – Результаты и формулы расчетов.

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование статей | Усл. обозн. | Значение (руб) | Методика расчета |
| 1 | 2 | 3 | 4 |
| Основная заработная плата исполнителей | *З*о |  | Определяются на основании расчетов |
| Дополнительная заработная плата исполнителей | *З*д |  |  |
| Отчисления в фонд социальной защиты населения | *Р*соц |  |  |
| Машинное время | *Р*м |  | Определяются на основании расчета. Цена 1 машино-часа рыночная |
| Расходы на материалы | *M*i |  |  |
| Расходы на научные командировки | *Р*нк |  |  |
| Прочие прямые расходы | *П*з |  |  |
| Накладные расходы | *Р*н |  |  |
| Полная себестоимость | *С*п |  | *С*п = *Р*м + *З*о + *З*д + *Р*соц + *Р*м + + *Р*нп + *Р*пр + *Р*н |
| Затраты сопровождения и адаптации | *Р*с |  |  |
| Затраты на производство и продвижение | *С*пп |  |  |
| Затраты на производство и продвижение | *С*пп |  |  |
| Прогнозируемая прибыль | *П*о |  |  |
| Прогнозируемая цена без налогов (цена предприятия) | *Ц*п |  |  |
| Налог на добавленную стоимость (НДС) | НДС |  |  |
| Прогнозируемая отпускная цена | *Ц*о |  |  |
| Месячная тарифная ставка 1-го разряда | *Т*м1 |  |  |
| Ставка налога на прибыль | *Н*п | Нп=18% |  |

Налог на добавленную стоимость ():

|  |  |
| --- | --- |
|  | (6.19) |

где – норматив НДС (%).

Прогнозируемая отпускная цена ():

|  |  |
| --- | --- |
| . | (6.20) |

Подставляя значения в формулу 6.20, получим:

(руб).

Прибыль за вычетом налога рассчитывается по формуле:

|  |  |
| --- | --- |
|  | (6.21) |

где *Н*п – ставка налога на прибыль (*Н*п = 18%).

Подставляя значения в формулу 6.21, получим:

(руб).

Данная сумма остается при реализации собственнику разработки и представляет собой экономический эффект у собственника.

Все расчеты себестоимости и прибыли представлены в таблице 6.4.

Таким образом, в результате оценки экономического эффекта у разработчика были получены следующие результаты:

* себестоимость проекта составила 45,9 миллиона рублей;
* прогнозируемая отпускная цена – 75,1 миллиона рублей;
* прибыль за вычетом налога составит 11,8 миллиона рублей.

## 6.3 Расчет экономического эффекта от применения программного средства у пользователя

Результатом использования программного продукта в своей сфере является прирост чистой прибыли и амортизационных отчислений.

Прирост чистой прибыли представляет собой экономию затрат на заработную плату и начислений на заработную плату, материальных затрат.

Прирост прибыли за счет экономии расходов на заработную плату, связанный с высвобождением работников с повременной оплатой труда, определяется по формуле:

|  |  |
| --- | --- |
|  | (6.22) |

Результатом использования программного продукта в своей сфере является прирост чистой прибыли и амортизационных отчислений.

Прирост чистой прибыли представляет собой экономию затрат на заработную плату и начислений на заработную плату, материальных затрат.

Прирост прибыли за счет экономии расходов на заработную плату,  
связанный с высвобождением работников с повременной оплатой труда, определяется по формуле:

|  |  |
| --- | --- |
|  | (6.22) |

где *К*пр - коэффициент премий за выполнение плановых заданий (1,2);

∆*Ч*i – абсолютное высвобождение работников i-й категории, человек;

*3*i - годовая заработная плата высвобождаемых работников i-й кате­гории, рублей;

*Н*д - процент дополнительной заработной платы (18%);

*Н*но - ставка отчислений от заработной платы, включаемых в себестоимость продукции (0,6%);

*n* - категории высвобождаемых работников.

Таким образом, полученный прирост прибыли за счет экономии расходов на заработную плату равен:

(руб).

Прирост чистой прибыли рассчитывается по формуле:

|  |  |
| --- | --- |
|  | (6.23) |

где *n* – виды затрат, по которым получена экономия;

*Э*i – сумма экономии, полученная за счет снижения i-го вида затрат, рублей;

*Н*п - ставка налога на прибыль (18%).

Следовательно,

(руб).

Прирост чистой прибыли по годам эксплуатации имеет одинаковое значение.

Амортизационные отчисления являются источником погашения инвестиций в приобретение программного продукта. Расчет амортизационных отчислений осуществляется по формуле:

|  |  |
| --- | --- |
|  | (6.24) |

где *Н*а - норма амортизации программного продукта (15%);

*И*пс – стоимость программного продукта, рублей.

Таким образом, амортизационные отчисления составят:

*(руб).*

При оценке эффективности инвестиционных проектов необходимо осуществить приведение затрат и результатов, полученных в разные периоды времени, к расчетному году, путем умножения затрат и результатов на коэффициент дисконтирования , который определяется следующим образом:

|  |  |
| --- | --- |
|  | (6.25) |

где *Е*н - требуемая норма дисконта;

*t* - порядковый номер года, затраты и результаты которого приводятся к расчетному году;

*t*р - расчетный год, в качестве расчетного года принимается год вложения инвестиций, *t*p= 1.

Следовательно,

|  |  |
| --- | --- |
|  |  |

Расчет чистого дисконтированного дохода за четыре года реализации проекта и срока окупаемости инвестиций представлены в таблице 6.5. Нормативная ставка дисконта составляет Е = 32%.

Таблица 6.5 – Экономические результаты работы предприятия

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Наименование показателей | Един. измер. | По годам производства | | | |
| 2015 | 2016 | 2017 | 2018 |
| Результат |  |  |  |  |  |
| Прирост чистой прибыли | рубль | - | 58 414 380 | 58 414 380 | 58 414 380 |
| Прирост амортизационных отчислений | рубль | - |  |  |  |
| Прирост результата | рубль | - | 49 025 949 | 49 025 949 | 49 025 949 |
| Коэффициент дисконтирования |  | 1 | 0,76 | 0,57 | 0,44 |
| Результат с учетом фактора времени | рубль | - | 37 259 721 | 27 944 791 | 21 571 418 |
| Затраты (инвестиции) |  |  |  |  |  |
| Инвестиции в приобретение программного продукта | рубль |  | - | - | - |
| Инвестиции с учетом фактора времени | рубль |  | - | - | - |
| Чистый дисконтированный доход по годам | рубль |  | 37 259 721 | 27 944 791 | 21 571 418 |
| Чистый дисконтированный доход нарастающим итогом | рубль |  | -25 329 821 | 2 614 970 | 24 186 388 |

В качестве результата выступает чистая прибыль, остающаяся в распоряжении предприятия.

Рассчитаем рентабельность инвестиций (РИ) по формуле:

|  |  |
| --- | --- |
|  | (6.26) |

где *П*чср - среднегодовая величина чистой прибыли за расчетный период в рублях, которая определяется по формуле:

|  |  |
| --- | --- |
|  | (6.27) |

где *П*чi - чистая прибыль, полученная в году i в рублях.

С учетом того, что

*П*чср = (58 414 380 + 58 414 380 + 58 414 380) / 4 = 43 810 785 (руб).

Следовательно, рентабельность инвестиций будет *Р*и = 0,95.

В результате технико-экономического обоснования инвестиций по производству нового изделия были получены следующие значения показателей их эффективности:

* Чистый дисконтированный доход нарастающим итогом в 2018-м году составит 24,1 миллиона рублей;
* Инвестиции окупаются на третий год;
* Рентабельность инвестиций составляет 95%.

Таким образом, разработка и применение программного продукта является эффективным.

# 7 Эргономическая экспертиза клиент-серверного приложения управления скидками

Эргономика (или человеческий фактор) – это научная дисциплина, занимающаяся изучением взаимодействия между людьми и другими элементами систем, и профессия, которая использует теорию, законы, данные и методы конструирования в целях обеспечения здоровья человека и оптимизации общего функционирования системы [17].

Эргономика является системно-ориентированной дисциплиной, которая в настоящий момент охватывает все аспекты человеческой деятельности. Эргономика развивает целостный подход, сочетающий рассмотрение и учет физических, когнитивных (мыслительных), социальных, организационных и других значимых факторов. Специалисты, практикующий эргономику должен иметь широкую эрудицию во всех этих сферах. Данные специалисты часто работают в конкретных секторах или предметных областях, которые постоянно эволюционируют – создаются новые, а старые получают новые перспективы развития.

Внутри эргономики имеются направления, которые более глубоко изучают специфические особенности человека и характеристики его взаимодействия.

Эргономика в силу своего комплексного характера предполагает интеграцию достижений различных наук о человеке, прежде всего психологии труда и инженерной психологии, границы которых представляются сегодня в некоторой степени размытыми. Качественно новый уровень эргономических исследований может быть достигнут только в условиях определенности предметного содержания и связей, составляющих эргономику. В первую очередь это касается психологических дисциплин. Психология труда для эргономических исследований может рассматриваться как базовая дисциплина, обеспечивающая «конфигурации» деятельности в конкретных видах труда: «Что делает, почему, как и зачем?» – что позволяет решать задачи оптимизации условий, совершенствования технических средств и организации деятельности. Что касается инженерно-психологических исследований, то они приобретают все более выраженные особенности эргономических разработок, а исследования закономерностей психических процессов (например, связанных с пониманием и принятием решения), а также способ повысить эффективность решения сложных задач с использованием технических средств, квалифицируются как инженерно-психологические. Сегодня важно вернуться к вопросу о соотношении инженерной психологии и эргономики, имеющих о6щие задачи, в частности проектирования технических средств [18].

Человеко-компьютерное взаимодействие, о котором написано выше, – это изучение, планирование и разработка взаимодействия между людьми (пользователями) и компьютерами. Зачастую его рассматривают как совокупность науки о компьютерах, бихевиоризма, проектирования и других областей исследования. Взаимодействие между пользователями и компьютерами происходит на уровне пользовательского интерфейса (или просто интерфейса), который включает в себя программное и аппаратное обеспечение. Например, образы или объекты, отображаемые на экранах дисплеев, данные, полученные от пользователя посредством аппаратных устройств ввода (таких как клавиатуры и мыши) и другие взаимодействия пользователя с крупными автоматизированными системами, такими как воздушное судно и электростанция [19].

Основной задачей человеко-компьютерного взаимодействия является улучшение взаимодействия между человеком и компьютером, делая компьютеры более удобными и восприимчивыми к потребностям пользователей. В частности, человеко-компьютерное взаимодействие занимается:

1. методологией и развитием проектирования интерфейсов (т. е., исходя из требований и класса пользователей, проектирование наилучшего интерфейса в заданных рамках, оптимизация под требуемые свойства, такие как обучаемость и эффективность использования);
2. методами реализации интерфейсов (например, программные инструментарии, библиотеки и рациональные алгоритмы);
3. методами для оценки и сравнения таких интерфейсов;
4. разработкой новых интерфейсов и методов взаимодействия;
5. развитием описательных и прогнозируемых моделей;
6. теорией взаимодействия.

Долгосрочной задачей человеко-компьютерного взаимодействия является разработка системы, которая снизит барьер между человеческой когнитивной моделью того, чего они хотят достичь, и пониманием компьютером поставленных перед ним задач.

Создание качественного человеко-компьютерного интерфейса, который можно назвать точкой связи между человеком и компьютером, есть конечная цель изучения человеко-компьютерного взаимодействия.

Обмен информацией между человеком и компьютером можно определить как узел взаимодействия. Узел взаимодействия включает в себя несколько аспектов:

1. область задач: условия и цели, ориентированные на пользователя;
2. область машины: среда с которой взаимодействует компьютер;
3. области интерфейса: непересекающиеся области, касающиеся процессов человека и компьютера, не относящиеся к сфере взаимодействия;
4. входящий поток: поток информации, который начинается в области задач, когда пользователь имеет несколько задач, которые требуют использования компьютера;
5. выходной поток: поток информации, который возникает в машине;
6. обратная связь: узлы взаимодействия, проходящие через интерфейс, оцениваются, модерируются и подтверждаются, так как они проходят от человека через интерфейс к компьютеру и обратно.

При оценке текущего пользовательского интерфейса или разработке нового интерфейса следует иметь в виду следующие принципы разработки:

С самого начала необходимо акцентировать своё внимание на пользователях и задачах: установить количество пользователей, требуемых для выполнения задачи и определить подходящих пользователей; кто-либо никогда не использовавший интерфейс, либо тот, кто никогда не будет его использовать в будущем является неподходящим пользователем. Кроме того, необходимо определить какие задачи и как часто будут выполнять пользователи, а в данном случае – это администраторы приложения.

Эмпирические измерения: на ранней стадии провести тест интерфейса с реальными пользователями, которые используют интерфейс каждый день. Имейте в виду, что результаты могут измениться, если уровень производительности пользователя не является точным отображением реального человеко-компьютерного взаимодействия. Установить количественные особенности практичности, такие как: количество пользователей, выполняющих задачи, время выполнения задачи, и количество ошибок, сделанных в ходе выполнения задачи.

Итеративное проектирование: после определения количества пользователей, поставленных задач, эмпирических измерений, выполните следующие шаги итеративной разработки:

1. разработайте пользовательский интерфейс,
2. проведите тестирование,
3. проанализируйте результаты,
4. повторите предыдущие шаги.

Повторяйте итеративную разработку до тех пор, пока не создадите практичный, удобный для пользователя интерфейс.

Человеко-машинный интерфейс (ЧМИ) (англ. Human machine interface, HMI) – широкое понятие, охватывающее инженерные решения, обеспечивающие взаимодействие человека-оператора с управляемыми им машинами [20].

Проектирование ЧМИ включает в себя:

1. создание рабочего места: кресла, стола, или пульта управления, размещение приборов и органов управления (устройства ввода данных) (соответствием всего этого физиологии человека занимается эргономика), освещение рабочего места и, возможно, микроклимат;
2. далее рассматриваются взаимодействие оператора со всеми органами управления: их доступность и необходимые усилия, эффективность и скорость доступа, согласованность (непротиворечивость) управляющих воздействий, расположение дисплеев и размеры надписей на них.

Достаточно сложной задачей является создание эффективного ЧМИ рабочих мест систем управления данными предприятия, одной из которых и является веб приложение для управления скидками, предназначенное для администраторов.

В промышленных условиях ЧМИ чаще всего реализуется с использованием типовых средств: компьютеров и типового программного обеспечения.

Информационная совместимость обеспечивается унификацией входных и выходных сигналов и применением стандартных интерфейсов. Интерфейс представляет собой систему унифицированных связей и сигналов - конструктивных, логических, физических, посредством которых технические средства соединяются друг с другом и производят обмен информацией.

Возникновение наиболее распространенных форм дискомфорта, испытываемых пользователями компьютеров, часто может быть связано с неправильным расположением экрана компьютера и устройств ввода (таких как клавиатуры и мыши).

Часто дискомфорт возникает, когда эти устройства расположены слишком высоко или слишком низко, либо слишком далеко или слишком близко к пользователю. Слишком высокое или слишком низкое расположение монитора заставляет пользователя вытягивать или наклонять голову и шею; нахождение в этих неудобных позах дольше определенного небольшого промежутка времени быстро вызывает дискомфорт [21].

В то же время слишком высоко расположенный экран компьютера обычно подвергает воздействию большую поверхность глаза, что приводит к более быстрому высыханию защитного слезного слоя, чем в других случаях. С ноутбуками, составляющими в настоящее время более половины объема всех компьютерных продаж (имеется в виду Европа и США), могут возникнуть особые проблемы в отношении высоты расположения из-за фиксированной высоты экрана относительно клавиатуры.

Основной принцип эргономики заключается в том, что работа должна быть удобной для человека, а не заставлять его приспосабливаться к своей работе. Регулируемые крепления позволяют пользователю подстраивать рабочее место под свои индивидуальные особенности, обеспечивая максимальный комфорт и производительность.

Например, для женщин, расстояние от глаза до локтя которых составляет 48, логично предположить, что если экран портативного компьютера имеет диагональ 15 дюймов, а пропорции изображения на экране составляют 4:3, и расстояние обзора составляет не менее 50 см, то угол обзора – 36,3°. Но поскольку идеальный диапазон обзора составляет от 15° до 30° ниже горизонтали, то экран ноутбука находится ниже предела допустимого диапазона.

Визуальные эргономические параметры ПК являются параметрами безопасности, и их неправильный выбор зачастую приводит к ухудшению здоровья пользователей. Опасности могут быть связаны с обострением хронических заболеваний глаз, проявлением наследственных предрасположенностей. Поэтому так важен режим работы с ПК, профилактические мероприятия, и самое главное – эргономические параметры видеомониторов.

Одним из основных параметров является частота вертикальной или кадровой развертки (частота обновления), которая (что признается подавляющим числом международных и национальных стандартов) должна быть не менее 85 Гц, желательно в режиме максимального разрешения. Особенно это важно при работе с графическими пакетами. В практической эргономике для определения усталостных характеристик человека-оператора применяется такой психофизиологический показатель, как критическая частота слияния мельканий (КЧСМ). КЧСМ зависит от яркости изображения, спектра излучения, местоположения изображения на сетчатке глаза, размеров наблюдаемого объекта, от возраста оператора и ряда других факторов, в том числе от времени работы человека с информационной моделью, вызывающей усталость. Но в любом случае она не может превысить 30-35 Гц в центральной области зрения. Этими цифрами, определяется нижняя граница допустимой частоты вертикальной развертки монитора. Особенно высокой чувствительностью к изменению яркости изображения обладают окраинные области сетчатки.

Для обеспечения надежного считывания информации при соответствующей степени комфортности ее восприятия должны быть определены оптимальные диапазоны визуальных эргономических параметров.

Прямой яркостной контраст для позитивного изображения на мониторах определяется по формуле:

,

где L – яркость элемента;

L⋅Ф – яркость фона.

Минимальная яркость экрана должна быть не менее , рекомендуемая – . Под неравномерностью яркости понимается положительная и отрицательная неравномерность:

,

где ;

n – число измеренных значений яркости;

– максимальное значение яркости;

– минимальное значение яркости.

Угловой размер знака – это угол между линиями, соединяющими крайние точки знака по высоте и глаз наблюдателя.

Угловой размер знака определяется из формулы:

то есть .

Угловые размеры знаков в зависимости от их сложности для восприятия должны быть регламентированы следующими диапазонами:

1. знак простой формы (2-3 элемента);
2. знак средней сложности (4-6 элементов);
3. знак более сложной формы (7-9 элементов).

Способы регулировок параметров мониторов могут быть различными, что несущественно, так как они производятся значительно реже одного раза в рабочую смену, а это по эргономическим меркам – редко используемые органы управления. Конструктивно они могут быть выполнены в виде ручных регулирующих органов управления или как экранное меню с соответствующим назначением. При установке ручных органов управления, естественно, следует стремиться к сокращению их номенклатуры. При необходимости расположения органов управления на лицевой панели они должны закрываться крышкой или быть утоплены в корпус. В случае экранного меню возможно бесконечно большое число регулировок в зависимости от уровня подготовленности пользователя (который, в свою очередь, тоже может быть регламентирован в виде меню) или специфики работы, единственное условие – оптимальное формирование информационной модели. В настоящее время самым распространенным с небольшими вариациями является вывод пункта настройки и кнопок плюс/минус для увеличения и уменьшения параметров.

Необходимо рассмотреть эргономические показатели не только оборудования при использовании программного средства, но и интерфейс самого программного средства. Диапазон критериев оценки интерфейсов весьма широк: время изучения, скорость работы (продуктивность), частота ошибок, простота исполнения и др. Для оценки интерфейсов используется множество методов, качественных и количественных, базирующихся на экспериментах, пассивном наблюдении и пр.

В работе рассматривается один из лучших и признанных подходов к количественной оценке эффективности интерфейсов по критерию скорости работы – классическая модель GOMS (the model of goals, objects, methods and selection rules, правила для целей, объектов, методов и выделения). Моделирование GOMS позволяет предсказать, сколько времени потребуется опытному пользователю на выполнение конкретной операции при использовании некоторой модели интерфейса, и сделать сравнительную оценку различных интерфейсов по эффективности их использования [22].

Основной целью работы является освоение модели GOMS и ее использование для сравнительного анализа интерфейсов. При этом предполагается сравнение как интерфейсов различных приложений при выполнении некоторой содержательной операции, так и вариантов интерфейса одного приложения при выполнении операции несколькими способами.

Концепции «объект-действие» и «действие-объект» определяют приоритет объекта или действия при интерактивном взаимодействии пользователя с интерфейсом. Первая концепция является более прогрессивной и предполагает ориентацию сначала на объекты деятельности, а затем на выбор действий с ними, вторая – наоборот (аналогично объектно-ориентированной и процедурной парадигмам программирования). Использование первой концепции повышает эффективность интерфейса.

В работе предлагается параллельно сделать оценку интерфейса по модели GOMS и выявить его характеристики с точки зрения упомянутых концепций. Модель предназначена для оценки времени выполнения конкретной операции при работе с конкретной моделью интерфейса. Используется набор установленных экспериментально средних интервалов времени выполнения стандартных операций.

Сначала записывается последовательность производимых пользователем действий с устройствами ввода и элементами интерфейса. Затем в эту последовательность добавляются ментальные операции, т.е. умственные операции по подготовке к следующему действию. На следующих шагах исключаются ментальные операции там, где последовательные действия не требуют времени на размышление. И, наконец, согласно полученной схеме подсчитывается суммарное время работы.

Таблица 7.1 – Временные интервалы в интерфейсе

|  |  |  |
| --- | --- | --- |
| Операция, интервал | Название | Смысл |
| K = 0,2 – 0,28 с | Нажатие  клавиши | Время, необходимое для нажатия клавиши, включая Alt, Ctrl и Shift. |
| B = 0,1 с | Нажатие кнопки | Время, необходимое для нажатия на кнопку мыши. |
| P = 1,1 с | Указание | Время, необходимое для указания на какую-то позицию на экране монитора. Согласно закону Фитса, время, затрачиваемое на перемещение курсора, зависит как от дистанции, так и от размера цели. Тем не менее, это число представляет достаточно точный компромисс. |
| H = 0,4 с | Перемещение | Время, необходимое для перемещения руки с клавиатуры на ГУВ или с ГУВ на клавиатуру. Фактически взятие или бросание мыши. |
| M = 1,2 – 1,35 с | Ментальная подготовка | Время, необходимое для умственной подготовки к следующему шагу, т.е. принятие решения о действии на следующем шаге. Обычно это самый сложный оператор, поскольку часто непонятно, в каких именно местах процедуры его необходимо ставить. |
| R – от 0,1 с | Ответ | Время ожидания ответа компьютера. Для базовых операций, таких как работа с меню, это время можно не засчитывать. |

Таблица 7.2 – Правила расстановки ментальных операций

|  |  |
| --- | --- |
| Правило | Описание |
| 1 | 2 |
| Правило 0.  Начальная  расстановка  операторов M | Операторы M следует устанавливать:  перед всеми операторами K;  перед всеми операторами P, предназначенными для выбора команд.  Перед операторами P, предназначенными для указания на аргументы выбранных команд, ставить оператор M не следует. |

Продолжение таблицы 7.2

|  |  |
| --- | --- |
| 1 | 2 |
| Правило 1.  Удаление  ожидаемых  операторов M | Из последовательности «Оп1 M Оп2» оператор M может быть удален, если оператор Оп2 является полностью ожидаемым с точки зрения оператора Оп1. Например, курсор мыши перемещается к нужному объекту с целью нажатия кнопки по его достижении (последовательность P K). Согласно правилу 0 эта последовательность преобразуется в P M K, затем, согласно данному правилу, в P K. |
| Правило 2.  Удаление  операторов M  внутри  когнитивных единиц | Когнитивной единицей является непрерывная последовательность вводимых символов, которые образовывают название команды или аргумент (например, имена файлов при сохранении, значения временных интервалов как параметры автосохранения и т.д.).  Если строка вида M K M K M K принадлежит когнитивной единице, то следует удалить все операторы M, кроме первого. |
| Правило 3.  Удаление M перед после-довательны-ми раздели-телями | Если оператор K означает лишний разделитель, стоящий в конце когнитивной единицы, то следует удалить находящийся перед ним оператор M. |
| Правило 4.  Удаление  операторов M,  которые являются  прерывателями  команд | Если оператор K является разделителем, стоящим после постоянной строки, то стоящий перед ним оператор M удаляется.  Постоянными строками являются, например, имена команд или ответы Y/N на запрос. Добавление пробела перед аргументами команды или нажатие клавиши Enter при ответе в этих случаях в силу привычности не требует ментального оператора. |
| Правило 5.  Удаление  Перекрывающих опера- торов M | Не следует учитывать любую часть оператора M, перекрывающую оператор R. Так, последовательность R M или M R означает, что ментальная операция может быть совмещена с ожиданием ответа компьютера. Если время ожидания превышает время ментальной операции, то последняя не учитывается полностью. |

Проведем анализ с помощью модели GOMS наиболее часто используемого администратором компонента системы – компонента для создания и редактирования скидок. Интерфейс для обрезания скидок из листов скидок магазинов представлен на рисунке 7.1. Интерфейс для сохранения скидки с кнопками непосредственно сохранения и отмены, а также полями для ввода названия продукта изображен на рисунке 7.2.

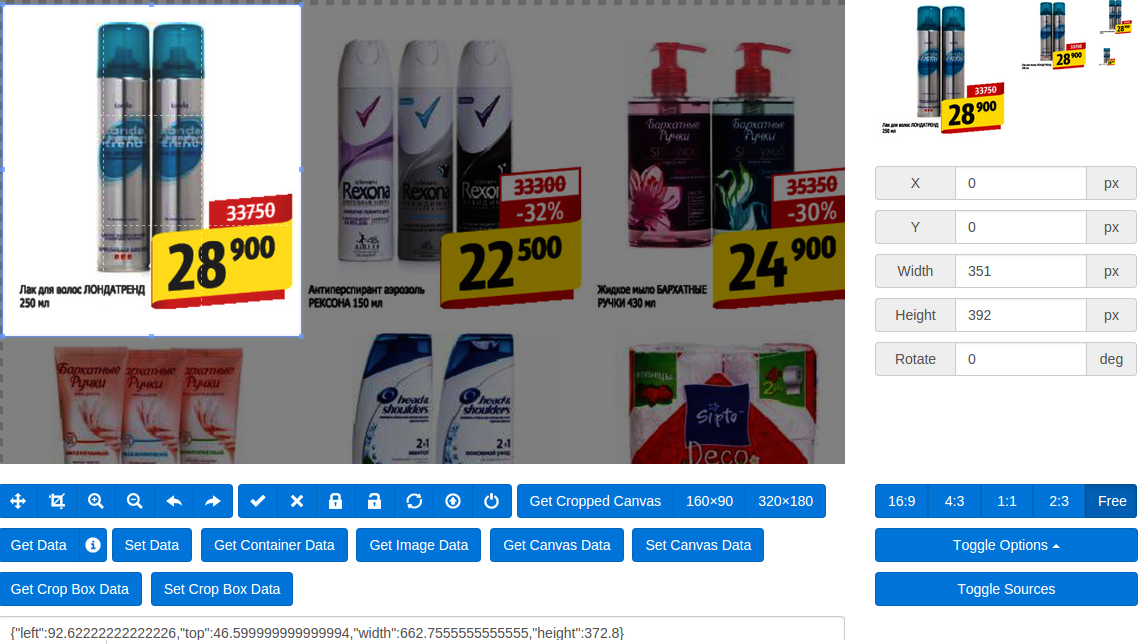


Рисунок 7.1 – Интерфейс для обрезания изображения скидки

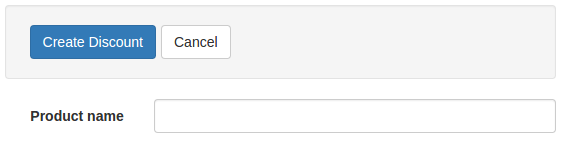


Рисунок 7.2 – Интерфейс сохранения скидки

В таблице 7.3 описаны очередные действия пользователя по порядку, а также формируемую в результате последовательность операций в соответствие с правилами и терминами таблиц 7.1 и 7.2.

В таблице 7.4 описаны действия по построению результирующей функции правил, а также подсчет итогового времени, необходимого на создание скидки на товар или услугу.

Таблица 7.3 - Действия пользователя

|  |  |
| --- | --- |
| Очередное действие пользователя | Формируемая последовательность операций |
| Перемещение руки к мыши | H |
| Перемещение курсора к изображению скидки | H P |
| Нажатие на кнопку мыши | H P K |
| Выделение изображения | H P K Р |
| Перемещение курсора к полю ввода названия продукта | H P K Р Р |
| Нажатие на кнопку мыши | H P K Р Р К |
| Перемещение руки к клавиатуре | H P K Р Р К Н |
| Ввод символов (в среднем 6) | H P K Р Р К Н К К К К К К |
| Нажатие клавиши <Enter> | H P K Р Р К Н К К К К К К К |

Таблица 7.4 - Расстановка ментальных операций

|  |  |
| --- | --- |
| Действие | Результирующая функция |
| Начальная расстановка операторов M согласно табл. 1 (аргументов команд в данном случае нет). | H М P М K М Р М Р М К Н М К М К М К М К М К М К М К |
| Удаление ожидаемых операторов M по правилу 1 (P M K преобразуется в P K). | H М P K М Р М Р К Н М К М К М К М К М К М К М К |
| Удаление операторов M внутри когнитивных единиц. Перед конечным K оператор M остается. | H М P K М Р М Р К Н М К К К К К К М К |
| Замена символов операторов соответствующими интервалами и подсчет общего времени работы. | H+ M+ ... = 0,4+1,35+ ... = 12,76 с |

В результате проведенной работы по эргономической экспертизе клиент-серверного приложения управления скидками изучены и проанализированы эргономические характеристики рабочего места администратора, который будет использовать данное приложение. Результаты показали, что, при использовании рекомендуемых параметров оборудования и рабочего места, работа с данным приложением является комфортной для администратора. Также с использованием методологии GOMS проанализированы наиболее часто используемые администратором функции приложения. В результате время редактирования и создания объявления получилось равным 12,76 секунд, что вполне соответствует реальному времени, полученному при тестировании программного средства. Результаты исследований показали, что модель GOMS применима для оценки пользовательских интерфейсов приложения для работы со скидками. Результаты методологии GOMS дают разработчикам подобных средств важную информацию для поиска новых идей в интерфейсах разрабатываемых приложений.

# Заключение

В ходе создания дипломного проекта изучена и проанализирована техническая литература по теме дипломного проекта, а также основные принципы и методы, используемые при разработке клиент-серверных приложений. Это включает в себя анализ особенностей использования баз данных в веб приложениях, сравнение наиболее распространенных типов баз данных, включая реляционные и нереляционные, изучение особенностей применения систем управления базами данных. Проанализирован и обоснован выбор технологий разработки, к чему относится выбор архитектуры и программной платформы разработки, выбор системы управления базами данных (сравнены наиболее широко используемые аналоги), выбор языка программирования, причем как в серверном приложении, так и в мобильном клиенте, а также выбор операционных систем для использования клиента и сервера приложения.

Проанализированы и сравнены существующие аналоги программного средства, выявлены их основные особенности, преимущества и недостатки. На основании последних сделаны выводы и сформулированы пути и методы их устранения. При моделировании предметной области разработаны функциональная и информационная модели, представляющие собой диаграмму вариантов использования, являющуюся частью унифицированного языка моделирования. Разработана спецификация требований к программному средству.

Выполнено проектирование клиент-серверного приложения. Разработана архитектура приложения, которая представлена в виде диаграммы компонентов, таких как используемые для работы приложения сторонние библиотеки, включенные в состав платформы разработки, а также контроллеры, модели и представления, которые являются основными составляющими приложения при использовании шаблона проектирования MVC. Разработаны схемы алгоритмов, которые описывают наиболее значимые функциональные возможности программного средства, а именно: схема работы программы (описан весь цикл работы как клиента, так и сервера, начиная с действий пользователя мобильного приложения и заканчивая отправкой данных с сервера и их отображением на клиенте), схема работы администратора (описаны возможные действия администратора приложения) а также схема добавления скидок, которая описывает наиболее важную часть работы серверного приложения и администратора.

Все части клиент-серверного приложения (мобильное приложение под Android и веб-приложение) протестированы, результаты чего задокументированы. Это подтверждает, что клиент-серверное приложение управления скидками на товары и услуги функционирует в полном соответствии со спецификацией требований. В рамках создания документации к проекту также разработана методика использования мобильного и веб приложения.

Выполнено технико-экономическое обоснование разработки и использования программного продукта. Проведена эргономическая экспертиза приложения управления скидками.

Результатом проделанной работы стало программное средство, по многим характеристикам превосходящее аналоги. К данным характеристикам можно отнести современную администраторскую панель, позволяющую в значительной степени автоматизировать работу, а также удобное мобильное приложение, которое позволит просматривать скидки не только дома, но и на работе, в дороге или непосредственно в магазине во время совершения покупок.

Таким образом, цель дипломного проектирования достигнута в полном соответствии с заданными целями и задачами.

# Список использованных источников

[1] Гурвиц, Г. Разработка реального приложения в среде клиент-сервер / Гурвиц. Г. – М. : ДВГУПС, 2005. – 259 с.

[2] REST [Электронный ресурс]. - Электронные данные – Режим доступа: <https://ru.wikipedia.org/wiki/REST>

[3] Зрюмов, Е.А., Базы данных для инженеров / Е.А. Зрюмов, А.Г. Зрюмова – Барнаул : АлтГТУ – 2010. – 131 с.

[4] gazetki.by [Электронный ресурс]. - Электронные данные – Режим доступа: <http://gazetki.by/>

[5] skidki.by [Электронный ресурс]. - Электронные данные – Режим доступа: <http://skidki.by/main.asp>

[6] minskskidky.by [Электронный ресурс]. - Электронные данные – Режим доступа: <http://minskidki.by/>

[7] shoppingplus [Электронный ресурс]. - Электронные данные – Режим доступа: <http://www.shoppingplus.ru/>

[8] Кросинг, Х. Администрирование PostgreSQL 9. Книга рецептов / . Х. Кросинг, С. Ригс – М : ДМК Пресс, 2013. – 328 с.

[9] SQLite about [Электронный ресурс] – Электронные данные – Режим доступа: https://www.sqlite.org/about.html

[10] Ruby-lang [Электронный ресурс] – Электронные данные – Режим доступа: <https://www.ruby-lang.org/ru/about/>

[11] 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 )

[12] Рексфорд, Дж. Web-протоколы. Теория и практика. HTTP/1.1, взаимодействие протоколов, кэширование, измерение трафика / Дж. Рексфорд, Б. Кришнамурти – СПб. : Бином, 2009. – 351 с.

[13] Java About [Электронный ресурс] – Электронные данные – Режим доступа: <https://www.java.com/ru/about/>

[14] Sparx Systems [Электронный ресурс] – Электронные данные – Режим доступа: <http://www.sparxsystems.com.au/about.html>

[15] Ларман, К. Применение UML 2.0 и шаблонов проектирования. 3-е изд. / К. Ларман – М. : ДВГУПС, 2013. – 143 с.

[16] RailRoady [Электронный ресурс] – Электронные данные – Режим доступа: <https://github.com/preston/railroady>

[17] Сайт межрегиональной эргономической ассоциации [Электронный ресурс] – Электронные данные – Режим доступа: [http://www.ergo-org.ru/ergo.html](http://www.ergo-org.ru/ergo.html%20)

[18] Анохин, А.Н Психология труда, инженерная психология и эргономика : сборник докладов / А.Н. Анохин, П.И. Падерно, С.Ф. Сергеев – СПб. : Межрегиональная эргономическая ассоциация, 2014. – 452 с.

[19] Человеко-компьютерное взаимодействие [Электронный ресурс] – Электронные данные – Режим доступа: <https://ru.wikipedia.org/wiki/Человеко-компьютерное_взаимодействие>

[20] Человеко-машинный интерфейс [Электронный ресурс] – Электронные данные – Режим доступа: <https://ru.wikipedia.org/wiki/Человеко-машинный_интерфейс>

[21] Ergotron [Электронный ресурс] – Электронные данные – Режим доступа: <http://www.ergotron-russia.ru/ergonomics/ergonomica_laptop1.html>

[22] Raskin interface [Электронный ресурс] – Электронные данные – Режим доступа: <http://raskin-interface.narod.ru/interface/chapter4.htm>

# Приложение А

**(обязательное)**

## Текст программы

class Admin::ChainsController < ApplicationController

inherit\_resources

def update\_sheet

resource.update\_sheet

redirect\_to admin\_chain\_sheets\_path(resource)

end

def create

create! { collection\_path }

end

def update

update! { collection\_path }

end

private

def permitted\_params

params.permit!

end

end

class Admin::DiscountsController < ApplicationController

inherit\_resources

belongs\_to :chain

def create

binding.pry

create! { collection\_path }

end

def update

binding.pry

update! { collection\_path }

end

private

def permitted\_params

params.permit!

end

class Admin::SettingsController < ApplicationController

inherit\_resources

def update

update! { admin\_root\_path }

end

private

def permitted\_params

params.permit!

end

end

end

class Admin::SheetsController < ApplicationController

inherit\_resources

belongs\_to :chain

def create

create! { collection\_path }

end

def update

update! { collection\_path }

end

private

def permitted\_params

params.permit!

end

end

class Admin::UsersController < InheritedResources::Base

end

class ApplicationController < ActionController::Base

# Prevent CSRF attacks by raising an exception.

# For APIs, you may want to use :null\_session instead.

before\_action :authenticate\_user!

protect\_from\_forgery with: :exception

end

module ChainsHelper

def chain\_actions(resource)

buttons = []

buttons << link\_to('Edit', edit\_resource\_path(resource), class: 'btn btn-default btn-sm')

buttons << link\_to('Delete', resource\_path(resource), method: :delete, class: 'btn btn-danger btn-sm')

buttons << link\_to('Discounts', admin\_chain\_discounts\_path(resource), class: 'btn btn-default btn-sm')

buttons << link\_to('Sheets', admin\_chain\_sheets\_path(resource), class: 'btn btn-default btn-sm')

buttons << link\_to('Update sheet', update\_sheet\_admin\_chain\_path(resource), method: :post, class: 'btn btn-default btn-sm')

buttons.join(' ').html\_safe

end

end

module DiscountsHelper

def discounts\_actions(resource)

buttons = []

buttons << link\_to('Edit', edit\_resource\_path(resource), class: 'btn btn-default btn-sm')

buttons << link\_to('Delete', resource\_path(resource), method: :delete, class: 'btn btn-danger btn-sm')

buttons.join(' ').html\_safe

end

end

module SheetsHelper

def sheets\_actions(resource)

buttons = []

buttons << link\_to('Edit', edit\_resource\_path(resource), class: 'btn btn-default btn-sm')

buttons << link\_to('Delete', resource\_path(resource), method: :delete, class: 'btn btn-danger btn-sm')

buttons.join(' ').html\_safe

end

def sheet\_images(sheet)

result = ''

sheet.images.each do |image|

result << image\_tag(image.attachment.url(:thumb))

end

result.html\_safe

end

end

# == Schema Information

#

# Table name: chains

#

# id :integer not null, primary key

# name :string not null

# created\_at :datetime not null

# updated\_at :datetime not null

#

class Chain < ActiveRecord::Base

has\_one :logo, as: :imageable, class\_name: Image, dependent: :destroy

has\_many :discounts, dependent: :destroy

has\_many :sheets, dependent: :destroy

has\_one :sheet\_setting, dependent: :destroy

validates :name, presence: true

accepts\_nested\_attributes\_for :logo, :sheet\_setting

def builded\_logo

logo || build\_logo

end

def builded\_sheet\_setting

sheet\_setting || build\_sheet\_setting

end

def update\_sheet

file = open(sheet\_setting.pdf\_url)

pdf = Grim.reap(file.path)

sheet = sheets.create

pdf.each do |page|

path = pdf.path + (index = (index || 0) + 1).to\_s + '.jpg'

page.save(path, density: Setting.instance.image\_density)

sheet.images << Image.new(attachment: File.open(path))

end

sheet.split\_into\_discounts

end

end

# == Schema Information

#

# Table name: discounts

#

# id :integer not null, primary key

# product\_name :string

# price :integer

# economy :float

# chain\_id :integer

# created\_at :datetime not null

# updated\_at :datetime not null

# sheet\_id :integer

# active\_from :datetime

# active\_till :datetime

#

# Indexes

#

# index\_discounts\_on\_active\_till\_and\_active\_from (active\_till,active\_from)

# index\_discounts\_on\_chain\_id (chain\_id)

# index\_discounts\_on\_sheet\_id (sheet\_id)

#

class Discount < ActiveRecord::Base

belongs\_to :chain

has\_one :image, as: :imageable, dependent: :destroy

accepts\_nested\_attributes\_for :image

def serializable\_hash(\*args)

host = 'http://www.discountswith.me/'

super(\*args).merge(original\_image: URI.join(host, image.attachment.url).to\_s,

thumb\_image: URI.join(host, image.attachment.url(:thumb)).to\_s)

end

def builded\_image

image || build\_image

end

end

# == Schema Information

#

# Table name: images

#

# id :integer not null, primary key

# imageable\_id :integer

# imageable\_type :string

# created\_at :datetime not null

# updated\_at :datetime not null

# attachment\_file\_name :string

# attachment\_content\_type :string

# attachment\_file\_size :integer

# attachment\_updated\_at :datetime

#

# Indexes

#

# index\_images\_on\_imageable\_type\_and\_imageable\_id (imageable\_type,imageable\_id)

#

class Image < ActiveRecord::Base

belongs\_to :imageable

has\_attached\_file :attachment, styles: { original: {}, medium: '300x300>', thumb: '100x100>' },

default\_url: '/images/:style/missing.png'

crop\_attached\_file :attachment

validates\_attachment\_content\_type :attachment, content\_type: /\Aimage\/.\*\Z/

validates :attachment, presence: true

def source\_url=(url)

self.attachment = URI.parse(url).to\_s

end

def source\_url;end

end

# == Schema Information

#

# Table name: settings

#

# id :integer not null, primary key

# fields :json

# created\_at :datetime not null

# updated\_at :datetime not null

#

class Setting < ActiveRecord::Base

store\_accessor :fields, :image\_density

validates :image\_density, numericality: { greater\_than: 0, less\_than: 1200,

only\_integer: true }

def self.instance

first

end

end

# == Schema Information

#

# Table name: sheets

#

# id :integer not null, primary key

# chain\_id :integer

# created\_at :datetime not null

# updated\_at :datetime not null

#

# Indexes

#

# index\_sheets\_on\_chain\_id (chain\_id)

#

class Sheet < ActiveRecord::Base

belongs\_to :chain

delegate :sheet\_setting, to: :chain

has\_many :images, as: :imageable, class\_name: Image

has\_many :discounts, dependent: :destroy

# def self.selector

# "a:contains('Скачать буклет в формате PDF')"

# end

# def self.page\_url

# 'http://sosedi.by/akcii/osobaya\_cena.php'

# end

# def self.get\_pdf\_url

# # doc = Nokogiri::HTML(open(page\_url).read)

# # doc.css(selector).first.attributes['href'].value

# 'http://sosedi.by//upload/iblock/c5b/c5b25b49a7eb5cb2e084702abb72dca1.pdf'

# end

# def self.create\_via\_grim(density)

# pdf.each do |page|

# path = @pdf.path + (index = (index || 0) + 1).to\_s + density.to\_s + '.jpg'

# page.save(path, density: density)

# Image.create!(attachment: File.open(path))

# end

# end

# def self.create\_images

# MiniMagick::Tool::Convert.new do |converter|

# converter.density 200

# converter << 'c5b25b49a7eb5cb2e084702abb72dca1.pdf'

# converter << 'foo-%02d.jpg'

# end

# # path\_to\_pdf = Sheet.first.attachment.path

# image\_from\_pdf = MiniMagick::Image.open(get\_pdf\_url)

# image\_from\_pdf.format 'png', nil

# image\_from\_pdf.pages.each\_with\_index do |page, index|

# file\_path = "/tmp/#{page.path}#{index}"

# page.write(file\_path)

# Image.create!(attachment: File.open(file\_path))

# end

# end

def split\_into\_discounts

image\_pathes = images.map(&:attachment).map(&:path)

magick\_images = image\_pathes.map { |path| MiniMagick::Image.open(path) }

magick\_images.each do |magick\_image|

tmp\_image\_path = crop\_image(magick\_image)

tile\_image(tmp\_image\_path)

tiles\_count = sheet\_setting.rows\_number \* sheet\_setting.columns\_number

tiles\_count.times do |index|

file = File.open(tiled\_image\_path(tmp\_image\_path, index))

discounts.create(chain\_id: chain\_id, image\_attributes: { attachment: file })

# File.delete(tmp\_image\_path)

end

# tile\_width = magick\_image.height / sheet\_setting.rows\_number

# tile\_height = magick\_image.width / sheet\_setting.columns\_number

#

# sheet\_setting.columns\_number.times do |column\_number|

# sheet\_setting.rows\_number.times do |row\_number|

# start\_column = column\_number \* tile\_width

# start\_row = row\_number \* tile\_height

# cropped = magick\_image.crop(start\_column, start\_row, tile\_width, tile\_height)

# cropped.write("#{index}\_#{column}x#{row}.jpg")

# end

# end

end

end

def tile\_image(image\_path)

MiniMagick::Tool::Convert.new do |converter|

converter << image\_path

converter.crop "#{sheet\_setting.columns\_number}x#{sheet\_setting.rows\_number}@"

converter << tiled\_image\_path(image\_path, '%d')

end

end

def tiled\_image\_path(image\_path, number)

"#{image\_path}#{number}.jpg"

end

def crop\_image(magick\_image)

right = sheet\_setting.margin\_right \* magick\_image.width / 100

left = sheet\_setting.margin\_left \* magick\_image.width / 100

top = sheet\_setting.margin\_top \* magick\_image.height / 100

bottom = sheet\_setting.margin\_bottom \* magick\_image.height / 100

result\_path = magick\_image.path + '.jpg'

MiniMagick::Tool::Convert.new do |converter|

converter << magick\_image.path

converter.crop "0x0-#{right}-#{bottom}"

converter.crop "0x0+#{left}+#{top}"

converter << result\_path

end

result\_path

end

end

# == Schema Information

#

# Table name: sheet\_settings

#

# id :integer not null, primary key

# page\_with\_link\_url :string

# download\_link\_selector :string

# rows\_number :integer

# columns\_number :integer

# margin\_left :integer

# margin\_top :integer

# margin\_right :integer

# margin\_bottom :integer

# type :string

# chain\_id :integer

# created\_at :datetime not null

# updated\_at :datetime not null

#

# Indexes

#

# index\_sheet\_settings\_on\_chain\_id (chain\_id)

#

class SheetSetting < ActiveRecord::Base

belongs\_to :chain

def pdf\_url

doc = Nokogiri::HTML(open(page\_with\_link\_url).read)

result = doc.css(download\_link\_selector).first.attributes['href'].value

uri = URI.parse(URI.escape(result))

return uri if uri.host.present?

uri = uri.to\_s

uri = uri[1..-1] if uri[0] == '.'

URI::HTTP.build host: URI(page\_with\_link\_url).host, path: uri

end

end

# == Schema Information

#

# Table name: users

#

# id :integer not null, primary key

# email :string default(""), not null

# encrypted\_password :string default(""), not null

# reset\_password\_token :string

# reset\_password\_sent\_at :datetime

# remember\_created\_at :datetime

# sign\_in\_count :integer default("0"), not null

# current\_sign\_in\_at :datetime

# last\_sign\_in\_at :datetime

# current\_sign\_in\_ip :inet

# last\_sign\_in\_ip :inet

# created\_at :datetime

# updated\_at :datetime

#

# Indexes

#

# index\_users\_on\_email (email) UNIQUE

# index\_users\_on\_reset\_password\_token (reset\_password\_token) UNIQUE

#

class User < ActiveRecord::Base

# Include default devise modules. Others available are:

# :confirmable, :lockable, :timeoutable and :omniauthable

devise :database\_authenticatable, :recoverable, :rememberable,

:trackable, :validatable

End

= simple\_form\_for [:admin, resource], html: { class: 'form-horizontal' },

wrapper: :horizontal\_form,

wrapper\_mappings: {\

file: :horizontal\_file\_input\

} do |f|

.well.well-sm.btn-toolbar

= f.button :submit, class: 'btn-primary'

= link\_to 'Cancel', collection\_path, class: 'btn btn-default'

= f.input :name

= f.simple\_fields\_for :logo, resource.builded\_logo do |logo\_form|

= logo\_form.input :attachment, as: :file, label: 'Logo file'

= image\_tag(logo\_form.object.attachment.url(:thumb)) if logo\_form.object.attachment.present?

= logo\_form.input :attachment, as: :url, label: 'or logo url'

.well.clearfix

h4 Sheet settings

= f.simple\_fields\_for :sheet\_setting, resource.builded\_sheet\_setting do |sheet\_setting\_form|

= sheet\_setting\_form.input :page\_with\_link\_url

= sheet\_setting\_form.input :download\_link\_selector

h5 Layout

.col-sm-6= sheet\_setting\_form.input :rows\_number

.col-sm-6= sheet\_setting\_form.input :columns\_number

h5 Margin

.col-sm-6= sheet\_setting\_form.input :margin\_left, label: 'left'

.col-sm-6= sheet\_setting\_form.input :margin\_right, label: 'right'

.col-sm-6= sheet\_setting\_form.input :margin\_top, label: 'top'

.col-sm-6= sheet\_setting\_form.input :margin\_bottom, label: 'bottom'

h3 Chain stores

= link\_to 'new', new\_resource\_path, class: 'btn btn-primary'

= table\_for collection, html: { class: 'table table-bordered' } do

- column :id

- column :name

- column :logo do |logo|

- image\_tag logo.try { |logo| logo.attachment.url(:thumb) }

- column :created\_at

- column :updated\_at

- column title: 'Actions' do |chain|

- chain\_actions(chain)

= simple\_form\_for [:admin, parent, resource], html: { class: 'form-horizontal', multipart: true },

wrapper: :horizontal\_form,

wrapper\_mappings: {\

file: :horizontal\_file\_input\

} do |f|

.well.btn-toolbar

= f.button :submit, class: 'btn-primary'

= link\_to 'Cancel', collection\_path, class: 'btn btn-default'

= f.input :product\_name

= f.simple\_fields\_for :image, resource.builded\_image do |image\_form|

- if f.object.persisted?

= image\_form.cropbox :attachment

= image\_form.crop\_preview :attachment

- else

= image\_form.input :attachment, as: :file, label: 'Image file'

= image\_form.input :source\_url, label: 'or image url'

= f.input :price

= f.input :economy, input\_html: { min: 0, max: 100 }

h3 Discounts

.well.btn-toolbar

= link\_to 'new', new\_resource\_path, class: 'btn btn-primary'

= link\_to 'Cancel', admin\_chains\_path, class: 'btn btn-default'

= table\_for collection, html: { class: 'table table-bordered' } do

- column :id

- column :product\_name

- column :image do |image|

- image\_tag image.try { |image| image.attachment.url(:thumb) }

- column :price

- column :economy

- column :created\_at

- column :updated\_at

- column title: 'Actions' do |discount|

- discounts\_actions(discount)

= simple\_form\_for [:admin, resource], html: { class: 'form-horizontal' },

wrapper: :horizontal\_form do |f|

.well.btn-toolbar

= f.button :submit, class: 'btn-primary'

= link\_to 'Cancel', admin\_root\_path, class: 'btn btn-default'

= f.input :image\_density

h3 Sheets

.well.btn-toolbar

= link\_to 'new', new\_resource\_path, class: 'btn btn-primary'

= link\_to 'Cancel', admin\_chains\_path, class: 'btn btn-default'

= table\_for collection, html: { class: 'table table-bordered' } do

- column :id

- column title: 'Images' do |sheet|

- sheet\_images(sheet)

- column :created\_at

- column :updated\_at

- column title: 'Actions' do |sheet|

- sheets\_actions(sheet)

nav.navbar.navbar-default

.container-fluid

.navbar-header

a.navbar-brand href='/admin'

| DiscountsWithMe

.collapse.navbar-collapse

ul.nav.navbar-nav

li = link\_to 'Chain stores', admin\_chains\_path

ul.nav.navbar-nav

li = link\_to 'Settings', edit\_admin\_setting\_path(Setting.instance)

ul.nav.navbar-nav.navbar-right

= render 'layouts/account\_links'

class V1::Chains < Grape::API

include Grape::Kaminari

resources :chains do

desc 'Returns all chains.'

get do

Chain.all

end

route\_param :id do

desc 'Returns discounts of chain with id passed in params.'

params do

requires :id, type: Integer, desc: 'Chain id'

end

paginate

get :discounts do

paginate Discount.where(chain\_id: params[:id]).includes(:image)

end

end

end

end

class API < Grape::API

prefix 'api'

mount V1::API

add\_swagger\_documentation api\_version: 'v1',

hide\_documentation\_path: true

end

Rails.application.configure do

config.action\_mailer.default\_url\_options = { host: 'localhost', port: 3000 }

# Settings specified here will take precedence over those in config/application.rb.

# In the development environment your application's code is reloaded on

# every request. This slows down response time but is perfect for development

# since you don't have to restart the web server when you make code changes.

config.cache\_classes = false

# Do not eager load code on boot.

config.eager\_load = false

# Show full error reports and disable caching.

config.consider\_all\_requests\_local = true

config.action\_controller.perform\_caching = false

# Don't care if the mailer can't send.

config.action\_mailer.raise\_delivery\_errors = false

# Print deprecation notices to the Rails logger.

config.active\_support.deprecation = :log

# Raise an error on page load if there are pending migrations.

config.active\_record.migration\_error = :page\_load

# Debug mode disables concatenation and preprocessing of assets.

# This option may cause significant delays in view rendering with a large

# number of complex assets.

config.assets.debug = true

# Asset digests allow you to set far-future HTTP expiration dates on all assets,

# yet still be able to expire them through the digest params.

config.assets.digest = true

# Adds additional error checking when serving assets at runtime.

# Checks for improperly declared sprockets dependencies.

# Raises helpful error messages.

config.assets.raise\_runtime\_errors = true

# Raises error for missing translations

# config.action\_view.raise\_on\_missing\_translations = true

End

require File.expand\_path('test\_helper', File.dirname(\_\_FILE\_\_))

class Book; end

class Folder; end

class BooksController < InheritedResources::Base

custom\_actions :collection => :search, :resource => [:delete]

actions :index, :show

end

class ReadersController < InheritedResources::Base

actions :all, :except => [ :edit, :update ]

end

class FoldersController < InheritedResources::Base

end

class Dean

def self.human\_name; 'Dean'; end

end

class DeansController < InheritedResources::Base

belongs\_to :school

end

module Controller

class User; end

class UsersController < InheritedResources::Base; end

module Admin

class UsersController < InheritedResources::Base; end

end

end

class ControllerGroup; end

module Controller

class GroupsController < InheritedResources::Base; end

end

module Library

class Category

end

class Subcategory

end

class SubcategoriesController < InheritedResources::Base

end

end

class ActionsClassMethodTest < ActionController::TestCase

tests BooksController

def test\_cannot\_render\_actions

assert\_raise AbstractController::ActionNotFound do

get :new

end

end

def test\_actions\_are\_undefined

action\_methods = BooksController.send(:action\_methods).map(&:to\_sym)

assert\_equal 4, action\_methods.size

[:index, :show, :delete, :search].each do |action|

assert action\_methods.include?(action)

end

instance\_methods = BooksController.send(:instance\_methods).map(&:to\_sym)

[:new, :edit, :create, :update, :destroy].each do |action|

assert !instance\_methods.include?(action)

end

end

def test\_actions\_are\_undefined\_when\_except\_option\_is\_given

action\_methods = ReadersController.send(:action\_methods)

assert\_equal 5, action\_methods.size

['index', 'new', 'show', 'create', 'destroy'].each do |action|

assert action\_methods.include? action

end

end

end

class DefaultsClassMethodTest < ActiveSupport::TestCase

def test\_resource\_class\_is\_set\_to\_nil\_when\_resource\_model\_cannot\_be\_found

assert\_nil ReadersController.send(:resource\_class)

end

def test\_defaults\_are\_set

assert\_equal Folder, FoldersController.send(:resource\_class)

assert\_equal :folder, FoldersController.send(:resources\_configuration)[:self][:instance\_name]

assert\_equal :folders, FoldersController.send(:resources\_configuration)[:self][:collection\_name]

end

def test\_defaults\_can\_be\_overwriten

BooksController.send(:defaults, :resource\_class => String, :instance\_name => 'string', :collection\_name => 'strings')

assert\_equal String, BooksController.send(:resource\_class)

assert\_equal :string, BooksController.send(:resources\_configuration)[:self][:instance\_name]

assert\_equal :strings, BooksController.send(:resources\_configuration)[:self][:collection\_name]

BooksController.send(:defaults, :class\_name => 'Fixnum', :instance\_name => :fixnum, :collection\_name => :fixnums)

assert\_equal Fixnum, BooksController.send(:resource\_class)

assert\_equal :fixnum, BooksController.send(:resources\_configuration)[:self][:instance\_name]

assert\_equal :fixnums, BooksController.send(:resources\_configuration)[:self][:collection\_name]

end

def test\_defaults\_raises\_invalid\_key

assert\_raise ArgumentError do

BooksController.send(:defaults, :boom => String)

end

end

def test\_url\_helpers\_are\_recreated\_when\_defaults\_change

BooksController.expects(:create\_resources\_url\_helpers!).returns(true).once

BooksController.send(:defaults, :instance\_name => 'string', :collection\_name => 'strings')

end

end

class BelongsToErrorsTest < ActiveSupport::TestCase

def test\_belongs\_to\_raise\_errors\_with\_invalid\_arguments

assert\_raise ArgumentError do

DeansController.send(:belongs\_to)

end

assert\_raise ArgumentError do

DeansController.send(:belongs\_to, :nice, :invalid\_key => '')

end

end

def test\_belongs\_to\_raises\_an\_error\_when\_multiple\_associations\_are\_given\_with\_options

assert\_raise ArgumentError do

DeansController.send(:belongs\_to, :arguments, :with\_options, :parent\_class => Book)

end

end

def test\_url\_helpers\_are\_recreated\_just\_once\_when\_belongs\_to\_is\_called\_with\_block

DeansController.expects(:create\_resources\_url\_helpers!).returns(true).once

DeansController.send(:belongs\_to, :school) do

belongs\_to :association

end

ensure

DeansController.send(:parents\_symbols=, [:school])

end

def test\_url\_helpers\_are\_recreated\_just\_once\_when\_belongs\_to\_is\_called\_with\_multiple\_blocks

DeansController.expects(:create\_resources\_url\_helpers!).returns(true).once

DeansController.send(:belongs\_to, :school) do

belongs\_to :association do

belongs\_to :nested

end

end

ensure

DeansController.send(:parents\_symbols=, [:school])

end

def test\_belongs\_to\_for\_namespaced\_controller\_and\_namespaced\_model\_fetches\_model\_in\_the\_namespace\_firstly

Library::SubcategoriesController.send(:belongs\_to, :category)

assert\_equal Library::Category, Library::SubcategoriesController.resources\_configuration[:category][:parent\_class]

end

def test\_belongs\_to\_for\_namespaced\_controller\_and\_non\_namespaced\_model\_sets\_parent\_class\_properly

Library::SubcategoriesController.send(:belongs\_to, :book)

assert\_equal Book, Library::SubcategoriesController.resources\_configuration[:book][:parent\_class]

end

def test\_belongs\_to\_without\_namespace\_sets\_parent\_class\_properly

FoldersController.send(:belongs\_to, :book)

assert\_equal Book, FoldersController.resources\_configuration[:book][:parent\_class]

end

end

class SpecialCasesClassMethodTest < ActionController::TestCase

def test\_resource\_class\_to\_corresponding\_model\_class

assert\_equal Controller::User, Controller::UsersController.send(:resource\_class)

assert\_equal Controller::User, Controller::Admin::UsersController.send(:resource\_class)

assert\_equal ControllerGroup, Controller::GroupsController.send(:resource\_class)

end

end

module Virtus

module Model

# @api private

def self.included(descendant)

super

descendant.send(:include, ClassInclusions)

end

# @api private

def self.extended(descendant)

super

descendant.extend(Extensions)

end

module Core

# @api private

def self.included(descendant)

super

descendant.extend(ClassMethods)

descendant.send(:include, ClassInclusions::Methods)

descendant.send(:include, InstanceMethods)

end

private\_class\_method :included

# @api private

def self.extended(descendant)

super

descendant.extend(Extensions::Methods)

descendant.extend(InstanceMethods)

end

private\_class\_method :included

end # Core

module Constructor

# @api private

def self.included(descendant)

super

descendant.send(:include, InstanceMethods::Constructor)

end

private\_class\_method :included

end # Constructor

module MassAssignment

# @api private

def self.included(descendant)

super

descendant.extend(Extensions::AllowedWriterMethods)

descendant.send(:include, InstanceMethods::MassAssignment)

end

private\_class\_method :included

# @api private

def self.extended(descendant)

super

descendant.extend(Extensions::AllowedWriterMethods)

descendant.extend(InstanceMethods::MassAssignment)

end

private\_class\_method :extended

end # MassAssignment

end # Model

end # Virtus

require 'spec\_helper'

module Hashie

module Extensions

describe DeepFetch do

subject { Class.new(Hash) { include Hashie::Extensions::DeepFetch } }

let(:hash) do

{

library: {

books: [

{ title: 'Call of the Wild' },

{ title: 'Moby Dick' }

],

shelves: nil,

location: {

address: '123 Library St.'

}

}

}

end

let(:instance) { subject.new.update(hash) }

describe '#deep\_fetch' do

it 'extracts a value from a nested hash' do

expect(instance.deep\_fetch(:library, :location, :address)).to eq('123 Library St.')

end

it 'extracts a value from a nested array' do

expect(instance.deep\_fetch(:library, :books, 1, :title)).to eq('Moby Dick')

end

context 'when one of the keys is not present' do

context 'when a block is provided' do

it 'returns the value of the block' do

value = instance.deep\_fetch(:library, :unknown\_key, :location) { 'block value' }

expect(value).to eq('block value')

end

end

context 'when a block is not provided' do

context 'when the nested object is an array' do

it 'raises an UndefinedPathError' do

expect do

instance.deep\_fetch(:library, :books, 2)

end.to(

raise\_error(

DeepFetch::UndefinedPathError,

'Could not fetch path (library > books > 2) at 2'

)

)

end

end

context 'when the nested object is a hash' do

it 'raises a UndefinedPathError' do

expect do

instance.deep\_fetch(:library, :location, :unknown\_key)

end.to(

raise\_error(

DeepFetch::UndefinedPathError,

'Could not fetch path (library > location > unknown\_key) at unknown\_key'

)

)

end

end

context 'when the nested object is missing' do

it 'raises an UndefinedPathError' do

expect do

instance.deep\_fetch(:library, :unknown\_key, :books)

end.to(

raise\_error(

DeepFetch::UndefinedPathError,

'Could not fetch path (library > unknown\_key > books) at unknown\_key'

)

)

end

end

context 'when the nested object is nil' do

it 'raises an UndefinedPathError' do

expect do

instance.deep\_fetch(:library, :shelves, :address)

end.to(

raise\_error(

DeepFetch::UndefinedPathError,

'Could not fetch path (library > shelves > address) at address'

)

)

end

end

end

end

end

end

end

end

require 'active\_support/core\_ext/hash/indifferent\_access'

require 'rack/utils'

module ActionDispatch

class TestRequest < Request

DEFAULT\_ENV = Rack::MockRequest.env\_for('/',

'HTTP\_HOST' => 'test.host',

'REMOTE\_ADDR' => '0.0.0.0',

'HTTP\_USER\_AGENT' => 'Rails Testing'

)

def self.new(env = {})

super

end

def initialize(env = {})

env = Rails.application.env\_config.merge(env) if defined?(Rails.application) && Rails.application

super(default\_env.merge(env))

end

def request\_method=(method)

@env['REQUEST\_METHOD'] = method.to\_s.upcase

end

def host=(host)

@env['HTTP\_HOST'] = host

end

def port=(number)

@env['SERVER\_PORT'] = number.to\_i

end

def request\_uri=(uri)

@env['REQUEST\_URI'] = uri

end

def path=(path)

@env['PATH\_INFO'] = path

end

def action=(action\_name)

path\_parameters[:action] = action\_name.to\_s

end

def if\_modified\_since=(last\_modified)

@env['HTTP\_IF\_MODIFIED\_SINCE'] = last\_modified

end

def if\_none\_match=(etag)

@env['HTTP\_IF\_NONE\_MATCH'] = etag

end

def remote\_addr=(addr)

@env['REMOTE\_ADDR'] = addr

end

def user\_agent=(user\_agent)

@env['HTTP\_USER\_AGENT'] = user\_agent

end

def accept=(mime\_types)

@env.delete('action\_dispatch.request.accepts')

@env['HTTP\_ACCEPT'] = Array(mime\_types).collect { |mime\_type| mime\_type.to\_s }.join(",")

end

alias :rack\_cookies :cookies

def cookies

@cookies ||= {}.with\_indifferent\_access

end

private

def default\_env

DEFAULT\_ENV

end

end

end

module ThreadSafe

module Util

# Provides a cheapest possible (mainly in terms of memory usage) +Mutex+

# with the +ConditionVariable+ bundled in.

#

# Usage:

# class A

# include CheapLockable

#

# def do\_exlusively

# cheap\_synchronize { yield }

# end

#

# def wait\_for\_something

# cheap\_synchronize do

# cheap\_wait until resource\_available?

# do\_something

# cheap\_broadcast # wake up others

# end

# end

# end

module CheapLockable

private

engine = defined?(RUBY\_ENGINE) && RUBY\_ENGINE

if engine == 'rbx'

# Making use of the Rubinius' ability to lock via object headers to avoid the overhead of the extra Mutex objects.

def cheap\_synchronize

Rubinius.lock(self)

begin

yield

ensure

Rubinius.unlock(self)

end

end

def cheap\_wait

wchan = Rubinius::Channel.new

begin

waiters = @waiters ||= []

waiters.push wchan

Rubinius.unlock(self)

signaled = wchan.receive\_timeout nil

ensure

Rubinius.lock(self)

unless signaled or waiters.delete(wchan)

# we timed out, but got signaled afterwards (e.g. while waiting to

# acquire @lock), so pass that signal on to the next waiter

waiters.shift << true unless waiters.empty?

end

end

self

end

def cheap\_broadcast

waiters = @waiters ||= []

waiters.shift << true until waiters.empty?

self

end

elsif engine == 'jruby'

# Use Java's native synchronized (this) { wait(); notifyAll(); } to avoid the overhead of the extra Mutex objects

require 'jruby'

def cheap\_synchronize

JRuby.reference0(self).synchronized { yield }

end

def cheap\_wait

JRuby.reference0(self).wait

end

def cheap\_broadcast

JRuby.reference0(self).notify\_all

end

else

require 'thread'

extend Volatile

attr\_volatile :mutex

# Non-reentrant Mutex#syncrhonize

def cheap\_synchronize

true until (my\_mutex = mutex) || cas\_mutex(nil, my\_mutex = Mutex.new)

my\_mutex.synchronize { yield }

end

# Releases this object's +cheap\_synchronize+ lock and goes to sleep waiting for other threads to +cheap\_broadcast+, reacquires the lock on wakeup.

# Must only be called in +cheap\_broadcast+'s block.

def cheap\_wait

conditional\_variable = @conditional\_variable ||= ConditionVariable.new

conditional\_variable.wait(mutex)

end

# Wakes up all threads waiting for this object's +cheap\_synchronize+ lock.

# Must only be called in +cheap\_broadcast+'s block.

def cheap\_broadcast

if conditional\_variable = @conditional\_variable

conditional\_variable.broadcast

end

end

end

end

end

end

require 'thread'

require 'rack/body\_proxy'

module Rack

# Rack::Lock locks every request inside a mutex, so that every request

# will effectively be executed synchronously.

class Lock

FLAG = 'rack.multithread'.freeze

def initialize(app, mutex = Mutex.new)

@app, @mutex = app, mutex

end

def request\_method=(method)

@env['REQUEST\_METHOD'] = method.to\_s.upcase

end

def host=(host)

@env['HTTP\_HOST'] = host

end

def port=(number)

@env['SERVER\_PORT'] = number.to\_i

end

def request\_uri=(uri)

@env['REQUEST\_URI'] = uri

End

def call(env)

old, env[FLAG] = env[FLAG], false

@mutex.lock

response = @app.call(env)

body = BodyProxy.new(response[2]) { @mutex.unlock }

response[2] = body

response

ensure

@mutex.unlock unless body

env[FLAG] = old

end

end

end