МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования

«Гомельский государственный университет

имени Франциска Скорины»

Факультет математики и технологий программирования

Кафедра вычислительной математики и программирования

|  |  |
| --- | --- |
|  | Допущена к защите  Зав. кафедрой\_\_\_\_\_\_\_\_\_\_\_\_\_Д.С. Кузьменков  "\_\_\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2018 г. |

**Создание Web-сервиса генерации псевдо рукописных документов на основании напечатанного текста с использованием технологии Bootstrap, Django, Mysql, Amazon S3 и Celery+RabbitMQ**

**Дипломная работа**

Исполнитель

студент группы ПМ-52 \_\_\_\_\_\_\_\_\_\_\_\_ E.C. Бычков

Научный руководитель

к.ф.-м.н., доцент \_\_\_\_\_\_\_\_\_\_\_\_ П.В. Бычков

Рецензент

к.ф.-м.н., доцент \_\_\_\_\_\_\_\_\_\_\_\_\_ А.В. Клименко

**Гомель 2018**

**Содержание**

[Введение 3](#_Toc515229311)

[1. Предметная область 4](#_Toc515229312)

[1.1 История и предпосылки возникновения 4](#_Toc515229313)

[1.2 Формы электронной коммерции 5](#_Toc515229314)

[1.3 Актуальность электронной коммерции 6](#_Toc515229315)

[1.4 Преимущества электронной коммерции 7](#_Toc515229316)

[1.5 Ограничения электронной коммерции 9](#_Toc515229317)

[1.6 Будущее электронной коммерции 9](#_Toc515229318)

[1.7 Цели и задачи Дипломного проекта 10](#_Toc515229319)

[2 Обзор технологий для разработки 11](#_Toc515229320)

[2.1 Ситуация на рынке серверных языков 11](#_Toc515229321)

[2.2 Базы данных и СУБД 12](#_Toc515229322)

[2.3 Обоснование выбора в пользу Python + Django 16](#_Toc515229323)

[2.4 Amazon Web Services 18](#_Toc515229324)

[3 Введение в Django 20](#_Toc515229325)

[3.1 История 20](#_Toc515229326)

[3.2 MVC через призму Django 20](#_Toc515229327)

[3.3 Django Models 22](#_Toc515229328)

[4 Создание сайта 23](#_Toc515229329)

[4.1 Установка Django и создание проекта 24](#_Toc515229330)

[4.2 Создание Почерков 25](#_Toc515229331)

[4.3 Система шаблонов 29](#_Toc515229332)

[4.4 Управление платежами 32](#_Toc515229333)

[4.5 Локализация и интернационализация 35](#_Toc515229334)

[5. Экономическая часть 39](#_Toc515229335)

[5.1 Расчет длительности этапа разработки 40](#_Toc515229336)

[5.2 Расчет заработной платы разработчика 40](#_Toc515229337)

[5.3 Расчет амортизационных отчислений основных средств 41](#_Toc515229338)

[5.4 Расчет затрат на сырье и материалы 43](#_Toc515229339)

[5.5 Вывод 44](#_Toc515229340)

[Заключение 45](#_Toc515229341)

# Введение

# 1. Предметная область

Электронная коммерция либо, её международное название, E-commerce считается видом бизнес-модели или сегментом бизнес-модели побольше, которая дает возможность компании либо отдельному человеку реализовывать предпринимательство с помощью электронной сети, как правило сети интернет.

## 1.1 История и предпосылки возникновения

Электронная коммерция формировалась весьма стремительными темпами последние лет 10-15. Её хроника наступает с 1991 года, когда сеть интернет стала открыта для коммерческого приложения.

По началу, слово «электронная коммерция» означал процесс выполнения торговой транзакции электронно, при поддержке распространенных в этот момент технологий Electronic Data Interchange (EDI) и Electronic Funds Transfer (EFT), которые давали возможность пользователям делиться бизнес-данными и осуществлять электронные транзакции. Данные технологические процессы возникли ещё в 70-х годах и определенные фирмы, и организации пользовались ими с целью передачи коммерческих бумаг в электронном варианте.

Несмотря на то, что сеть интернет начала приобретать собственную известность среди массового населения только в 1994 года, понадобилось порядка 4 лет на разработку протокола, отвечающих за защищенность (к примеру, HTTP) и DSL, давший высокоскоростной доступ и непрерывное соединение с вебом. В 2000 г. колоссальное число фирм в США и Западной Европе представляли собственные сервисы в сети интернет. В то время роль определения E-commerce поменялось. Общество стало определять его как процесс приобретения общедоступных продуктов и сервисов посредством сети интернет при помощи безопасного соединения и электронных сервисов оплаты. Несмотря на то, что крах «дот-комов» в 2000 г. привел к негативным результатам и многочисленные компании, занимавшиеся подобным типом бизнеса закрылись, все больше фирм стали признавать достоинства электронной коммерции и стали дополнять вероятность покупки собственных продуктов посредством сеть интернет. Уже к окончанию 2001 года наиболее обширная сфера электронной коммерции – модель B2B (бизнес для бизнеса) – обладала в своем счету порядка 700 млрд $ в транзакциях.

Электронная коммерция обладала рядом положительных сторон по сравнению с классическими торговыми центрами. Покупатели могли с легкостью просматривать большие каталоги товаров и услуг. Могли видеть актуальные цены, формировать и обмениваться списками желаний (тогда ещё по электронной почте) и находить наиболее интересные предложения.

Со стороны продавца также существовали собственные плюсы. Сеть интернет с собственными поисковыми системами гарантировали неплохой способ рассказать о собственном существовании возможным потребителям, никак не расходуясь при этом на маркетинговые кампании. Даже маленькие интернет-магазины имели возможность выходить на мировые рынки. Веб-технологии, по мимо всего остального, позволили прослеживать предпочтения юзеров и обеспечивать им персональные предложения и рекламу.

## 1.2 Формы электронной коммерции

Электронная коммерция подразделяется на 5 основных форм:

а) Business to Business (B2B): данная модель подразумевает продажу собственных услуг либо товаров одним производителем или организацией другой. Отличительными особенностями являются цены, отличающиеся для различных клиентов (они представляют разный интерес), непростая процедура заказа, крупные размеры отдельных заказов. Является одной из наиболее крупных (в деньгах) форм

б) Business to Consumer (B2C): форма, когда фирма продает что- либо покупателю. Одним из наиболее распространенных примеров является интернет- магазин;

в) Consumer to Consumer (C2C): достаточно современная форма, предполагающая связь покупателей между собой. В отличие от классического подхода (B2C), когда покупатель направляется к фирме с целью приобрести продукт, в C2C фирма предоставляет площадку для торговли пользователям;

г) Business to Government (B2G): считается производной от B2B формой, охватывающей реализацию товаров и сервисов всевозможным муниципальным структурам на различных уровнях через особые, зачастую интегрированные концепции. Одним из образцов считаются концепции электронных госзакупок;

д) Government to Consumer (G2C): для предоставления услуг жителям и ускорения отдельных внутренних процессов правительство использует аналогичные системы. Реализация различных документов, конфигураций, копий и т.д.

Наиболее многочисленными являются формы B2B и B2C. У магазинов, имеющих отношение к этим формам довольно много общего, к примеру: наличие поиска, навигации, детальное представление отдельного продукта, страницы с историей заказа. Но, в отдельных случаях, отличия крайне существенные. Множество B2B фирм обладают довольно непростой процедурой заказа, сами заказы совершаются на регулярной основе, в отличие от единичных покупок в случае с B2C, сами поставки происходят не в режиме скорейшей доставки, а при помощи созданных грузоперевозок. Для наглядности объединим ключевые различия в таблицу (таблица 1).

**Таблица 1 - Характерные черты B2C и B2B**

|  |  |
| --- | --- |
| **B2C** | **B2B** |
| Отдельный покупатель | Орган принятия решений |
| Фиксированные цены для всех покупателей | Специализированные цены для разных заказчиков |
| Прямые платежи | Оплата по кредитам |
| Склады для скорейшей доставки | Организованные грузоперевозки |
| Низкая частота покупок | Многократные закупки |
| Единичное посещение | Длительные отношения между закупщиком и производителем |
| Покупки по желанию | Покупки как часть работы |

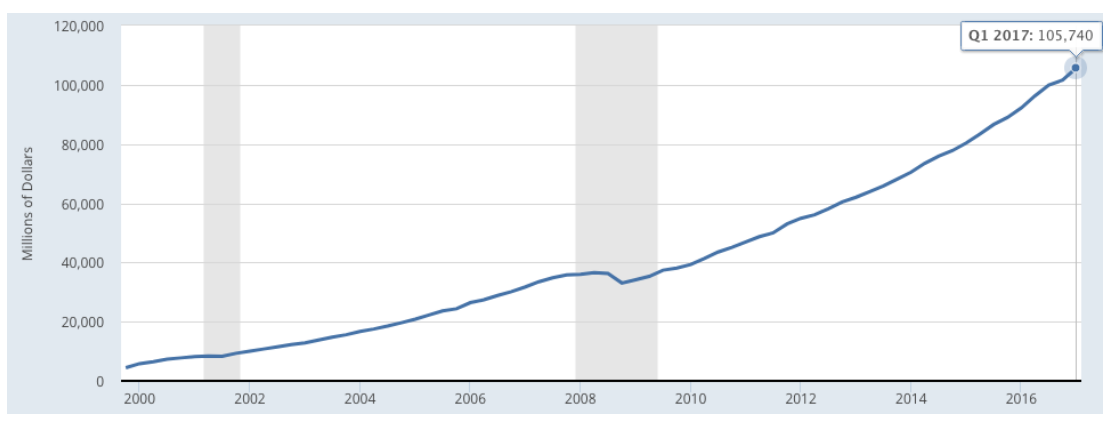
## 1.3 Актуальность электронной коммерции

Обстановка в бизнесе сегодня, равно как и в большей части 20 столетия весьма жесткая и конкурентная и покупатели становятся все более и более влиятельными. Способность компании жить долгое время требует значительных и конкурентоспособных преимуществ по сравнению с прочими фирмами, действующими в данном секторе. Факторы в экономической сфере могут изменяться невероятно стремительно и применение электронной коммерции создает систему наиболее гибкой и чуткой. С её поддержкой возможно прослеживать увлечения покупателей, раскрывать всевозможные закономерности, и, наконец, изменять цены и предложения там, где это целесообразно. Классические способы, наподобие уменьшения себестоимости либо ликвидация бесприбыльных компаний более не считаются необходимыми для того, чтобы оставаться успешными.

Электронная коммерция стала источником того преимущества для завоевания рынка и предоставления лучшего потребительского сервиса.

По информации от Федерального Резервного Банка (Сент-Луис, штат Миссури) на первый квартал 2017 года приходится порядка 105 миллиардов долларов выручки с продаж посредством электронной коммерции. При этом уточняется, что оплата могла быть произведена как наличными, так и картой или другим безналичным способом.

Выручка с продаж посредством электронной коммерции приведена на рисунке 1.



**Рисунок 1**

## 1.4 Преимущества электронной коммерции

Электронная коммерция дает возможность фирмам осуществлять свой бизнес по всему миру, требуя не очень больших инвестиций. Присутствие подобного мирового рынка дает возможность небольшим фирмам в разных точках земли снабжать значительную платежеспособную фирму, при этом обе стороны обладают личный интерес в этом бизнесе.

Электронная коммерция позволяет ликвидировать разнообразные неэффективные нюансы в цепи поставок и уменьшить расходы на руководство. Вероятны сценарии, когда продукт станет произведен уже после того, как будет заказан покупателем – в таком случае уменьшатся расходы на содержание.

С её помощью также вполне вероятно справляться с различными физическими ограничениями. Компьютерные системы готовы гарантировать круглосуточную поддержку пользователей в отсутствии выходных и праздников. Ранее фирмам понадобилось бы встретиться с различными правовыми лимитированиями и возросшей стоимостью ведение бизнеса в таком беспрерывном распорядке.

Электронные трейдерские площадки имеют все шансы предоставлять индивидуальные предложения или рекомендации. К примеру, торговый центр Amazon содержит персонализированный интерфейс, который подстраивается под определенного юзера на основании его покупок и рецензий.

То, как электронная коммерция позволяет компаниям взаимодействовать с потребителями является, наверно, самым важным ее преимуществом. Раньше взаимодействие было односторонним: телевидение, радио, реклама в журналах.

Интернет сделал возможным ситуацию, когда компания отвечает на запросы, желания или рекомендации пользователей, иногда даже в реальном времени. Как результат, потребитель имеет заметно больше контроля и вероятность позитивного для обеих сторон результата возрастает очень сильно.

Наконец, электронная коммерция повысила ценность отдельных покупателей и общества в целом. Люди получили в свое распоряжение безграничный выбор продавцов и товаров, доступных им 24 часа в сутки. Кроме того, они получили возможность собирать информацию об интересующих их товарах быстрее, чем, когда либо, что в итоге повысила вероятность найти товар, который бы их удовлетворил.

Общество в целом скорее всего тоже получило свою выгоду, так как высокая конкуренция способствует снижениям цен, что повышает уровень жизни.

На рисунке 2 приведены данные о том, сколько тратят на покупки в интернете люди из разных стран.



**Рисунок 2**

## 1.5 Ограничения электронной коммерции

Разработка и обслуживание сайта электронной коммерции требует определенных технических знаний. Его функциональность, как правило, эквивалентна всему магазину, а цена ошибки очень большая.

Даже сегодня не все люди доверяют ведению бизнеса в интернете. Уже сам по себе интернет имеет определенные трудности с законом. Определенные юрисдикции трудно установить, в связи с отсутствием там географических и национальных границ. Некоторым людям не совсем комфортно вести с кем-то дело, не имея визуального контакта с человеком. Кроме того, нет возможности предварительно потрогать товар и проверить. Наконец, существуют люди, для которых посещение магазинов своего рода отдых, и они зачастую не чувствуют того же удовольствия от покупок в интернете.

Главной причиной для людей не связываться с электронной коммерцией была и остается безопасность. Большая часть проблем связана с платежами – транзакции, как правило, происходят посредством кредитных карт. Кредитные карты физически не существуют на стороне продавца и не всегда существуют даже на стороне покупателя. Это дает шанс потенциальным мошенникам. Последняя проблема – проблема сохранности и приватности предоставляемых данных. Даже в 2018 году регулярно происходят утечки, когда множество данных о кредитных картах людей оказываются похищенными.

## 1.6 Будущее электронной коммерции

Ее будущее по большому счету зависит от будущего интернета. Все больше и больше людей получает к нему доступ, тем самым увеличивая рынок. Развитие носимой электроники сейчас делает возможные новые более эффективные способы оплаты, у которых конечно со своей стороны тоже много нерешенных проблем. Машинное обучение в купе с биометрическими системами, возможно, расширят границы электронной коммерции еще больше.

## 1.7 Цели и задачи Дипломного проекта

Целью Дипломного проекта является демонстрация альтернативного подхода к разработке веб-сайтов на примере веб-сервиса для генерации псевдо рукописного текста. Предполагается, что веб сервис будет иметь минималистичный дизайн. Главной страницей будет являться рабочая область для генерации конспекта с различными настройками, предоставляющую возможность настраивать отображение почерка и размер страниц. Каждый почерк будет иметь картинку для возможности предпросмотра, название и собственный набор символов. Пользователь будет иметь возможность предварительного просмотра конспекта и возможность скачать полный конспект Вместо регистрации пользователи будут вводить свои данные непосредственно при заказе, после чего им будет предложено его оплатить в одной из электронных платежных систем.

Отдельное внимание планируется уделить отработке «лучших практик» в веб-разработке. Обширный их список был подчерпнут при изучении тематической литературы и различных ресурсов в интернете.

# 2 Обзор технологий для разработки

## 2.1 Ситуация на рынке серверных языков

В этот период PHP применяется в более чем 82% веб-сайтах, середи тех, относительно которых имеется достоверные данные. Данный язык первоначально задумывался в виде серверного и достаточно уже длительное время считается абсолютным фаворитом в плане распространенности. Благодаря этому, есть колоссальное число библиотек и фреймворков на PHP, делающие его годным для почти каждой задачи; помимо этого, сообщество PHP-разработчиков попросту громадно, таким образом то что при появлении тот или иной трудности, отыскать в сети интернет разрешение станет значительно проще, нежели с другими языками. Другое преимущество PHP состоит в том, что он довольно прост в изучении, в отличие, к примеру, от Java или C++. Невзирая на данные значимые достоинства, PHP в последние лет 10 утрачивает собственную известность. Вот ключевые предпосылки этого:

а) посредственный дизайн – это одна из основных претензий в адрес языка. В PHP используется динамическая типизация, которая, с одной стороны, предоставляет больше свободы и немного ускоряет процесс разработки, а с другой – ведет ко многим ошибкам. Кроме того, он основан сразу на множестве других языков, например, Perl, C, C++, Java и т.д., так что в некотором роде он лишен собственного стиля;

б) легкость написания «плохого» кода – этот пункт вытекает из предыдущего. Недостатки в дизайне оставляют пространство для написания неэффективного, неочевидного, непостоянного кода. В связи с невероятной популярностью языка, существует большое количество уже написанного плохого кода, который можно по неопытности использовать в своем проекте. Это также негативно сказывается на работе в команде – разбираться в чужом коде приходится еще более тщательно и делать это становится сложнее;

в) слабая типизация – хотя это и способствует гибкости языка, делает его более сильно подверженным различным багам;

г) масштабируемость – несмотря на то, что PHP хорошо подходит для проектов среднего размера, масштабируемость дальше может вызывать проблемы. Связано это с низкой модульностью языка и уже упомянутыми проблемами излишней гибкости и написания неэффективного кода.

Если не учитывать распространенность, то у PHP появляется несколько серьезных конкурентов. Самые популярные приведены на рисунке 3.



**Рисунок 3**

Останавливаться на всех не будем, вкратце рассмотрим только первые три:

а) Python – язык общего назначения, который можно встретить практически в любой области от робототехники, до финансовых структур и статистического анализа. Его популярность заключается в сочетании простоты в изучении с хорошим и лаконичным дизайном, который способствует быстрой разработке и прототипированию. Для веб-разработки на Python существует множество фреймворков, например Flask, Pyramid и Django. У всех них есть свои достоинства, про достоинства последнего будет рассказано далее;

б) Ruby on Rails – фреймворк для языка Ruby, основное преимущество которого заключается как раз в самом Ruby, а именно: объектно-ориентированность, легкость для чтения и понимания, предсказуемый синтаксис и поведение. Ruby on Rails во многом похож на Django, продвигает схожие идеалы и подход к разработке;

в) Node.js – фреймворк для превращения client-side языка JavaScript в нечто, исполняемое на сервере. Одно из главных преимуществ Node.js заключается в том, что многие люди, занимающиеся веб-разработкой уже знают JavaScript, так что изучение и использование этого фреймворка для них заметно облегчено.

## 2.2 Базы данных и СУБД

Базы данных созданы для того, чтобы решать очень важную проблему: хранение данных. Это могут быть какие угодно данные: сотрудники, товары, заказы, посетители сайтов; могут быть представлены в виде дат, целых или дробных чисел, строк или булевых переменных. На самом деле, одно лишь наличие данных – не достаточно хороший повод заведения базы данных. Многие небольшие компании довольно долго эффективно пользовались таблицами в Excel и аналогичных программах. Сами данные не являются проблемой, но они появляются дальше. Вот список некоторых:

а) Размер – то, что начиналось как небольшой объем данных имеет тенденцию разрастаться. Таким образом однажды хорошо работающий документ, содержащий в себе сотню строк, становится очень неповоротливым и тяжелым, накопив в себе несколько миллионов строк;

б) Простота обновления – при работе с системой, построенной на файлах, возникает проблема, когда несколько пользователей пытаются внести изменения одновременно. В итоге получится что каждый изменяет данные предыдущего;

в) Точность – как правило нет никакой системы защиты от случайных ошибок, так что чем больше разрастается таблица, тем менее достоверная она становится;

г) Безопасность – часто этой таблицей необходимо с кем-то делиться. Но необходимость делиться с кем-то, не означает что делиться можно со всеми. Возникает множество вопросов, которые проблематично решать в рамках данного подхода – кто может просматривать эту таблицу? Кто может ее редактировать? Если таблицу отредактировали, то, как узнать, кто ответственен за какие изменения?

д) Избыточность и дублирование – сама по себе избыточность не является минусом – резервные копии тоже избыточны. Однако относительно самой информации она не слишком полезна, так как ведет к конфликтам;

е) Важность – случается всякое, бывают сбои, ведущие к потере информации. Иногда потеря информации означает лишь небольшую потерю времени, неудобство, но есть множество отраслей, где даже минимальная потеря информации является неприемлемой.

Все это – причины, почему понадобились базы данных и это те проблемы, которые они призваны решить. Базы данных предоставляют структурированный подход к хранению данных, в котором правила настраиваются исходя из потребностей и приоритетов отрасли.

Системы управления базами данных или СУБД – это обобщающий термин, которым обозначаются всевозможные решения (то есть компьютерные программы или встраиваемые библиотеки), зачастую работающие совсем по- разному. Эти приложения управляют или помогают управлять наборами информации. Так как поступающая информация может отличаться в размерах и формах, было разработано множество СУБД.[1]

СУБД построены на моделях – структурах, определенных для управления данными. Каждая появляющаяся СУБД работает отлично от других, касательно определений и операций хранения и получения упомянутых данных.

Каждая СУБД реализует различную модель базы данных, логически структурирующие данные. Эти модели являются первым шагом и главным решающим фактором в вопросе того, как база данных будет обрабатывать и управляться с информацией. Существует несколько типов моделей, которые четко определяют, как именно данные будут структурированы. Самая популярная из них – реляционная.

Реляционная модель была представлена в 70-х годах. Она предлагает очень математический способ структурирования, хранения и использования данных, а также расширения старых концептов, вроде плоской модели, модели сети и т.д. при помощи связей. Связи предоставляют преимущества группированного хранения данных в качестве ограниченных коллекций, посредством чего таблицы, содержащие информацию в структурированном виде (например, имя и адрес), связывают все входные данные, присваивая значения различным атрибутам.

Благодаря десятилетиям изучений и разработок, базы данных, основанные на реляционной модели, работают очень эффективно и надежно. В купе с продолжительным опытом работы у программистов с такими решениями, использование реляционных баз данных стало стандартным выбором в проектах, где сохранность данных являлось приоритетом.

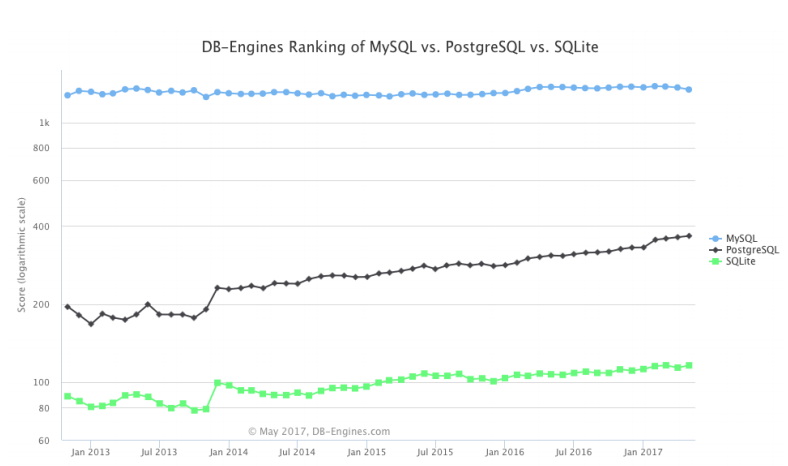
Хотя реляционная модель является очень гибкой и мощной, все же существует несколько проблем, которые ей не решаются.

В последнее время ряд решений, называющихся NoSQL, начали стремительно набирать популярность. Происходило это благодаря обещанию решить те проблемы, которые реляционным базам не удалось, а также благодаря некоторой другой дополнительной функциональности. За счет искоренения жесткой структурированности данных, хранящих описания стилей внутри реляционной модели, эти базы данных работали, предлагая намного более свободный путь работы с данными, обеспечивая, тем самым, простоту и гибкость. Используя такой неструктурированный (или структурируемый на лету) подход, NoSQL решения стремятся избавиться от ограничений, накладываемых строгими связями, и предлагают множество разных способов хранения и использования данных для специфических сценариев использования.

Несмотря на то, что существует масса различных СУБД, в каждый период времени существует лишь небольшое количество решений, которые бы быстро стали популярными и оставались таковыми на протяжении длительного времени. Последние несколько десятилетий безоговорочной популярностью пользуются

Реляционные СУБД. Реляционная СУБД берет свое название от модели, которую она реализует, то есть Реляционной, о которой говорилось ранее. На сегодняшний день и в обозримом будущем они – самый популярный выбор для надежного, эффективного и безопасного хранения данных.

Эти СУБД требуют четко определенного набора схем для того, чтобы принимать данные. Эти схемы определяются пользователем и формируют то, как данные хранятся и используются.

Ниже на рисунке 4 приведено сравнение популярности MySQL, PostgreSQL и SQLite по данным сайта DB-Engines: 

**Рисунок 4.**

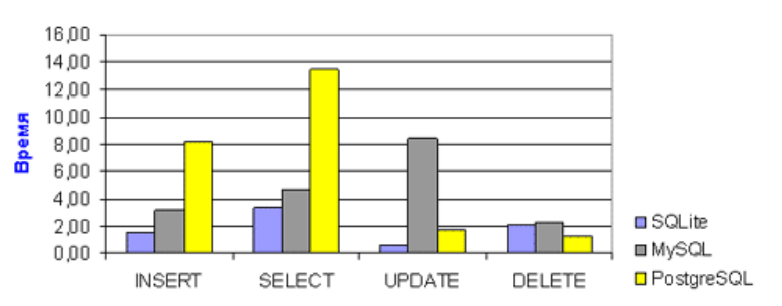
Вот некоторые популярные реляционные СУБД:

а) SQLite – мощная встраиваемая файловая СУБД. Представляет из себя самостоятельную библиотеку, подключаемую в приложение, которое будет ее использовать. SQLite предлагает хороший набор инструментов для работы с любым типом информации с простотой и меньшим количеством ограничений, по сравнению с серверными базами. Когда приложение делает запрос, интеграция обрабатывает его при помощи функций и прямых обращений к файлу, содержащему данные, вместо того, чтобы общаться посредством различных интерфейсов (портов, сокетов). Это делает SQLite очень быстрым и эффективным. Основной недостаток – нет поддержки привилегий пользователей (и системы пользователей вообще);

б) MySQL – самая популярная СУБД, которая используется в огромном множестве сайтов и приложений. Имеет богатую функциональность и относительно прост в изучении. Будучи очень популярным, имеет большую поддержку со стороны сообщества в виде различных обучающих материалов и готовых библиотек. Имеет немало недостатков, среди которых: относительная ограниченность, проблемы с надежностью, стагнация в развитии;

в) PostgreSQL – самая продвинутая СУБД из трех. Является очень расширяемой и программируемой СУБД, поддерживающей собственный вид процедур – «хранимые процедуры». Это функции, которые упрощают выполнение сложных, но повторяющихся операций. Благодаря всему этому является очень универсальным р ешением. Недостаток заключается в том, что в ситуациях, где требуется большое количество простых запросов, PostgreSQL уступает тому же MySQL в производительности.

Сравнение скорости выполнения основных операций приведены на рисунке 5:



**Рисунок 5**

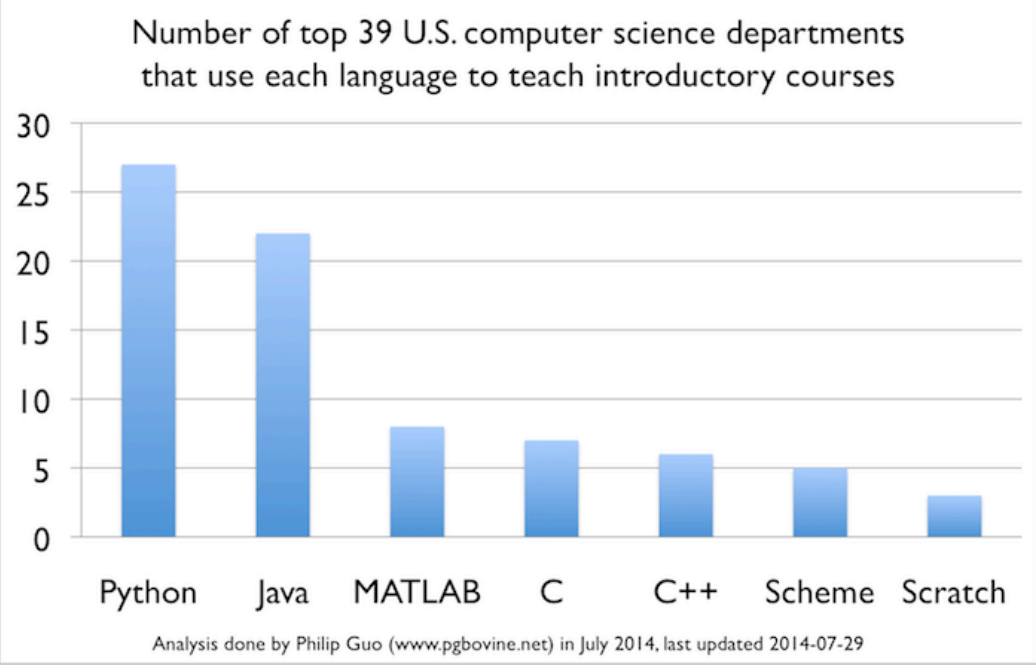
## 2.3 Обоснование выбора в пользу Python + Django

Для разработки дипломной работы было решено использовать Python, а

именно фреймворк Django. Так как написан и предназначен он именно для Python, этот фреймворк старается следовать той же философии. Разработчики стремились сделать его как можно более простым, прямолинейным, очевидным, легкопортируемым. Рассмотрим основные преимущества использования Django.

Python – это, скорее всего, самый простой язык программирования дляизучения. Благодаря использованию конструкций из естественных языков (параграфы и отступы) и простого синтаксиса понимание структуры программы существенно упрощается, по сравнению с другими популярными языками программирования. В доказательство этому, большая доля различных курсов программирования в институтах и колледжах используют именно Python.[2] На приведенном ниже рисунке представлен график, показывающий сколько учебных заведений, входящих в список 39 лучших в США, используют тот или иной язык программирования. Кроме того, Python универсален. Он лежит в основе множества как серверного ПО, так и пользовательских программ на PC и Mac.

Сравнение популярности различных языков программирования в учебных заведениях США представлен на рисунке 6.



**Рисунок 6**

Комплект поставки – обо многих часто используемых в веб-разработке вещах сообщество Django уже позаботилось, так что начиная делать проект не придется писать все с нуля, часто достаточно будет настроить. Примерами такой функциональности являются: приложение для администрации, аутентификация, работы с базой данных, работы со специальными страницами (вроде пользовательского соглашения), геолокация, сессия и обработка cookie и т.д.

Безопасность – безопасность является очень серьезным аспектом и имеет высокий приоритет в Django. Фреймворк снабжен множеством инструментов, позволяющим бороться со многими популярными видами атак, вроде XSS и SQL-инъекций.

Масштабируемость – используемые системы кэширования и дизайн фреймворка предполагает возможность масштабирования. Многие популярные сайты, которые справляются с огромной нагрузкой ежедневно написаны на Django.

Проверен временем – хороший способ узнать, надежен ли какой-либо веб-фреймворк – это посмотреть, долго ли он существует, развивается ли он и кто из больших компаний его использует. С 2005 года, когда был открыт его исходный код, Django развивался и получил широкое распространение среди больших международных компаний, вроде Pinterest, Instagram, Disqus, Bitbucket, EvenBrite и других.

Документация – даже в ранних версиях, разработчики Django старались делать документацию максимально емкой и доступной. Документация этого фреймворка, наравне с документацией для Python, является одной из лучших в отрасли.[3]

## 2.4 Amazon Web Services

Amazon Web Services (AWS) — инфраструктура платформ облачных веб-сервисов, представленная компанией Amazon в начале 2006 года. В данной инфраструктуре представлено много сервисов для предоставления различных услуг, таких как: хранение данных (файловый хостинг, распределённые хранилища данных), аренда виртуальных серверов, предоставление вычислительных мощностей и др [4].

Amazon Web Services предлагает широкий набор глобальных сервисов вычисления, хранилища, базы данных, аналитики, приложения и развертывания, которые помогают организациям быстрее развиваться, сократить затраты на ИТ и масштабировать приложения. Крупнейшие предприятия и многообещающие стартапы доверяют этим сервисам поддержку работы широкого спектра функций, включая мобильные и интернет- приложения, «Интернет вещей», разработку игр, обработку и хранение данных, хранилища, архивацию и многое другое [5].

AWS – это модульный конструктор, в котором клиент сам создаёт требуемый сервис с пользовательскими параметрами конфигурации.

Изначально, платформа AWS задумывалась как настраиваемый доступ к облачному вычислительному ресурсу. С высокой скоростью доступа и максимальной надёжностью хранения данных, достигаемой за счёт кластеризации. Производными от этой идеи являются:

а) виртуальный вычислительный ресурс (более простой термин: "виртуальная машина");

б) виртуальный выделенный сервер (VDS) (под Linux, Windows и др.);

в) облачное хранилище данных (тома памяти для VDS, реляционные базы данных, файловые хранилища и др.);

г) системы настраиваемой аналитики (в том числе и Mobile Analitics);

д) системы распределённой доставки контента (CDN, в терминологии системы - Amazon CloudFront);

е) сервисы почтовых рассылок [6]. AWS Elastic Beanstalk – это простой в использовании сервис для развертывания и масштабирования веб-приложений и сервисов, разработанных с помощью Java, .NET, PHP, Node.js, Python, Ruby, Go и Docker, на серверах Apache, Nginx, Passenger и IIS.

Elastic Beanstalk автоматически выполняет развертывание загруженного кода: выделяет ресурсы, занимается балансировкой нагрузки, автоматическим масштабированием и мониторингом работоспособности приложения. При этом сохраняется полный контроль над ресурсами AWS для разрабатываемого приложения Доступ к ним можно получить в любое время [7].

# 3 Введение в Django

Django – это бесплатный веб-фреймворк с открытым исходным кодом, написанный на Python. Он поддерживается Django Software Foundation (DSF), независимой некоммерческой организацией.

## 3.1 История

Django был создан осенью 2003 года двумя веб-программистами – Адрианом Головатый и Саймоном Виллисоном во время их работы в газетном издательстве Lawrence Journal-World. Как раз тогда они начали использовать Python для создания веб-приложений. В июле 2005 года Django был выпущен публично с лицензией BSD. Название такое фреймворк получил в честь французского джазз гитариста Джанго Рейнхардта.

В июле 2008 было объявлено, что только что организованная компания Django Software Foundation будет заниматься поддержкой Django в будущем.

## 3.2 MVC через призму Django

MVC – шаблонный дизайн Model-View-Controller. Концепт, предполагающий разделения данных приложения, пользовательского интерфейса и логики на соответствующие компоненты: модель, представление, контроллер.

Этот концепт существует уже очень долгое время – впервые был описан Трюгве Реенскаугом еще в 1978 году. Однако получил широкое распространение и экспоненциальный рост с развитием интернета, так как оказалось, что это лучший способ проектирования клиент-серверных приложений. Все лучшие современные веб-фреймворки построены вокруг концепта MVC. Сам концепт довольно прост для понимания:

a) Model(M) – модель или формализация данных. Это не сами данные, а интерфейс для доступа к ним. Модель позволяет получать данные с базы данных, не требуя знания тонкостей этой самой базы. Модель зачастую обеспечивает уровень абстракции над базой данных, так что одна и та же модель может быть использована с разными базами данных;

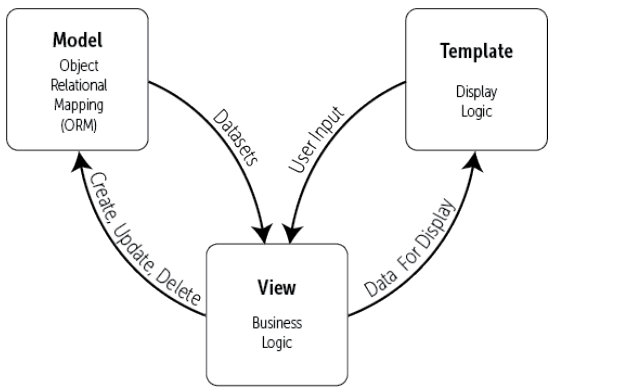
б) View(V) – это то что мы видим, слой представления для нашей модели. На компьютере это, что мы видим в браузере при работе с веб- приложением, например. View также обеспечивает интерфейс для сбора пользовательского ввода;

в) Controller(C) – контролирует течение информации между model и view, используя программную логику чтобы решить, какая информация должна быть извлечена из базы данных и какая информация передана view. Он также получает информацию от пользователя при помощи view и вносит соответствующие изменения в отображение или изменяет сами данные через model.

Сложности возникают там, где требуется конкретно определить, на каком уровне что происходит – разные фреймворки имплементируют схожую функциональность по-разному.

Так и в случае с Django – несмотря на то, что он четко следует MVC, не обошлось без собственной логики в имплементации. Имеется в виду, что по большому счету Controller управляется самим фреймворком, а разработчик взаимодействует с model, темплейтами и view (MTV). Про темплейты будет подробнее дальше.

Графическое представление модели MTV приведено на рисунке 7.



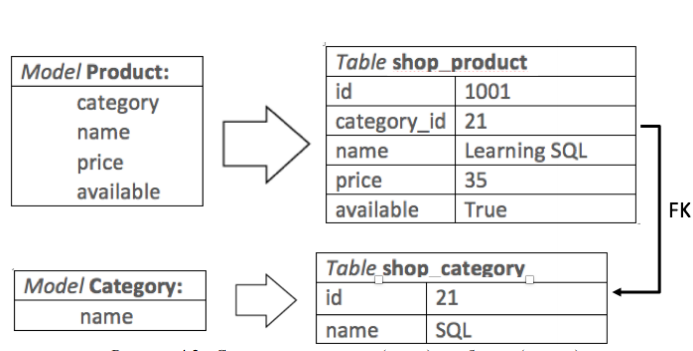
**Рисунок 7**

## 3.3 Django Models

Модели обеспечивают уровень абстракции для базы данных. Используемая техника носит название Object-relational Mapping (ORM).

Большая часть баз данных используют SQL, однако каждая из них реализует это по-своему. Знание тонкостей SQL может быть довольно проблематичным. ORM, с другой стороны, обеспечивает простое сопоставление между объектом и базой данных. При этом программисту не нужно знать тонкостей устройства используемой базы данных или уметь писать сложные SQL запросы.

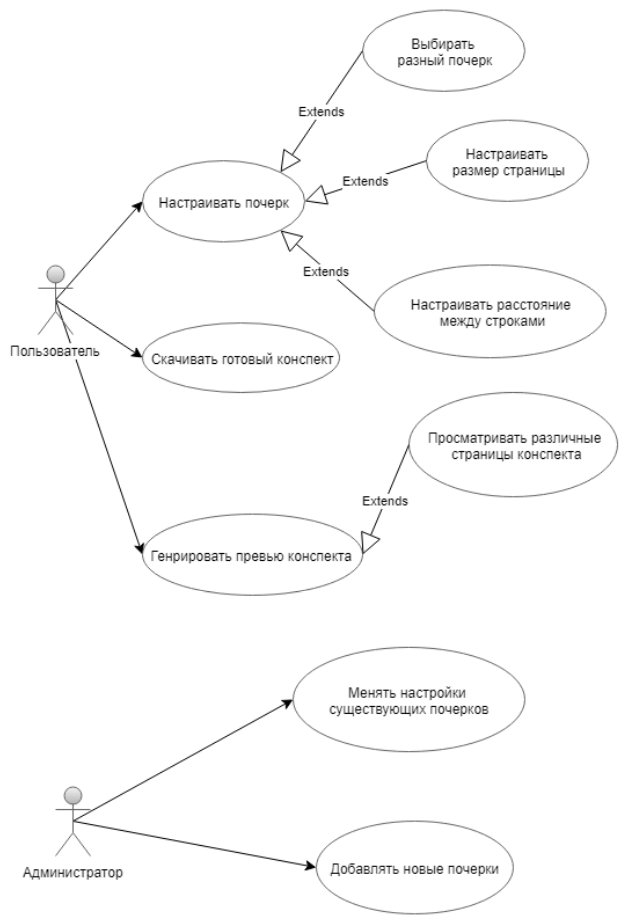
В Django модель - и есть этот объект. При создании модели Django создает соответствующую таблицу в базе данных, без необходимости писать SQL вручную. Это продемонстрировано на рисунке 8.



**Рисунок 8**

# 4 Создание сайта

Сайт будет представлять из себя простой интернет-ресурс с возможностью предпросмотра конспекта и генерации полноформатного конспекта. Use-Case диаграмма представлена на рисунке 9.



**Рисунок 9**

На приведенной Use-Case диаграмме видны предполагаемые сценарии использования для пользователя и администратора.

Пользователь гибко конфигурировать почерк для достижения наибольшего сходства с оригиналом, просматривать отдельные страницы конспекта для проверки их корректной генерации, и скачивать конспект к себе на персональный компьютер для последующей печати.

Администратор отвечает за добавление новых почерков и настройку уже существующих.

## 4.1 Установка Django и создание проекта

Django работает как с Python 2.7, так и с 3.х. В данной работе будет использоваться Python версии 3.6.2 – последняя на момент написания работы версия. Для установки всех дополнительных пакетов в Python используется PIP (акроним для «Pip Installs Packages»). PIP поставляется вместе с Python 2.7.9 и позднее и Python 3.4 и позднее.

Находясь в активном виртуальном окружении следующим шагом, будет установка Django. Она ничем не отличается от установки большинства других пакетов в Python, устанавливается все так же через утилиту PIP.

После завершения установки можно создавать проект. Делается это командой:

django-admin startproject SmartNote

Будет создана директория проекта SmartNote внутри той директории, где мы находились до этого. Вот что создала команда startproject:

├── manage.py

└── SmartNote

├── \_\_init\_\_.py

├── settings.py

├── urls.py

└── wsgi.py

Немного подробнее об этих файлах:

а) manage.py – утилита для командной строки, позволяющая взаимодействовать с нашим проектом многими способами;

б) SmartNote /\_\_init\_\_.py – пустой файл, который указывает Python, что данная директория SmartNote является модулем Python;

в) SmartNote /settings.py – конфигурационный файл для данного проекта. В нем определяются используемая база данных, параметры подключения к ней, подключенные к проекту приложения и многое другое;

г) SmartNote /urls.py – определение URL’ов для данного проекта; в некотором смысле «содержание» будущего сайта;

д) SmartNote /wsgi.py – входная точка для WSGI-совместимых веб- серверов для исполнения нашего проекта.

После этого создаем нужные приложения, из которых и будет состоять будущий сайт. Произведем это на примере приложения NoteCreator:

django-admin startapp shop

И затем подключаем это приложение в конфигурационном файле settings.py:

INSTALLED\_APPS = [

'FontParser.apps.FontparserConfig',

**'NoteCreator.apps.NotecreatorConfig',**

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles'

]

Теперь это приложение активно в данном проекте.

## 4.2 Создание Почерков

Основой интернет-магазина является буква в почерке. Каждый символ своим атрибутом будет иметь имя, путь к изображению , параметры ширины , высоты и отступа, а так же указание на то, к какому почерку относится данный символ. Все символы будут распределены по почеркам. Для создания почерков и букв для них обратимся к файлу models.py, который Django автоматически создал при инициализации приложения NoteCreator. Добавим туда следующий код:

from django.db import models  
from django.contrib.auth.models import User  
  
  
   
class Font(models.Model):  
 name = models.CharField(max\_length=10)  
 preview\_image = models.ImageField(upload\_to='fonts/previews/' ,  
 null=True)  
   
 def \_\_str\_\_(self): return self.name  
   
   
class Letter(models.Model):  
 char = models.CharField(max\_length=1)  
 font = models.ForeignKey(Font ,  
 on\_delete=models.CASCADE)  
 img = models.ImageField(upload\_to='fonts/')  
 x\_offset = models.SmallIntegerField(default=0)  
 y\_offset = models.SmallIntegerField(default=0)  
 width = models.DecimalField(max\_digits=10, decimal\_places=2)  
 approved = models.NullBooleanField(default=False)  
   
 def \_\_str\_\_(self): return str(self.font) + ': ' + self.char

Здесь мы создали модели для Почерков и Символов. Модель символа состоит из:

а) char – текстовое отображение данного символа;

б) font – шрифт к которому относится данный символ, внешний ключ к модели Font. Являет собой отношение один-ко-многим: символ принадлежит только к одному почерку, а почерк, в свою очередь, содержит множество символов;

в) img – ссылка на изображение с этим символом;

г) x\_offset – смещение символа относительно левого края изображения в пикселях. Представлено полем PositiveIntegerField – целое положительное число;

д) y\_offset – смещение символа относительно верхнего края изображения в пикселях. Представлено полем PositiveIntegerField – целое положительное число;

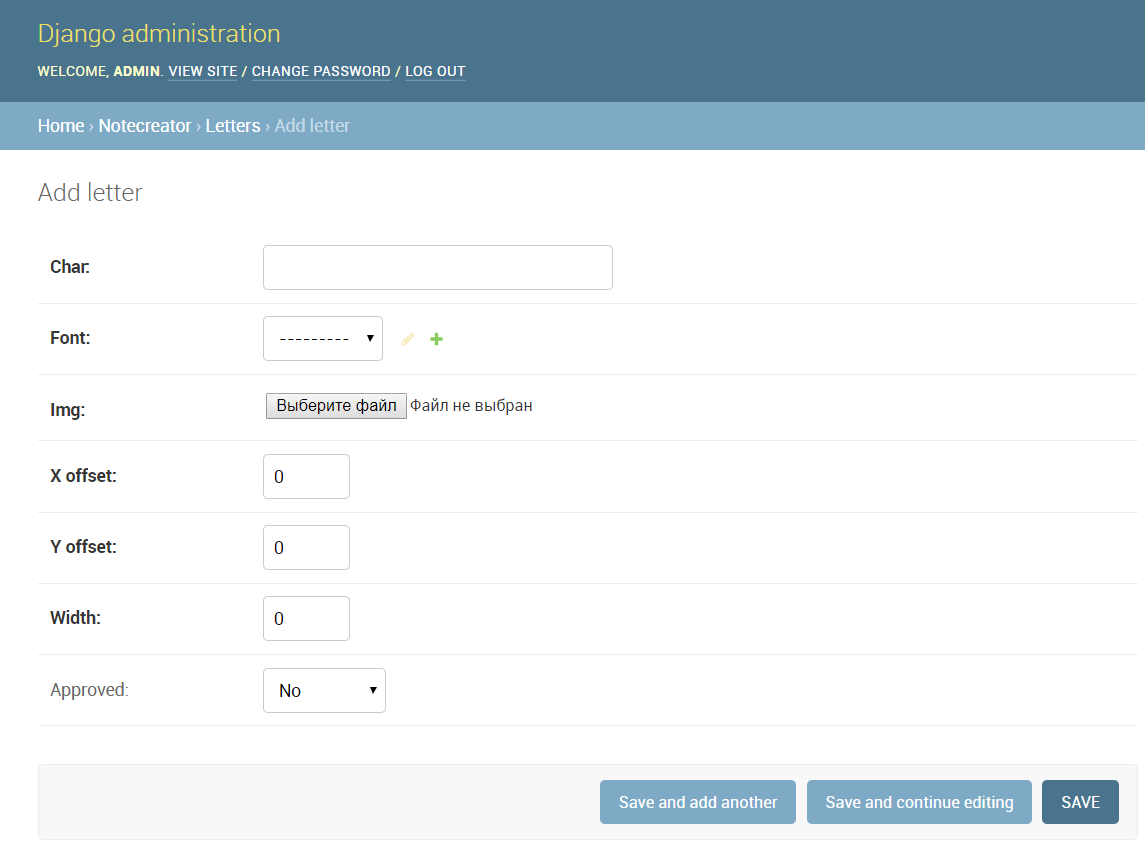
е) width – ширина символа. Представляет собой поле DecimalField – десятичное число с фиксированной точностью, использующее объект decimal.Decimal Python’а. Максимальное количество знаков (включая знаки после запятой) определяется атрибутом max\_digits, а количество знаков после запятой – атрибутом. Для хранения денежных чисел используется именно тип Decimal, а не Float, так как округление у последнего не всегда очевидно;

ж) approved – флаг указывающий на то, был ли данный символ разрешен администратором;

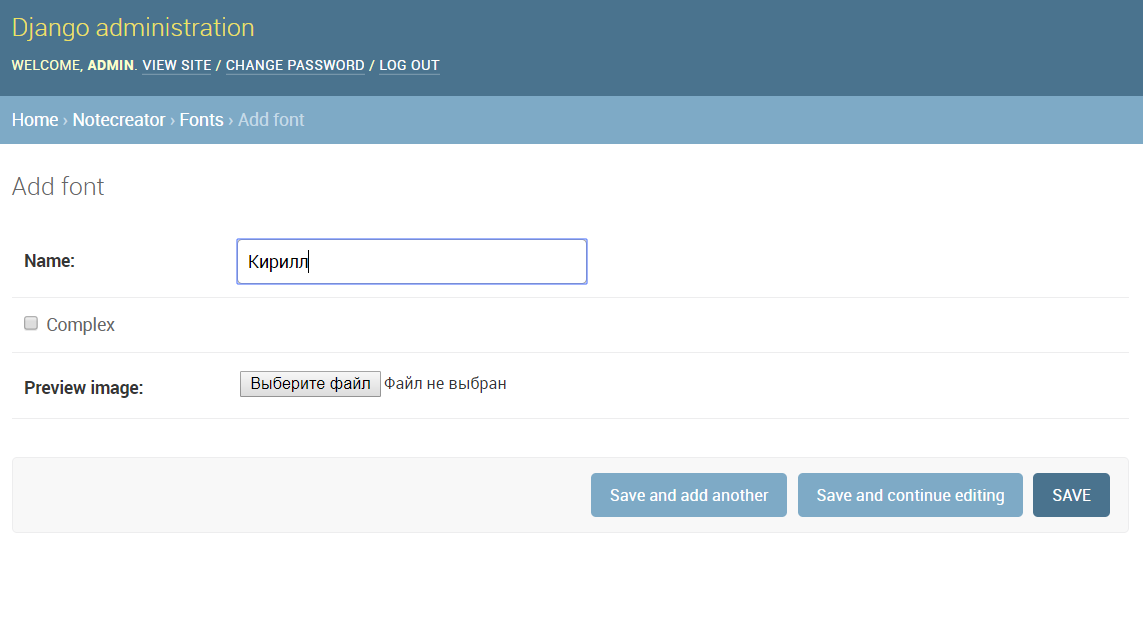
В модель Font добавили только 2 поля : name и preview\_image.

И наконец, оба класса содержат в себе метод \_\_str\_\_(). Он используется для того, чтобы сделать отображение объектов этого класса читаемым, а именно использовать поле имени этих классов при отображении.

На приведенных ниже рисунках 10 и 11 виден и процесс добавление нового символа и нового почерка в систему.



**Рисунок 10**

**Рисунок 11**

Можно теперь посмотреть, какой код сгенерировал Django для Mysql, который я выбрал для использования в этом проекте:

BEGIN;

--

--

-- Create model Font

--

--

CREATE TABLE "NoteCreator\_font"

("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT,

"name" varchar(10) NOT NULL, "complex" bool NOT NULL,

"preview\_image" varchar(100) NOT NULL);

--

--

-- Create model Letter

--

--

CREATE TABLE "NoteCreator\_letter"

("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT,

"char" varchar(1) NOT NULL, "img" varchar(100) NOT NULL,

"x\_offset" smallint NOT NULL, "y\_offset" smallint NOT NULL,

"width" smallint NOT NULL, "approved" bool NOT NULL,

"font\_id" integer NOT NULL REFERENCES "NoteCreator\_font" ("id") DEFERRABLE INITIALLY DEFERRED);

CREATE INDEX "NoteCreator\_letter\_font\_id\_150634f6" ON "NoteCreator\_letter" ("font\_id");

COMMIT;

Следующим шагом после создания почерков является созданиеявляется управление доступом к документам. Это позволит пользователям пред просматривать сгенерированный конспект в и временно его хранить, ожидая оформления заказа. Конечно, документ над которым работает пользователь должен сохраняться на время сессии визита пользователя. Для того, чтобы этого добиться используется внутренний фреймворк Django, который называется Session. Документ будет сохранен до тех пор, пока сессия будет активна или заказ не будет произведен.

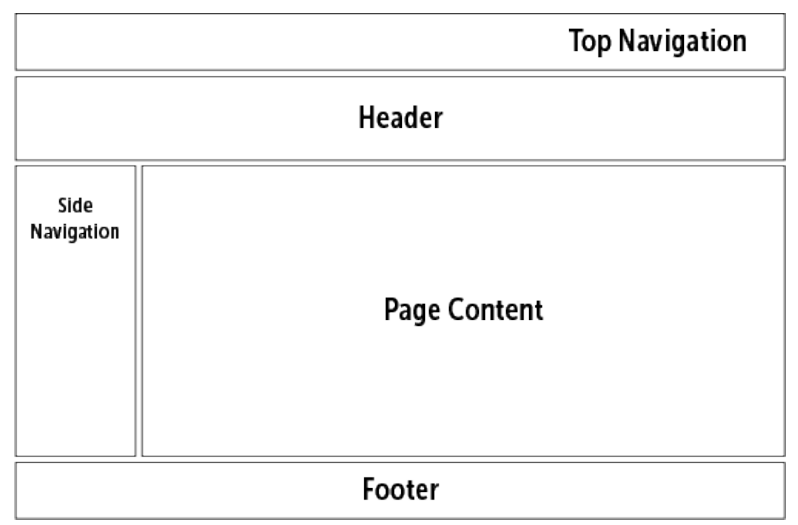
Django поставляет фреймворк, который поддерживает анонимные и неанонимные сессии и позволяющий хранить произвольные данные для каждого посетителя. Этот промежуточный слой хранит сессию, берет на себя отправку и принятие куки, содержащие id сессии. По умолчанию сессии хранятся в базе данных, но это можно изменить, обратившись к файлу settings.py. Эта функциональность доступна по умолчанию и готова к работе после создания проекта командой startproject.

Текущая сессия становится доступна в объекте request. Доступ к ней можно получить, использовав request.session, применяя стандартные для Python словари (dictionary) для хранения и извлечения информации.

## 4.3 Система шаблонов

Идея, стоящая за шаблонами примерно следующая: большая часть веб сайтов имеют одинаковый дизайн от страницы к странице. Например, в интернет-магазине, перемещаясь от товара к товару можно увидеть одну и ту же навигационную панель, шапку, «подвал». Такие статичные элементы стараются изолировать, чтобы, когда потребуется поменять, например, внешний вид навигационной панели, можно было отредактировать только один файл, а не сотню всех, где используется эта панель. Кроме того, большая часть сайтов со схожей функциональностью вообще имеют практически идентичное расположение элементов, так что было бы полезно иметь возможность использовать шаблон для базовых элементов, упомянутых ранее.

Схематичное изображение типичной верстки представлено на рисунке 12.



**Рисунок 12**

Эту идею развивает Django Template Language (DTL) – язык, позволяющий создавать динамический html, изолировать элементы, уменьшать количество одинаковых повторяющихся строк в html-файлах и другое. Он спроектирован с целью описания представления, не программной логики. Язык оперирует тэгами, поведение которых похоже на многие конструкции из программирования. Например, if – для проверки истинности, for – для циклов и т.д.[8]

Рассмотрим один из html-файлов моего проекта – base.html, находящийся внутри приложения SmartNote:

{% load static %}

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<title>{% block title %}Smart Note{% endblock %}</title>

<link href="{% static "css/base.css" %}" rel="stylesheet">

</head>

<body>

<div id="header">

<a href="/" class="logo">Smart Note</a>

</div>

<div id="subheader">

<div class="cart">

{% with total\_items=doc|length %}

{% if doc|length > 0 %}

Your document:

<a href="{% url "doc:doc\_detail" %}">

{{ total\_items }} item{{ total\_items|pluralize

}}, ${{ doc.get\_page\_count }}

</a>

{% else %}

Your document is empty.

{% endif %}

{% endwith %}

</div>

</div>

<div id="content">

{% block content %}

{% endblock %}

</div>

</body>

</html>

Как видно, это обычный HTML-код, но несколько дополненный.

Несколько раз встречающаяся конструкция {% block %} {% endblock %} означает, что содержимое этого блока может быть переопределено дочерними шаблонами.

Логика DTL используется для отображения информации о текущем документе – сколько в ней товаров и суммарная их стоимость.

## 4.4 Управление платежами

Для приема платежей онлайн требуется создать некий платежный шлюз, с помощью которого будут обрабатываться заказы пользователей. Использование такого шлюза позволяет перенести обработку самих платежей надежному третьему лицу. Их существует довольно много, в своем проекте я реализовал один – PayPal.

PayPal является одной из самых популярных платежных систем. По состоянию на 2012 год, PayPal работает в 190 странах, имеет больше 164 млн. зарегистрированных пользователей и работает с 24 национальными валютами. Касательно оплаты покупок важнейшей особенностью PayPal является предоставление гарантий безопасности как покупателю, так и продавцу.

PayPal предоставляет несколько методов интеграции. Стандартная интеграция представляет собой кнопку «Buy Now», которую можно видеть на огромном количестве веб-сайтов. Она представлена на рисунке 13.



**Рисунок 13**

Эта кнопка перенаправляет покупателя в PayPal для обработки платежа.

Для начала работы требуется создать PayPal аккаунт для бизнеса. Информация, использованная при регистрации, в дальнейшем еще понадобится при настройке системы.

Для работы с Django будет использоваться стороннее приложение под названием Django-paypal. Оно позволяет упростить интеграцию.

Сначала его необходимо установить при помощи PIP, а затем включить в список используемых приложений в файле settings.py проекта. Кроме того, в этом же файле прописывается дополнительная информация:

а) PAYPAL\_RECEIVER\_EMAIL – электронный адрес, на который регистрировался PayPal аккаунт

б) PAYPAL\_TEST – булева переменная, отображающая включено или выключено должно быть Sandbox. Это окружение, предназначенное для тестирования платежей, до того, как переходить к работе с реальными деньгами. Ее значение выставляется True.[7]

Далее создается новое приложение payment, которое будет управлять платежами. Внутри файла views.py этого приложения следующий код:

def payment\_process(request):

order\_id = request.session.get('order\_id')

order = get\_object\_or\_404b(Order, id=order\_id)

host = request.get\_host()

paypal\_dict = {

'business': settings.PAYPAL\_RECEIVER\_EMAIL,

'amount': '%.2f' %

order.get\_total\_cost().quantize(Decimal('.01')),

'item\_name': 'Order {}'.format(order.id),

'invoice': str(order.id),

'currency\_code': 'EUR',

'notify\_url': 'http://{}{}'.format(host, reverse('paypalipn')),

'return\_url': 'http://{}{}'.format(host,

reverse('payment:done')),

'cancel\_return': 'http://{}{}'.format(host,

reverse('payment:canceled')),

}

form = PayPalPaymentsForm(initial=paypal\_dict)

return render(request, 'payment/process.html', {'order': order,

'form':form})

Здесь генерируется собственная кнопка «Buy Now» для оплаты заказа. Сначала получаем заказ order\_id из сессии, затем получаем объект Order для нужного ID и, наконец, строим форму PayPalPaymentsForm. Она будет включать следующие поля:

а) business – PayPal аккаунт продавца для проведения платежа. Здесь используется значение PAYPAL\_RECEIVER\_EMAIL

б) amount – общая сумма к оплате

в) item\_name – название продаваемого товара. Используется ID заказа, так как в нем может содержаться несколько товаров

г) invoice – номер выставляемого счета

д) currency\_code – код валюты для платежа. Должен использоваться тот, который указан в аккаунте PayPal

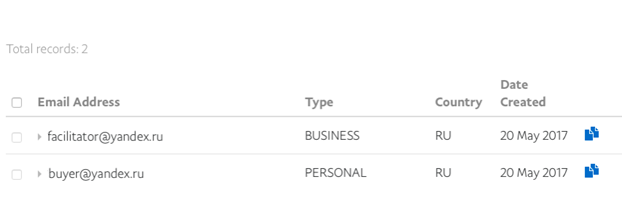
е) notify\_url – адрес, на который PayPal будет отправлять запрос на уведомление

ж) return\_url – адрес, на который пользователь будет переведен в случае успешного платежа

з) cancel\_return – адрес, на который пользователь будет переведен в случае возникновения каких-либо ошибок при проведении платежа или его отмене

Эта форма является стандартной формой со скрытыми полями, и пользователь будет видеть лишь кнопку «Buy Now», по нажатию на которую форма будет отправлена PayPal при помощи метода POST.

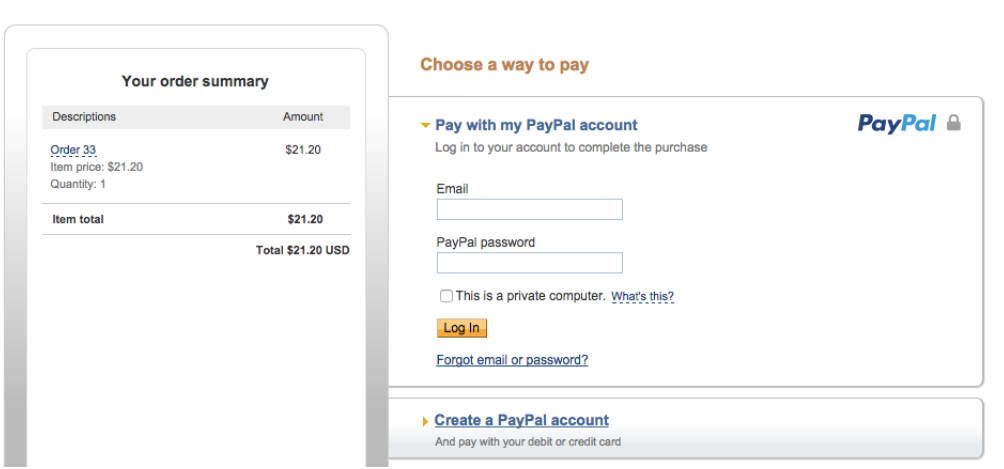
Пользоваться тестовым окружением Sandbox можно перейдя по ссылке developer.paypal.com. Для моего аккаунта там автоматически были созданы 2 учетные записи – одна покупателя, одна посредника, как показано на рисунке 14.



**Рисунок 14**

Раскрыв одну из них и перейдя к Profile, можно увидеть детали учетной записи, в том числе информацию о тестовой кредитной карте и балансе. Эти тестовые аккаунты могут быть использованы для оплаты на нашем сайте при помощи Sandbox.

По итогу, нажав на кнопку «Buy Now», покупатель будет перенаправлен в PayPal и увидит следующую страницу (рисунок 15):



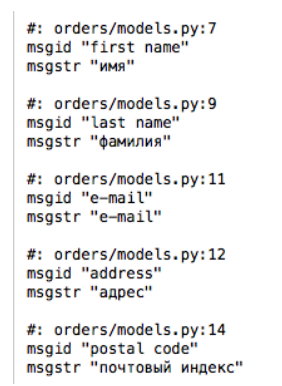
**Рисунок 15**

А после оплаты перенаправлен обратно в магазин по заданному URL. Однако, после прохождения процедуры оплаты наш сервер не получит уведомление о статусе платежа. Вернее, PayPal не сможет его отправить, так как проект запускается на локальном хосте с IP 127.0.0.1, который публично недоступен.

## 4.5 Локализация и интернационализация

Django предоставляет полную поддержку интернационализации и локализации. Он позволяет переводить приложение на несколько языков и поддерживает специфические форматы дат, времени и временных зон. Дадим определения понятиям интернационализация и локализация. Интернационализация (часто используется аббревиатура i18n) – это процесс адаптации программного обеспечения для потенциального использования на разных языках и в разных странах, при этом он не привязан к определенному языку или стране. Локализация (аббревиатура l10n) – это процесс непосредственно перевода программного обеспечения и адаптации для конкретной страны. Сам Django переведен на более чем 50 языков.

Модуль интернационализации дает возможность помечать строки и в коде, и в внутри темплейтов. Он полагается на использование набора средств под GNU gettext для генерации и управления файлами сообщений. Файлы сообщений – это просто текст, представляющий язык. Они содержат в себе все или часть переводимых строк, найденных в приложении и их соответствующий перевод на определенный язык. Эти файлы имеют расширение .po. Пример такого файла приведен на рисунке 16:



**Рисунок 16**

По завершению перевод, файлы сообщений компилируются, чтобы предоставлять быстрый доступ к переведенным строкам. Такие файлы имеют расширение .mo. Django предоставляет довольно много настроек для интернационализации. Ниже в таблице 2 приведены наиболее актуальные из них:

**Таблица 2 - Настройки локализации и интернационализации**

|  |  |
| --- | --- |
| USE\_I18N | Булева переменная. Указывает, включен ли модуль интернационализации или нет. По умолчанию установлено значение True |
| USE\_L10N | Булева переменная. Указывает, включена ли локализация. Когда она включена, используются специализированные форматы даты и времени. По умолчанию установлено значение False |
| USE\_TZ | Булева переменная. Указывает, используются ли часовые пояса. По умолчанию установлено значение True |
| LANGUAGE\_CODE | Код языка проекта по умолчанию. Использует стандартизированный формат, например, en-gb для британского английского, ru для русского и так далее. Использование этой опции предполагает, что интернационализация влючена |
| LANGUAGES | Кортеж (tuple), содержащий доступные языки для проекта. Они, в свою очередь, тоже представлены кортежем, состоящим из кода языка и названия языка |
| LOCALE\_PATHS | Список директорий, где Django следует искать файлы сообщений с переводами для проекта |
| TIME\_ZONE | Строка, представляющая временную зону для проекта |

Используя эти параметры для проекта устанавливаются два языка – английский и русский:

# Internationalization

# https://docs.djangoproject.com/en/1.8/topics/i18n/

LANGUAGE\_CODE = 'en'

from django.utils.translation import gettext\_lazy as \_

LANGUAGES = (

('en', \_('English')),

('ru', \_('Russian')),

)

LOCALE\_PATHS = (

os.path.join(BASE\_DIR, 'locale/'),

)

TIME\_ZONE = 'UTC'

USE\_I18N = True

USE\_L10N = True

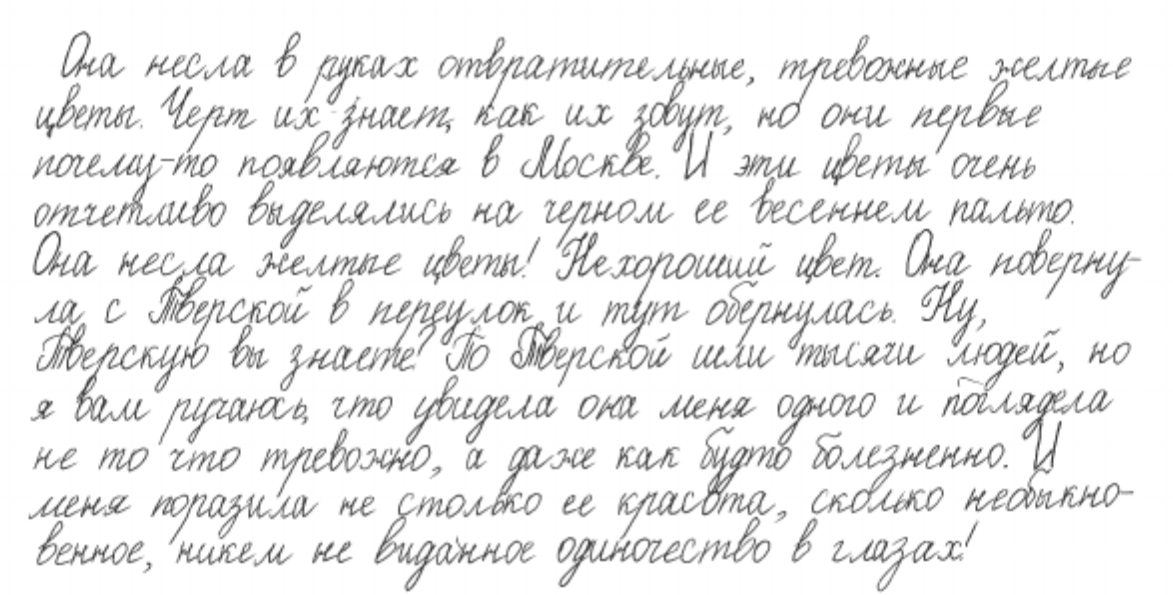
USE\_TZ = True

В Django также существуют «ленивые» (lazy) версии всех функций, использующихся в переводе. При их использовании, строки переводятся только когда к ним пытаются получить доступ, а не при вызове функции. Такой подход полезно использовать при переводе отдельных модулей.

# 5. Экономическая часть

Темой Дипломного проекта является разработка сайта электронной коммерции. Как частный ее случай был выбран интернет ресурс с платными услугами. Тематика – Создание псевдо рукописного текста для последующего использования в качестве конспектов.

На данный момент в Русскоязычном сегменте практически невозможно найти услуги по переписыванию конспекта. Эта услуга востребована у студентов и учащихся старших классов в школах. За символическую плату можно получить псевдо рукописный текст, который очень сложно отличить от оригинала (см. Рисунок 17). Стоит отметить, что подобная услуга может пользоваться спросом и в странах Европы.



**Рисунок 17**

В дальнейшем, если потребуется, возможно было бы сделать для этого сервиса регистрацию форму обратной связи и форму загрузки персональных почерков.

Сама разработка сайта не является особенно прибыльным занятием для человека, не имеющего большого практического опыта в этой области. И даже в таком случае, платят в основном за уникальный дизайн, который будет сделан вручную, а не сгенерирован из множества доступных шаблонов.

## 5.1 Расчет длительности этапа разработки

Расчет длительности этапа разработки был произведен по формуле:

Результаты сведены в таблицу 3:

**Таблица 3 – Длительность этапа разработки**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Наименование работы** | **Длительность работы , ед. t** | | |
| **tmin, ч** | **tmax, ч** | **t0, ч** |
| 1 | Предпроектное исследование | 15 | 25 | 19 |
| 2 | Проектирование | 50 | 80 | 62 |
| 3 | Разработка | 170 | 250 | 202 |
| 4 | Отладка и тестирование | 20 | 40 | 28 |
| 5 | Оформление пояснительной записки | 15 | 25 | 19 |
| **ИТОГО:** | | **270** | **420** | **330** |

Продолжительность работ (ед. t) было решено измерять в человеко-часах. Ставка заработной платы будет использоваться, соответственно, за час.

## 5.2 Расчет заработной платы разработчика

Разработкой сайта занимается один человек, месячная заработная плата принимается 800 руб (з/п инженера). Для определения часовой ставки ЗП необходимо разделить заработную плату за месяц на количество рабочих часов в месяце (168 часов). Таким образом, получаем часовую ставку 4,76 руб./час.

На основе данных о трудоемкости выполняемых работ и часовой ставки определим расходы на заработную плату и отчислений на страховые взносы на обязательное социальное, пенсионное и медицинское страхование. Расходы на основную заработную плату определяются по формуле:

где Зосн.з/пл - расходы на основную заработную плату исполнителей (руб.);

k – количество исполнителей (в данном случае разработкой занимается только 1 человек);

Ti – время, затраченное i-м исполнителем на проведение работы (в данном случае в часах);

Ci– ставка i-го исполнителя (руб./час). Расходы на дополнительную заработную плату исполнителей определяются по формуле:

Где Здоп.з/пл – расходы на дополнительную заработную плату исполнителей (руб.);

Ндоп – норматив дополнительной заработной платы (%). Этот норматив принимаем равным 14%.[9]

Отчисления на страховые взносы на обязательное социальное, пенсионное и медицинское страхование с основной и дополнительной заработной платы исполнителей определяются по формуле:

Где Зсоц - отчисления на социальные нужды с заработной платы (руб.);

Нсоц – норматив отчислений на страховые взносы на обязательное социальное, пенсионное и медицинское страхование (%). Этот норматив принимаем равным 30%.

## 5.3 Расчет амортизационных отчислений основных средств

**Таблица 4 – Стоимость оборудования**

|  |  |  |
| --- | --- | --- |
| **Наименование оборудования** | **Описание** | **Стоимость (с НДС), руб.** |
| Macbook Air 2014 | Ноутбук [Intel Core i5 1.3-2.7 GHz, SSD 128 GB, RAM 8 GB] | 2000 |

**Таблица 5 – Стоимость программного обеспечения**

|  |  |  |
| --- | --- | --- |
| **Наименование оборудования** | **Описание** | **Стоимость, руб.** |
| macOS 10.12 | Операционная система | Распространяется бесплатно |
| IDLE 3.5.2 | Среда разработки | Распространяется бесплатно |
| Sublime Text 3 | Текстовый редактор | Распространяется бесплатно |

Для разработки сайта использовались основные средства, амортизационные отчисления по которым определяются как:

где Аi – амортизационные отчисления за год по i-му основному средству

(руб.);

Цп.н.i – первоначальная стоимость i-го основного средства (руб.);

Нai - годовая норма амортизации i-го основного средства (%).

Ноутбук, использовавшийся в разработке был приобретен в ноябре 2014 года. Эксплуатироваться компьютер начал в месяц приобретения. Указанное основное средство входит во вторую амортизационную группу, как имущество со сроком полезного использования свыше 2 лет до 3 лет включительно.

Соответственно, срок полезного использования персонального компьютера устанавливаем в количестве 3 лет, т.е. получаем 1/3 ∗ 100% = 33,3% .

Величина амортизационных отчислений по i-му основному средству, используемом при работе над Дипломным проектом, определяется по формуле:

где АiДП – амортизационные отчисления по i-му основному средству, используемому в работе над ДП (руб.);

ТiДП - время, в течение которого использовалось i-ое основное средство (мес.) (ДП пишется 6 месяцев).

## 5.4 Расчет затрат на сырье и материалы

Единственным видом расходных материалов, использовавшихся при создании ДП, является бумага для печати работы. Объем ДП, с учётом перепечатки отдельных компонентов, составляет 100 листов, цена печати одного листа при односторонней печати 0,5 руб. Рассчитаем по следующей формуле:

Где Зм – затраты на сырье и материалы (руб.);

L – индекс вида сырья или материала;

Gl – норма расхода l-того материала на единицу продукции (ед.);

Цl – цена приобретения единицы l-го (руб./ед.);

Нт.з – норма транспортно-заготовительных расходов (%). При выполнении расчетов в ДП норму транспортно-заготовительных расходов (Нт.з) принимаем равной 10%.[10]

Результаты вычислений, по указанным выше формулам:

**Таблица 6 – Результаты вычислений**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Зосн.з/пл , руб | Здоп.з/пл , руб | Зсоц , руб | Аi, руб | AiДП , руб | Зм, руб |
| 1571,46 | 220 | 537,43 | 660 | 330 | 55 |

**Таблица 7 – Смета затрат на Дипломный проект**

|  |  |  |
| --- | --- | --- |
| **№** | **Наименование статьи** | **Сумма , руб** |
| 1 | Расходы на оплату труда | 1791,46 |
| 2 | Отчисления на социальные нужды | 537,43 |
| 3 | Амортизационные отчисления | 330 |
| 4 | Сырье и материалы | 55 |
| **ИТОГО затрат** | | **2713,89** |

# 5.5 Вывод

Расходы на разработку дипломного проекта составляют 2713,89 руб. В данном разделе проводится анализ себестоимости разработки проекта. В рамках составления технико-экономического обоснования выполнены следующие расчеты: оценка величины заработной платы и социальных отчислений участников исследования, определение величины амортизационных отчислений используемых основных средств, расчет совокупной величины затрат, связанных с выполнением работы.

Представленные выводы могут быть использованы при разработке последующих проектов для снижения издержек.

Большое время разработки проекта обусловлено моей неопытностью и необходимости обучаться в процессе разработки. В дальнейшем подобные проекты можно будет разрабатывать на порядок быстрее, что значительно уменьшит себестоимость и сделает разработку конкурентоспособной. На данном же этапе, этот проект может рассматриваться только как работа над портфолио.

# Заключение

По результатам работы над дипломным проектом был разработан веб-сайт электронной коммерции и продемонстрирован альтернативный подход к веб- разработке. Была реализована следующая функциональность:

а) Панель администрирования для управления почерками, добавления новых почерков, изменение их конфигурации;

б) Функционал для персональной настройки почерка для собственных нужд. Возможность выбора размера бумаги и расстояния между строками;

в) Возможность предпросмотра конспектов;

г) Возможность скачать документ с полным конспектом;

д) Интеграция платежной системы PayPal. Это одна из наиболее популярных и надежных платежных систем, работающая со многими национальными валютами;

е) Локализация. Изначально сайт разрабатывался на английском языке, но в последствии была добавлена возможность смены языка на русский.

Помимо этого, было приведено описание понятия электронной коммерции: история возникновения, актуальность, преимущества и недостатки, потенциал развития. Проведена оценка экономической эффективности разработки и подсчитана себестоимость.

1. Советов Б.Я., Цехановский В.В., Чертовской В.Д. Базы данных: теория и практика. Учебник М.: «Юрайт», 2012. – 464 с. Гриф УМО по университетскому политехническому образованию.

2. Филипп Гуо Python is Now the Most Popular Introductory Teaching Language at Top U.S. Universities // Исследовательская организация Communications of the ACM – 2014. – 7 июля [Электронный ресурс]. URL: <https://cacm.acm.org/blogs/blog-cacm/176450-python-is-now-the-most-popularintroductory-teaching-language-at-top-u-s-universities/fulltext> (дата доступа: 22.04.2018)

3. Nigel George Mastering Django: Core. Chicago: GNW Independent Publishing; 1st ed., 2016

4. Amazon Web Services. wikipedia. [Электронный ресурс] Режим доступа: <https://ru.wikipedia.org/wiki/Amazon_Web_Services>. – Дата доступа: 01.05.2018.

5. Облачные продукты. amazon. [Электронный ресурс] Режим доступа: <https://aws.amazon.com/ru/products/> . – Дата доступа: 01.05.2018.

6. Amazon AWS. actualtraffic. [Электронный ресурс] Режим доступа: <http://actualtraffic.ru/site/aws-amazon/> – Дата доступа: 01.05.2018.

7. AWS Elastic Beanstalk. amazon. [Электронный ресурс] Режим доступа: <https://aws.amazon.com/ru/elasticbeanstalk/> – Дата доступа: 01.05.2018.

8. Daniel Roy Greenfield, Audrey Roy Greenfield Two Scoops of Django: Best Practices for Django 1.8. Chicago: Two Scoops Press; 3rd ed., 2015

9. Астреина Л. А. Экономика предприятия: методические указания по одноименной дисциплине / Л. А. Астреина, Ю. А. Гарайбех, М. В. Чигирь. – СПб.: Изд-во СПбГЭТУ «ЛЭТИ». 2004. — 65 c.

10. Семенова В. М. Экономика предприятия: учебник для вузов / В. М. Семенова. — М.: Центр экономики и маркетинга. 2004. — 120 с.