*Федеральное государственное бюджетное образовательное учреждение*

*высшего профессионального образования*

|  |  |
| --- | --- |
| Gerb-BMSTU_01 | ***«Московский государственный технический университет   имени Н.Э. Баумана»***  ***(МГТУ им. Н.Э. Баумана)*** |

Факультет «Информатика и системы управления»

Кафедра «Программное обеспечение ЭВМ и информационные технологии»

**РАСЧЁТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовому проекту по курсу:**

«Базы данных»

**на тему:**

«Web-приложение для кредитного скоринга»

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Тагер А.Д.

(Подпись, дата) (И.О.Фамилия)

Руководитель курсовой работы \_\_\_\_\_ **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Рогозин О.В.

(Подпись, дата) (И.О.Фамилия)

Москва, 2016 г.

Оглавление

**Введение**4

**Аналитическая раздел**5

1. Анализ предметной области 5

2 Определение требований к структуре5

2.1 Выделение сущностей…………………………………………..5

2.2 Определение способа использования данных…………………7

2.3 Нормализация отношений………………………………………7

2.3.3. Первая нормальная форма8

2.3.3. Вторая нормальная форма9

2.3.3. Третья нормальная форма10

2.3.3 Нормальная форма Бойса-Кодда 12

2.3.3 Четвертая нормальная форма15

2.3.3. Пятая нормальная форма16

2.3.3 Доменно-ключевая нормальная форма 17

2.3.3 Шестая нормальная форма 18

**Конструкторский раздел**20

1. Определение связей между таблицами 20

2. Определение ограничений на данные21

3. Нормализация базы данных24

3.1 Первая нормальная форма25

3.2 Вторая нормальная форма25

3.3. Третья нормальная форма25

4. Структура программы 26

**Технологический раздел**27

1. Выбор языка программирования и среды разработки 27

2. Выбор web-фреймворка 27

3. Выбор СУБД 28

4. Выбор технологий для стилизации клиентской части приложения28

5. Взаимодействие с базой данных 28

6. Выбор предсказательной модели 31

7. Интерфейс программы32

8. Технические требования37

**Заключение**38

**Список литературы**39

**Введение**

В настоящее время в мире наблюдается тенденция использования в приложениях реляционных баз данных для хранения и обработки структурированной информации. Одним из примеров таких приложений можно назвать Web приложения. Web приложениями называются приложения, работа с которыми осуществляется средствами веб - интерфейса. В данной работе целью является создание такого Web приложения.

Темой разрабатываемого приложения стало предсказание шансов на получение кредита, используя алгоритм машинного обучения. Машинное обучение - обширный подраздел искусственного интеллекта, изучающий методы построения алгоритмов, способных обучаться. Один из этих алгоритмов будет применяться в приложении для кредитного скоринга. Скоринг – используемая банками система оценки клиентов, в основе которой заложены статистические методы. Как правило, это компьютерная программа, куда вводятся данные потенциального заемщика. В ответ выдается результат – стоит ли предоставлять ему кредит.

Приложение осуществляет работу с базой данных, содержащую информацию о клиентах, запрашиваемых кредит в банке. Приложение дает возможность пользователю изучить информацию о модели, а так же оценить свои шансы на получение кредита. Так же приложение должно содержать администраторскую область, которая дает возможность администраторам просмотра информации, об уже внесенных в базу клиентах, а так же добавить в базу нового клиента.

**Аналитический раздел**

## 1. Анализ предметной области

Разрабатываемое Web приложение должно предоставлять пользователю удобный и понятный интерфейс для проверки своих шансов на получение кредита, а так же администратору для просмотра информации, об уже внесенных клиентах и добавления новых клиентов в базу. Приложение должно обладать следующей функциональностью:

* Возможность просмотра основной информации о предсказательной модели.
* Возможность оценивания шансов на получение кредита.
* Возможность просмотра информации, об уже внесенных клиентах для администратора.
* Возможность внесения информации о новых клиентах в базу для администратора.

Стоит отметить, что в реальной жизни, для получения полной информации о клиенте, банк делает запрос в бюро кредитных историй. Бюро кредитных историй — компания, оказывающая в соответствии с законодательством услуги по формированию, обработке и хранению кредитныхисторий, а также по предоставлению кредитных отчетов. Так как реализация запроса в курсовом проекте невозможна, было принято решение имитировать бюро кредитных историй, и для этого были созданы таблицы, содержащие информацию о предыдущих кредитах клиентов.

## 2. Определение требований к структуре данных

### 1 Выделение сущностей

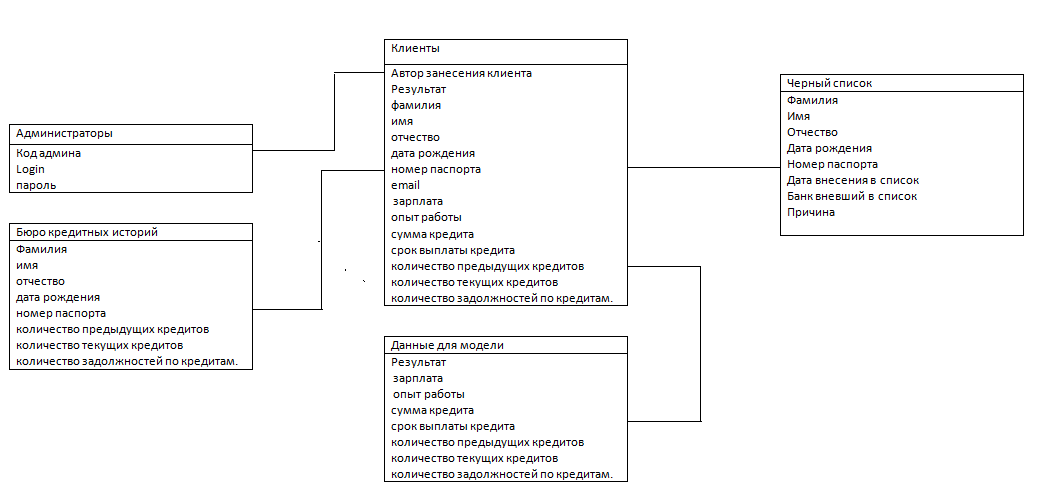
При анализе предметной были выделены следующие сущности:

* Клиенты
* Данные для модели
* Администраторы
* Бюро кредитных историй
* Черный список

В данной таблице перечислены атрибуты каждой разработанной сущности.

|  |  |
| --- | --- |
| **Сущность** | **Атрибуты** |
| Клиенты | Результат, фамилия, имя, отчество, дата рождения, номер паспорта, email, место работы, зарплата, опыт работы, сумма кредита, срок выплаты кредита, количество предыдущих кредитов, количество текущих кредитов, количество задолжностей по кредитам. |
| Данные для модели | Результат, зарплата, опыт работы, сумма кредита, срок выплаты кредита, количество предыдущих кредитов, количество текущих кредитов, количество задолжностей по кредитам. |
| Администраторы | Login, пароль |
| Бюро кредитных историй | Фамилия, имя, отчество, дата рождения, номер паспорта, количество предыдущих кредитов, количество текущих кредитов, количество задолжностей по кредитам. |
| Черный список | Фамилия, имя, отчество, дата рождения, номер паспорта, дата внесения в список, банк, внесший в список, причина |

*Таблица 1 (Сущности и их атрибуты).*

**

*Рисунок 1(Инфологическая модель)*

## 2.2 Определение способа использования данных

## Для данной системы существует две категории пользователей:

## Пользователь приложения. Данная категория пользователей имеет доступ к веб интерфейсу. Но так как информация, находящаяся в базе данных является конфиденциальной, доступа к ней она не имеет.

* Администратор базы данных. Данный категория пользователей также имеет все правда предыдущей категории, а также доступ к базе данных. Таким образом, она может добавлять и просматривать информацию о клиентах банка.

## 2.3 Нормализация отношений

Нормальная форма — свойство отношения в реляционной модели данных, характеризующее его с точки зрения избыточности, потенциально приводящей к логически ошибочным результатам выборки или изменения данных. Нормальная форма определяется как совокупность требований, которым должно удовлетворять отношение.

Процесс преобразования отношений базы данных к виду, отвечающему нормальным формам, называется **нормализацией**. Нормализация предназначена для приведения структуры БД к виду, обеспечивающему минимальную логическую избыточность, и не имеет целью уменьшение или увеличение производительности работы или же уменьшение или увеличение физического объёма базы данных. Конечной целью нормализации является уменьшение потенциальной противоречивости хранимой в базе данных информации

### Первая нормальная форма (1NF)

Отношение находится в 1НФ, если все его атрибуты являются простыми, все используемые домены должны содержать только скалярные значения. Не должно быть повторений строк в таблице.  
  
Например, есть таблица «Автомобили»:

|  |  |
| --- | --- |
| Фирма | Модели |
| BMW | M5, X5M, M1 |
| Nissan | GT-R |

Нарушение нормализации 1НФ происходит в моделях BMW, т.к. в одной ячейке содержится список из 3 элементов: M5, X5M, M1, т.е. он не является атомарным. Преобразуем таблицу к 1НФ:

|  |  |
| --- | --- |
| Фирма | Модели |
| BMW | M5 |
| BMW | X5M |
| BMW | M1 |
| Nissan | GT-R |

### Вторая нормальная форма (2NF)

Отношение находится во 2НФ, если оно находится в 1НФ и каждый не ключевой атрибут неприводимо зависит от Первичного Ключа (ПК).  
  
Неприводимость означает, что в составе потенциального ключа отсутствует меньшее подмножество атрибутов, от которого можно также вывести данную функциональную зависимость.  
  
Например, дана таблица:

|  |  |  |  |
| --- | --- | --- | --- |
| Модель | Фирма | Цена | Скидка |
| M5 | BMW | 5500000 | 5% |
| X5M | BMW | 6000000 | 5% |
| M1 | BMW | 2500000 | 5% |
| GT-R | Nissan | 5000000 | 10% |

Таблица находится в первой нормальной форме, но не во второй. Цену машины зависит от модели и фирмы. Скидка зависят от фирмы, то есть зависимость от первичного ключа неполная. Исправляется это путем декомпозиции на два отношения, в которых не ключевые атрибуты зависят от ПК.

|  |  |  |
| --- | --- | --- |
| Модель | Фирма | Цена |
| M5 | BMW | 5500000 |
| X5M | BMW | 6000000 |
| M1 | BMW | 2500000 |
| GT-R | Nissan | 5000000 |

|  |  |
| --- | --- |
| Фирма | Скидка |
| BMW | 5% |
| Nissan | 10% |

### Третья нормальная форма (3NF)

Отношение находится в 3НФ, когда находится во 2НФ и каждый не ключевой атрибут нетранзитивно зависит от первичного ключа. Проще говоря, второе правило требует выносить все не ключевые поля, содержимое которых может относиться к нескольким записям таблицы в отдельные таблицы.  
  
Рассмотрим таблицу:

|  |  |  |
| --- | --- | --- |
| Модель | Магазин | Телефон |
| BMW | Риал-авто | 87-33-98 |
| Audi | Риал-авто | 87-33-98 |
| Nissan | Некст-Авто | 94-54-12 |

Таблица находится во 2НФ, но не в 3НФ.  
В отношении атрибут «Модель» является первичным ключом. Личных телефонов у автомобилей нет, и телефон зависит исключительно от магазина.  
Таким образом, в отношении существуют следующие функциональные зависимости: Модель → Магазин, Магазин → Телефон, Модель → Телефон.  
Зависимость Модель → Телефон является транзитивной, следовательно, отношение не находится в 3НФ.  
В результате разделения исходного отношения получаются два отношения, находящиеся в 3НФ:

|  |  |
| --- | --- |
| Риал-авто | 87-33-98 |
| Риал-авто | 87-33-98 |
| Некст-Авто | 94-54-12 |

|  |  |
| --- | --- |
| Модель | Магазин |
| BMW | Риал-авто |
| Audi | Риал-авто |
| Nissan | Некст-Авто |

### Нормальная форма Бойса — Кодда (BCNF)

Определение 3НФ не совсем подходит для следующих отношений:  
1) отношение имеет две или более потенциальных ключа;  
2) два и более потенциальных ключа являются составными;  
3) они пересекаются, т.е. имеют хотя бы один атрибут.  
  
Для отношений, имеющих один потенциальный ключ (первичный), НФБК является 3НФ.  
  
Отношение находится в НФБК, когда каждая нетривиальная и неприводимая слева функциональная зависимость обладает потенциальным ключом в качестве детерминанта.  
  
Предположим, рассматривается отношение, представляющее данные о бронировании стоянки на день:

|  |  |  |  |
| --- | --- | --- | --- |
| Номер стоянки | Время начала | Время окончания | Тариф |
| 1 | 09:30 | 10:30 | Бережливый |
| 1 | 11:00 | 12:00 | Бережливый |
| 1 | 14:00 | 15:30 | Стандарт |
| 2 | 10:00 | 12:00 | Премиум-В |
| 2 | 12:00 | 14:00 | Премиум-В |
| 2 | 15:00 | 18:00 | Премиум-А |

Тариф имеет уникальное название и зависит от выбранной стоянки и наличии льгот, в частности:

* «Бережливый»: стоянка 1 для льготников
* «Стандарт»: стоянка 1 для не льготников
* «Премиум-А»: стоянка 2 для льготников
* «Премиум-B»: стоянка 2 для не льготников.

Таким образом, возможны следующие составные первичные ключи: {Номер стоянки, Время начала}, {Номер стоянки, Время окончания}, {Тариф, Время начала}, {Тариф, Время окончания}.  
  
Отношение находится в 3НФ. Требования второй нормальной формы выполняются, так как все атрибуты входят в какой-то из потенциальных ключей, а не ключевых атрибутов в отношении нет. Также нет и транзитивных зависимостей, что соответствует требованиям третьей нормальной формы. Тем не менее, существует функциональная зависимость Тариф → Номер стоянки, в которой левая часть (детерминант) не является потенциальным ключом отношения, то есть отношение не находится в нормальной форме Бойса — Кодда.  
  
Недостатком данной структуры является то, что, например, по ошибке можно приписать тариф «Бережливый» к бронированию второй стоянки, хотя он может относиться только к первой стоянки.  
  
Можно улучшить структуру с помощью декомпозиции отношения на два и добавления атрибута. И**меет льготы**, получив отношения, удовлетворяющие НФБК (подчёркнуты атрибуты, входящие в первичный ключ.):  
  
**Тарифы**

|  |  |  |
| --- | --- | --- |
| Тариф | Номер стоянки | Имеет льготы |
| Бережливый | 1 | Да |
| Стандарт | 1 | Нет |
| Премиум-А | 2 | Да |
| Премиум-В | 2 | Нет |

**Бронирование**

|  |  |  |
| --- | --- | --- |
| Тариф | Время начала | Время окончания |
| Бережливый | 09:30 | 10:30 |
| Бережливый | 11:00 | 12:00 |
| Стандарт | 14:00 | 15:30 |
| Премиум-В | 10:00 | 12:00 |
| Премиум-В | 12:00 | 14:00 |
| Премиум-А | 15:00 | 18:00 |

### Четвёртая нормальная форма (4NF)

Отношение находится в 4НФ, если оно находится в НФБК и все нетривиальные многозначные зависимости фактически являются функциональными зависимостями от ее потенциальных ключей.  
  
В отношении R (A, B, C) существует **многозначная зависимость** R.A -> -> R.B в том и только в том случае, если множество значений B, соответствующее паре значений A и C, зависит только от A и не зависит от С.  
  
Предположим, что рестораны производят разные виды пиццы, а службы доставки ресторанов работают только в определенных районах города. Составной первичный ключ соответствующей переменной отношения включает три атрибута: {Ресторан, Вид пиццы, Район доставки}.  
  
Такая переменная отношения не соответствует 4НФ, так как существует следующая многозначная зависимость:  
{Ресторан} → {Вид пиццы}  
{Ресторан} → {Район доставки}  
  
То есть, например, при добавлении нового вида пиццы придется внести по одному новому кортежу для каждого района доставки. Возможна логическая аномалия, при которой определенному виду пиццы будут соответствовать лишь некоторые районы доставки из обслуживаемых рестораном районов.  
  
Для предотвращения аномалии нужно декомпозировать отношение, разместив независимые факты в разных отношениях. В данном примере следует выполнить декомпозицию на {Ресторан, Вид пиццы} и {Ресторан, Район доставки}.  
  
Однако, если к исходной переменной отношения добавить атрибут, функционально зависящий от потенциального ключа, например цену с учётом стоимости доставки ({Ресторан, Вид пиццы, Район доставки} → Цена), то полученное отношение будет находиться в 4НФ и его уже нельзя подвергнуть декомпозиции без потерь.

### Пятая нормальная форма (5NF)

Отношения находятся в 5НФ, если оно находится в 4НФ и отсутствуют сложные зависимые соединения между атрибутами.  
Если «Атрибут зависит от «Атрибута\_2», а «Атрибут\_2» в свою очередь зависит от «Атрибута\_3», а «Атрибут\_3» зависит от «Атрибута\_1», то все три атрибута обязательно входят в один кортеж.  
  
Это очень жесткое требование, которое можно выполнить лишь при дополнительных условиях. На практике трудно найти пример реализации этого требования в чистом виде.  
  
Например, некоторая таблица содержит три атрибута «Поставщик», «Товар» и «Покупатель». Покупатель1 приобретает несколько Товаров у Поставщика1. Покупатель 1 приобрел новый Товар у Поставщика2. Тогда в силу изложенного выше требования I поставщик1 обязан поставлять Покупателю1 тот же самый новый Товар, а Поставщик2 должен поставлять Покупателю 1, кроме нового Товара, всю номенклатуру Товаров Поставщика1. Этого на практике не бывает. Покупатель свободен в своем выборе товаров. Поэтому для устранения отмеченного затруднения все три атрибута разносят по разным отношениям (таблицам). После выделения трех новых отношений (Поставщик, Товар и Покупатель) необходимо помнить, что при извлечении информации (например, О покупателях и товарах) необходимо в запросе соединить все три отношения. Любая комбинация соединения двух отношений изгрех неминуемо приведет к извлечению неверной (некорректной) информации. Некоторые СУБД снабжены специальными механизмами, устраняющими извлечение недостоверной информации. Тем не менее следует придерживаться общей рекомендации: структуру базы данных строить таким образом, чтобы избежать применения 4НФ и 5НФ.  
  
Пятая нормальная форма ориентирована на работу с зависимыми соединениями. Указанные зависимые соединения между тремя атрибутами встречаются очень редко. Зависимые соединения между четырьмя, пятью и более атрибутами указать практически невозможно.

### Доменно-ключевая нормальная форма (DKNF)

Переменная отношения находится в ДКНФ тогда и только тогда, когда каждое наложенное на неё ограничение является логическим следствием ограничений доменов и ограничений ключей, наложенных на данную переменную отношения.  
Ограничение домена – ограничение, предписывающее использовать для определённого атрибута значения только из некоторого заданного домена. Ограничение по своей сути является заданием перечня (или логического эквивалента перечня) допустимых значений типа и объявлением о том, что указанный атрибут имеет данный тип.  
  
Ограничение ключа – ограничение, утверждающее, что некоторый атрибут или комбинация атрибутов является потенциальным ключом.  
  
Любая переменная отношения, находящаяся в ДКНФ, обязательно находится в 5НФ. Однако не любую переменную отношения можно привести к ДКНФ.

### Шестая нормальная форма (6NF)

Переменная отношения находится в шестой нормальной форме тогда и только тогда, когда она удовлетворяет всем нетривиальным зависимостям соединения. Из определения следует, что переменная находится в 6НФ тогда и только тогда, когда она неприводима, то есть не может быть подвергнута дальнейшей декомпозиции без потерь. Каждая переменная отношения, которая находится в 6НФ, также находится и в 5НФ.  
  
Идея «декомпозиции до конца» выдвигалась до начала исследований в области хронологических данных, но не нашла поддержки. Однако для хронологических баз данных максимально возможная декомпозиция позволяет бороться с избыточностью и упрощает поддержание целостности базы данных.  
  
Для хронологических баз данных определены U\_операторы, которые распаковывают отношения по указанным атрибутам, выполняют соответствующую операцию и упаковывают полученный результат. В данном примере соединение проекций отношения должно производится при помощи оператора U\_JOIN.  
  
**Работники**

|  |  |  |  |
| --- | --- | --- | --- |
| Таб.№ | Время | Должность | Домашний адрес |
| 6575 | 01-01-2000:10-02-2003 | слесарь | ул.Ленина,10 |
| 6575 | 11-02-2003:15-06-2006 | слесарь | ул.Советская,22 |
| 6575 | 16-06-2006:05-03-2009 | бригадир | ул.Советская,22 |

Переменная отношения «Работники» не находится в 6НФ и может быть подвергнута декомпозиции на переменные отношения «Должности работников» и «Домашние адреса работников».  
  
**Должности работников**

|  |  |  |
| --- | --- | --- |
| Таб.№ | Время | Должность |
| 6575 | 01-01-2000:10-02-2003 | слесарь |
| 6575 | 16-06-2006:05-03-2009 | бригадир |

**Домашние адреса работников**

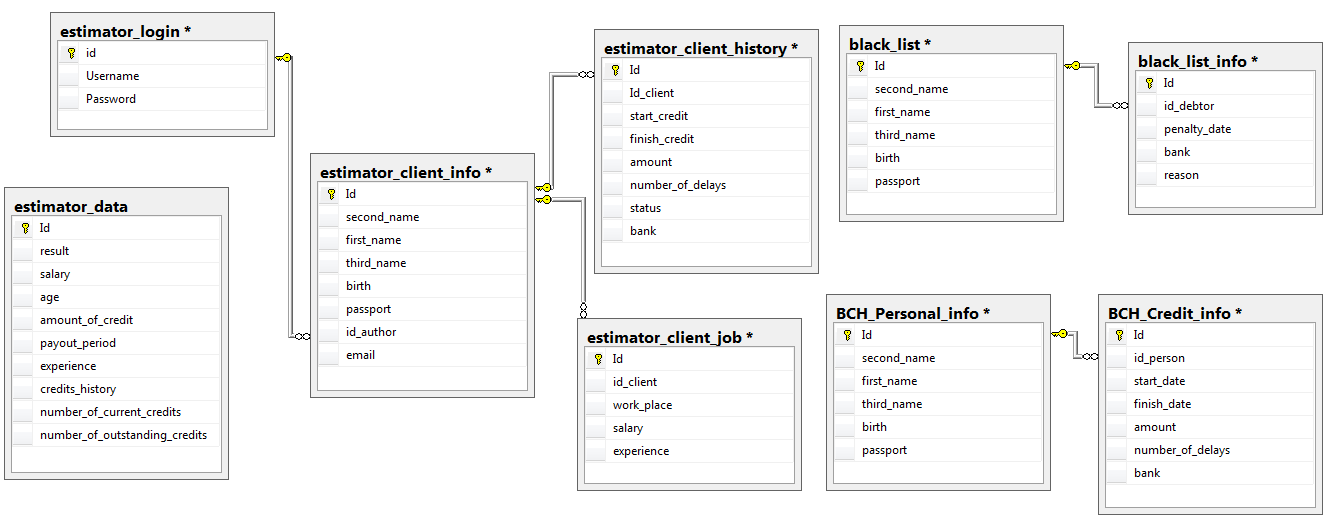
|  |  |  |
| --- | --- | --- |
| Таб.№ | Время | Домашний адрес |
| 6575 | 01-01-2000:10-02-2003 | ул.Ленина,10 |
| 6575 | 11-02-2003:15-06-2006 | ул.Советская,22 |

**Конструкторский раздел**

Для того что бы определить какие таблицы будут созданы в базе данных, в аналитическом разделе были разработаны сущности. Исходя из этого в базе данных «cosmetics» были созданы следующие таблицы:

* Estiator\_client\_info – соответствует информации о клиенте в сущности «Клиенты»
* Estiator\_client\_info – соответствует информации о кредитах в сущности «Клиенты»
* Estiator\_client\_job – соответствует информации о работе клиента в сущности «Клиенты»
* Estimator\_data – соответствует сущности «Данные для модели»
* Estimator\_login – соответствует сущности «Администраторы»
* BCH\_Personal\_info– соответствует информации о клиенте в сущности «Бюро кредитных историй»
* BCH\_Credit\_info – соответствует информации о кредитах в сущности «Бюро кредитных историй»
* Black\_list - соответствует информации о клиентах, находящихся в черном списке, в сущности «Черный список»
* Black\_list\_info - соответствует информации о причинах нахождения клиента в черном списке, в сущности «Черный список»

## 1. Определение связей между таблицами

* Таблица estimator\_data полностью состоит из элементов таблиц estimator\_clint\_info и estimator\_clint\_history,
* Так как для доступа информации из таблицы estimator\_clint\_history, нам надо получить информацию из таблицы estimator\_clint\_info, то эти таблицы связаны внешним ключом
* Так как для доступа информации из таблицы estimator\_clint\_job, нам надо получить информацию из таблицы estimator\_clint\_info, то эти таблицы связаны внешним ключом
* Так как для доступа информации из таблицы BCH\_Credit\_info, нам надо получить информацию из таблицы BCH\_Personal\_info, то эти таблицы связаны внешним ключом
* Так как для доступа информации из таблицы Black\_list\_info, нам надо получить информацию из таблицы Black\_list, то эти таблицы связаны внешним ключом

*Рисунок 1 (Структура базы данных).*

## 2. Определение ограничений на данные

В представленной выше базе данных каждая таблица содержит поле с названием **«id»**, в нем содержится идентификационный номер записи. Поле является первичным ключом таблицы («Primary Key»).

Таблица **«estimator\_client\_info»** - содержит персональную информацию о клиентах.

* **Id\_author** – список id из таблицы estimator\_LogIn, integer
* **first\_name** – имя, char (20 символов).
* **last\_name** – фамилия, char (20 символов).
* **third\_name** – отчество, char (20 символов).
* **birth** – дата рождения, date.
* **passport** – номер паспорта, integer.
* **email**(необязательное поле) – email, char (256 символов).

Таблица **«estimator\_client\_job»** - содержит информацию о работе клиентах.

* **Id\_client** – список id из таблицы estimator\_client\_info, integer
* **Work\_place** – место работы, char (20 символов).
* **salare** – зарплата, integer
* **experience** – опыт работы, integer.

Таблица **«estimator\_client\_history»** - содержит информацию о кредитах клиентов.

* **Id\_client** – список id из таблицы estimator\_client\_info, integer.
* **start\_credit** – дата выдачи кредита, date
* **finish\_date** – дата погашения кредита, date.
* **amount** – сумма кредита, integer.
* **number\_of\_delays**  – количество задержек, integer.
* **status** – статус кредита, char (20 символов).

Таблица **«estimator\_data»** содержит информацию о данных для анализа.

* **result** – результат, bool .
* **salary**– зарплата, integer.
* **age** (необязательное поле) – возраст, integer.
* **amount\_of\_credit** – сумма кредита, integer.
* **experience** – опыт работы, integer.
* **payout\_period –** сроквыплаты кредита, integer.
* **credits\_history –** количество до этого взятых кредитов, integer.
* **number\_of\_current\_credits –** количество текущих кредитов, integer.
* **number\_of\_outstanding\_credits -** количество задолжностей по

кредитам, integer

Таблица «**estimator\_login**» содержит информацию о администраторах.

* **username** – username, char(20)
* **password –** пароль, integer.

Таблица «**BCH\_Personal\_info**» содержит кредитной истории.

* **first\_name** – имя, char (20 символов).
* **last\_name** – фамилия, char (20 символов).
* **third\_name** – отчество, char (20 символов).
* **birth** – дата рождения, date.
* **passport** – номер паспорта, integer

Таблица «**BCH\_Credit\_info**» содержит кредитной истории.

* **Id\_person** – список id из таблицы BCH\_Personal\_info, integer.
* **start\_credit** – дата выдачи кредита, date
* **finish\_date** – дата погашения кредита, date.
* **amount** – сумма кредита, integer.
* **number\_of\_delays**  – количество задержек, integer.
* **bank** – название банка, integer

Таблица «**Black\_list**» содержит список клиентов, находящихся в черном списке.

* **first\_name** – имя, char (20 символов).
* **last\_name** – фамилия, char (20 символов).
* **third\_name** – отчество, char (20 символов).
* **birth** – дата рождения, date.
* **passport** – номер паспорта, integer

Таблица «**Black\_list\_info**» содержит кредитной истории.

* **Id\_deptor** – список id из таблицы Black\_list, integer.
* **Penalty\_date** – дата внесения клиента в черный список, date
* **bank** – название банка, integer
* **reason** – причина, char(50 символов)

## 3. Нормализация базы данных

Каждая нормальная форма налагает определенные ограничения на данные. Каждая нормальная форма более высокого уровня предполагает, что анализируемая таблица уже находится в нормальной форме на уровень ниже рассматриваемой. В ходе нормализации схема базы данных становится все более строгой, а ее таблицы все менее подвержены различного рода аномалиям.

Для реляционных баз данных необходимо, чтобы ее таблицы находились в 1НФ. Нормальные формы более высоких уровней могут использоваться разработчиками по своему усмотрению. Однако грамотный специалист стремится к тому, чтобы довести уровень нормализации базы данных хотя бы до 3НФ, тем самым исключив избыточность данных и аномалии обновления. Надо сказать, что НФБК, 4НФ и 5НФ используются крайне редко. Поэтому и мы рассмотрим только первые три.

**3.1 Первая нормальная форма**

Отношение находится в 1НФ, если все его атрибуты являются простыми, все используемые домены должны содержать только скалярные значения. Не должно быть повторений строк в таблице.

В данном случае все таблицы находятся в первой нормальной форме, все поля имеют атомарные значения.

**3.2. Вторая нормальная форма**

Таблица находится во второй нормальной форме, если она находится в первой нормальной форме, а каждое не ключевое поле функционально полно зависит от составного ключа.

Таблица estimator\_data не приведена ко второй нормальной форме, но так как данные из нее используются исключительно для обучения предсказательной модели, считаем это отступление приемлемым в рамках данного курсового проекта.

Все остальные таблицы также приведены ко второй нормальной форме, т.е. каждое поле каждой таблицы неприводимо зависит от Первичного Ключа

**3.3. Третья нормальная форма**

Отношение находится в третьей нормальной форме, если находится во второй нормальной форме и каждый не ключевой атрибут нетранзитивно зависит от первичного ключа.

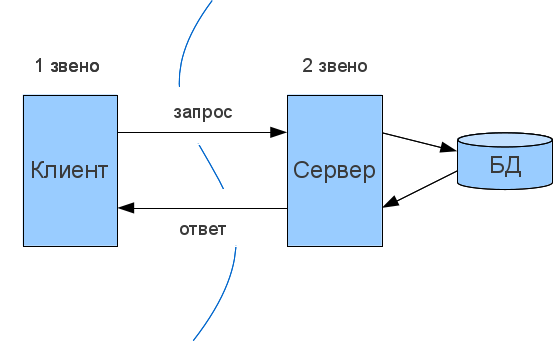
Таким образом, в этом случае все таблицы, за исключением estimator\_data находятся в третьей нормальной форме.

## Структура программы

Web-приложение состоит из трёх основных частей:

1. Клиентская часть – графический интерфейс приложения для удобного взаимодействия пользователя с данными и логикой приложения. Как правило, интерфейс приложения отображается в браузере.
2. Серверная часть – программа, которая описывает всю логику приложения, связывая интерфейс и базу данных в единое целое, и обрабатывает запросы от пользователя.
3. База данных и СУБД - программное обеспечение, которое управляет организацией и хранением данных в БД.

Схема взаимодействия частей приложения:



*Рисунок 2 (Схема работы Web приложения).*

**Технологический раздел**

## 1. Выбор языка программирования и среды разработки

В качестве языка программирования для реализации проекта был выбран язык программирования Python исходя из следующих критериев:

* Python - простой и популярный язык
* Код программы на Python читабелен
* Python имеет огромную базу библиотек
* Python является высокоуровневым языком программирования, что упрощает и ускоряет разработку приложения

Для разработки использовалась 2.7 версия Python, так как она включает в себя все необходимые инструменты разработки приложения.

В процессе разработки не использовалась никакая среда программирования.

## 2. Выбор web-фреймворка

На данный момент существует достаточно большое количество фреймворком для языка Python. Самыми яркими представителями являются: Django, Zope, Web2Py, Pylons, Flask. Для разработки приложения был выбран самый популярный из представленных - Django, так как он обладает следующими свойствами:

* Очень гибкая работа с url (при помощи регулярных выражений)
* Подробная и доступная документация
* Высокая скорость работы, может выдержать высокую нагрузку, также имеет встроенные возможности кэширования и распределения нагрузки.
* Расширяемая система шаблонов с тегами и наследованием

## 3. Выбор СУБД

В настоящее время Django может работать с такими СУБД: MySQL, SQLite, Microsoft SQL Server, DB2, Firebird, SQL Anywhere и Oracle.

Для разработки приложения была выбрана встроенная СУБД SQLite, так как она обладает следующими свойствами:

* Простота и удобство использования.
* Высокая производительность.
* Удобное тестирование функционала с помощью SQLite.
* Встроенная работа с SQLite в python.

## 4. Выбор технологий для стилизации клиентской части приложения

Для разработки внешнего вида приложения были использованы следующие средства:

* HTML
* CSS
* Bootstrap

HTML был использован для написания шаблонов страниц, как наиболее популярное и постое средство. В качестве описания внешнего вида шаблона был использован CSS 3.

## 5. Взаимодействие с базой данных

Для взаимодействия с базой данных был использован подход объектно-реляционного отображения с помощью Django ORM.

ORM (англ. object-relational mapping, рус. объектно-реляционное отображение) — технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных».

Использование ORM подхода имеет несколько преимуществ по сравнению с ручным написанием SQL:

* Безопасность. Параметры запросов экранируются, что делает атаки типа внедрение SQL-кода маловероятными.
* Производительность. Повышается вероятность повторного использования запроса к серверу базы данных, что может позволить ему в некоторых случаях применить повторно план выполнения запроса.
* Переносимость. Django ORM позволяет писать код на Python, совместимый с несколькими back-end СУБД. Несмотря на стандартизацию языка SQL, между базами данных имеются различия в его реализации, абстрагироваться от которых и помогает ORM.
* Нет необходимости создавать вспомогательные таблицы для связей типа многие ко многим. Django ORM создаст необходимые таблицы для обеспечения связей автоматически.

Для работы с базой данных были созданы следующие модели:

*class Estimate(models.Model):*

*id = models.AutoField(primary\_key=True)*

*Name = models.CharField(max\_length=20, null=True)*

*Second\_Name = models.CharField(max\_length=20, null=True)*

*Last\_Name = models.CharField(max\_length=20, null=True)*

*birth = models.DateField(blank=True, null=True)*

*passport = models.PositiveIntegerField(null=True)*

*Amount = models.PositiveIntegerField(null=True)*

*Salary = models.PositiveIntegerField(null=True)*

*Work\_Experience = models.PositiveIntegerField(null=True)*

*Repayment\_Period = models.PositiveIntegerField(null=True)*

*class LogIn(models.Model):*

*Username = models.CharField(max\_length=20)*

*Password = models.CharField(max\_length=20)*

*class client\_info(models.Model):*

*Id = models.AutoField(primary\_key=True)*

*id\_author = models.IntegerField(null=True)*

*second\_name = models.CharField(max\_length=20, null=True)*

*first\_name = models.CharField(max\_length=20, null=True)*

*third\_name = models.CharField(max\_length=20, null=True)*

*birth = models.DateField(default=timezone.now)*

*passport = models.IntegerField(null=True)*

*email = models.EmailField(blank=True, null=True)*

*class client\_history(models.Model):*

*Id = models.AutoField(primary\_key=True)*

*Id\_client = models.IntegerField(blank=True)*

*start\_credit = models.DateField(blank=True, null=True)*

*finish\_credit = models.DateField(blank=True, null=True)*

*amount = models.IntegerField(blank=True)*

*number\_of\_delays = models.IntegerField(blank=True)*

*status = models.CharField(max\_length=20, null=True)*

*class client\_job(models.Model):*

*Id = models.AutoField(primary\_key=True)*

*id\_client = models.IntegerField(blank=True)*

*work\_place = models.CharField(max\_length=20, null=True)*

*salary = models.IntegerField(null=True)*

*experience = models.IntegerField(null=True)*

## 

## 6. Выбор предсказательной модели

На данный момент существует множество алгоритмов для построения предсказательной модели, такие как дерево поиска, случайный лес, градиентный бустинг и другие.

Был выбран алгоритм – случайный лес (Random forest), так как он обладает следующими свойствами:

* высокой скорости обучения на данных, что при работе в web-приложении в режиме online является крайне важным
* малое количество настраиваемых параметров
* высокое качество предсказаний,
* реализованный алгоритм на языке python

Сам алгоритм является композицией алгоритмов дерево поиска, которое строится следующим образом:

1. Выбрать очередной признак P, поместить его в текущую вершину дерева

2. Для каждого значения выбранного признака v:

I Из тестовых данных оставить только те, у которых P = v

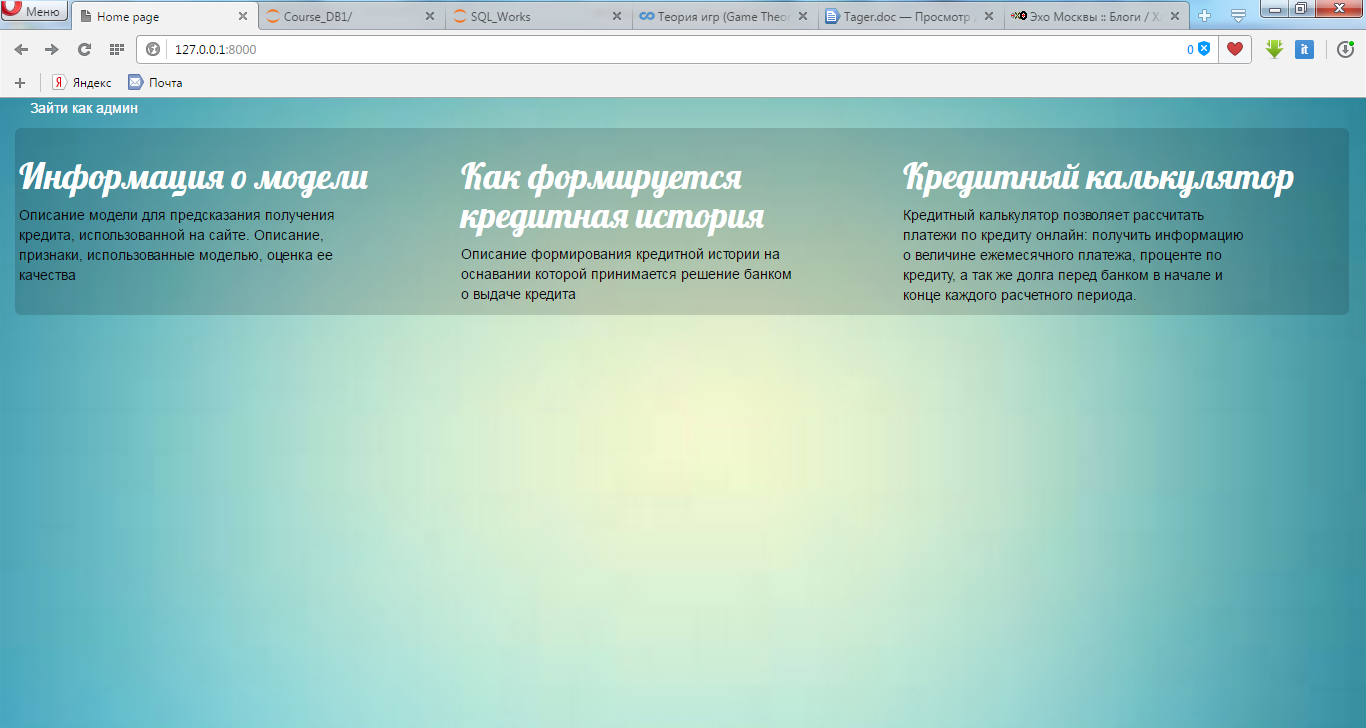
II Рекурсивно построить дерево на выбранных данных, рассматривая оставшихся признаки

3. Остановиться, если все тестовые объекты принадлежат одному классу, если закончились признаки, или по другим критериям

Данные разделяются на подвыборке, по каждой из которых строится дерево поиска. Результатом анализа будет усредненный ответ каждого дерева. В случае классификации формула выглядит следующим образом:

a(x) = sign∑ bi(x), где bi - это ответ i-го дерева.

## 7. Интерфейс программы

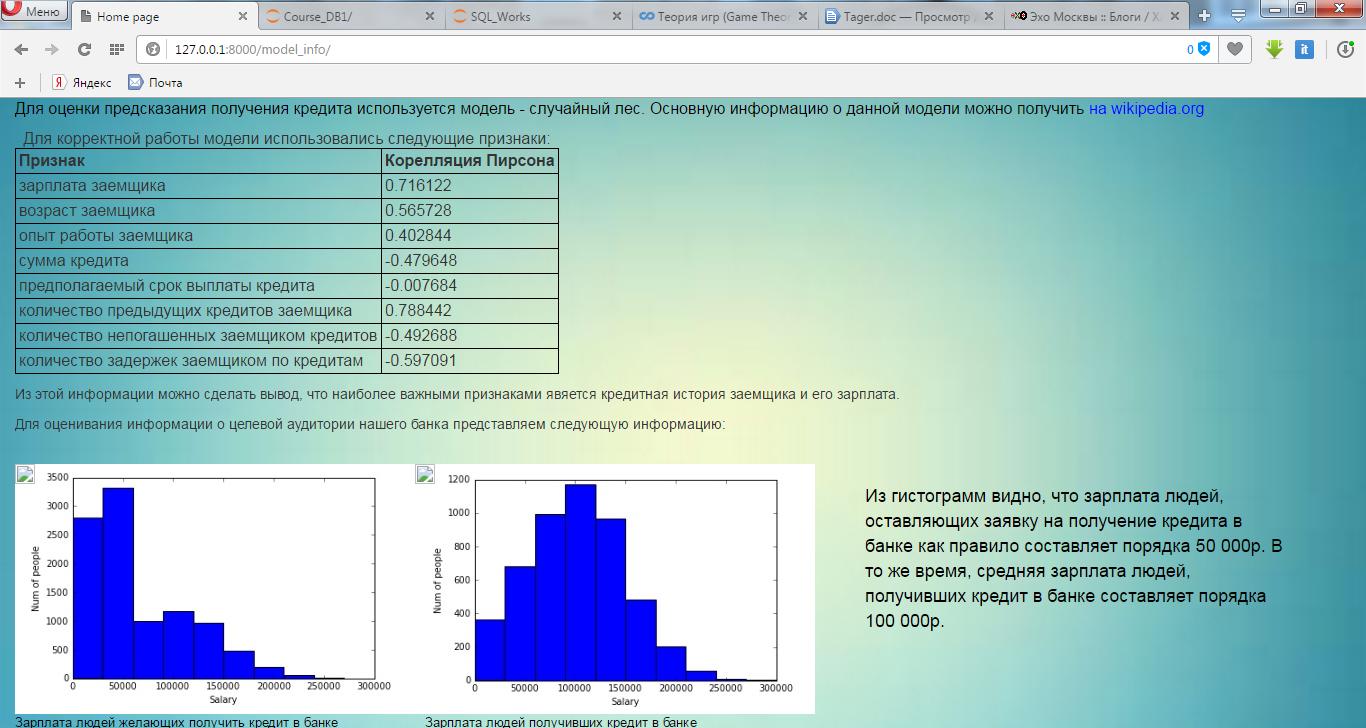


*Рисунок 3 (Главная страница сайта).*

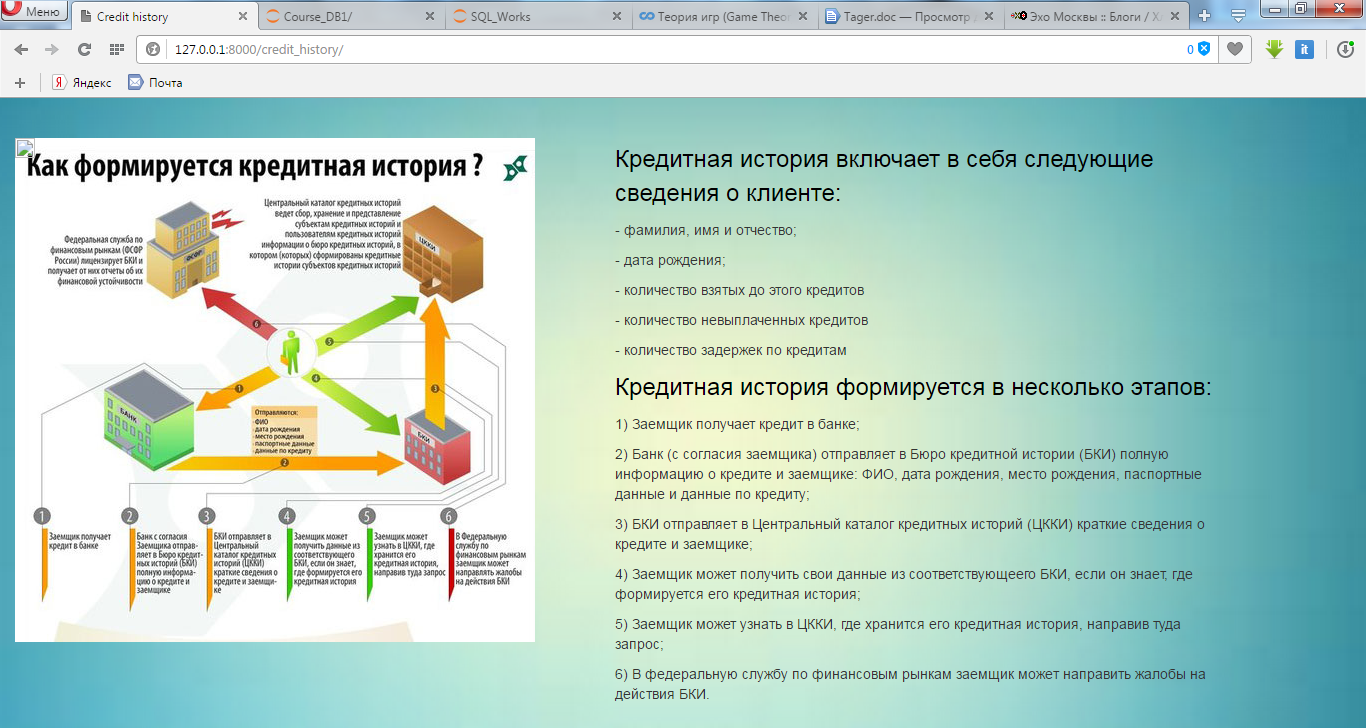
На рисунке выше представлена главная страница сайта, на ней располагается ссылки на такие страницы как:

* Информация о модели
* Информация о формировании кредитной истории
* Кредитный калькулятор
* Зайти как администратор

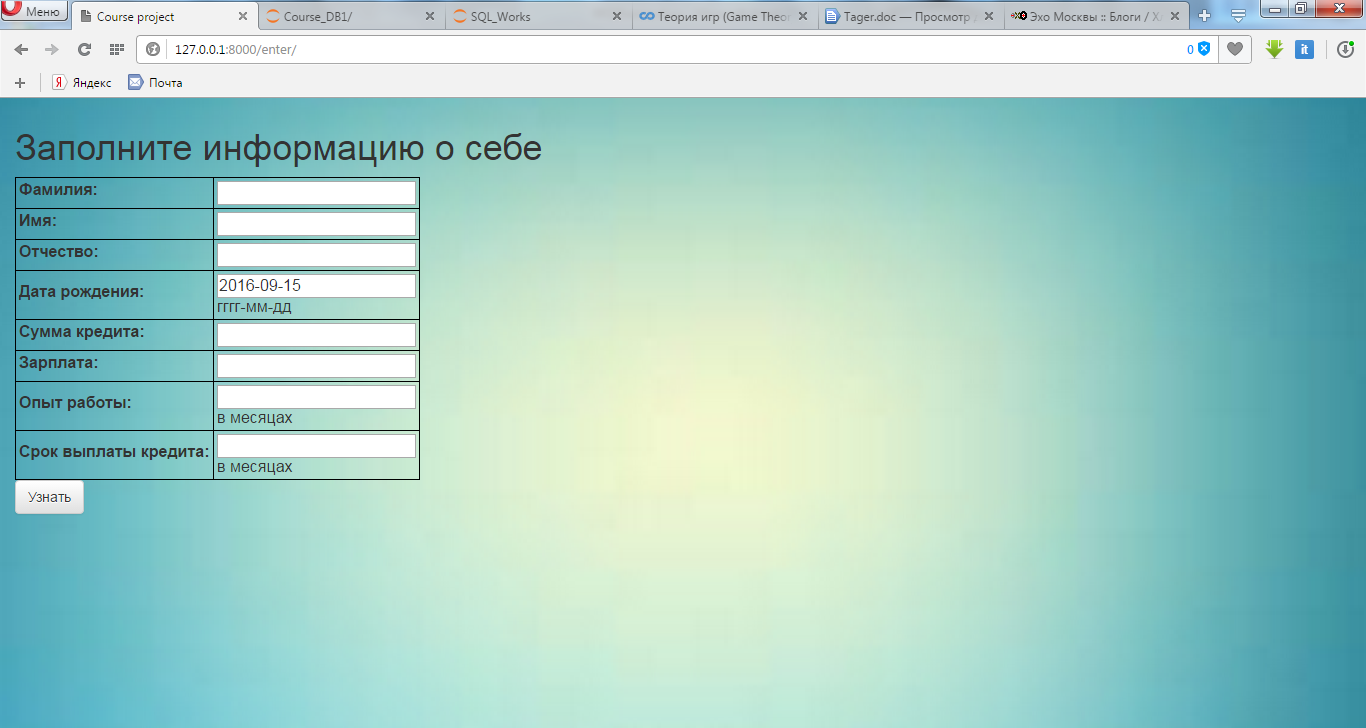
На рисунках 4, 5, 6 представлены результаты перехода по первым трем из этих ссылок:



*Рисунок 4 (Результат перехода по ссылке “Информация о модели”).*



*Рисунок 5 (Результат перехода по ссылке “Как формируется кредитная история”).*

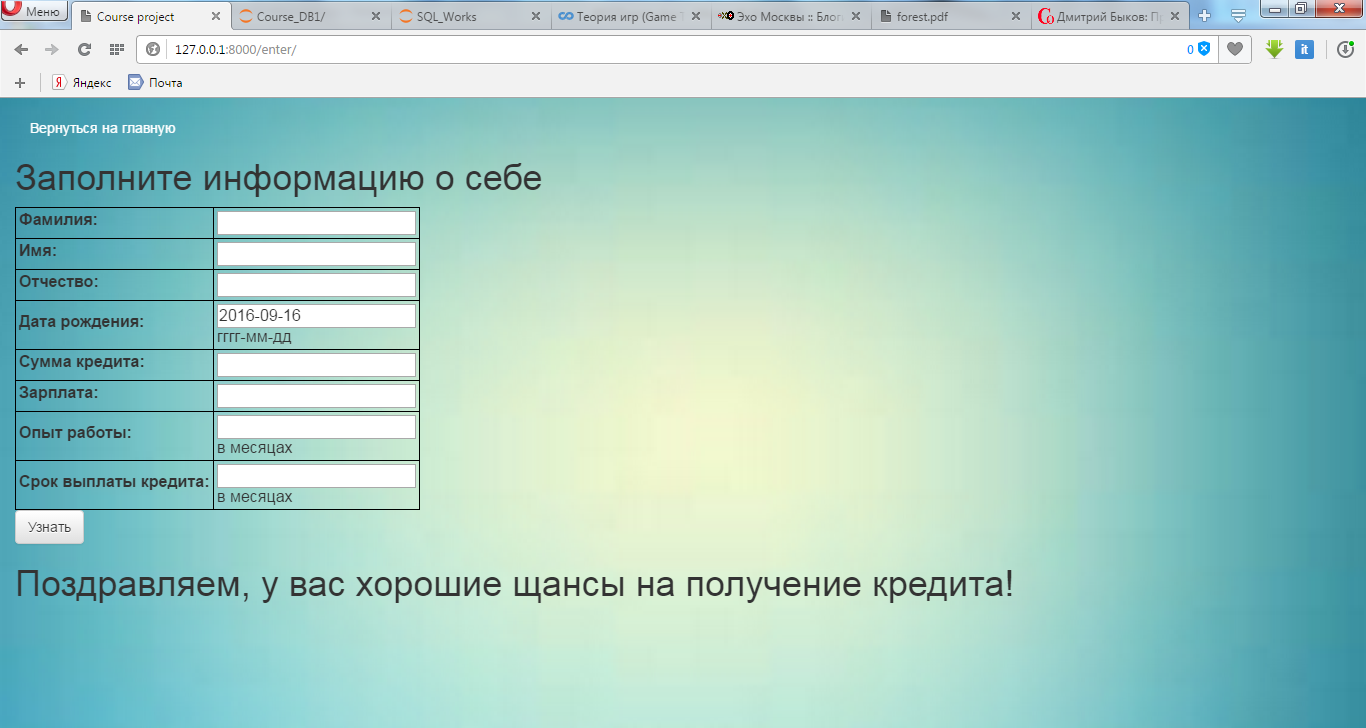


*Рисунок 6 (Результат перехода по ссылке “Кредитный калькулятор”).*

При заполнении необходимых форм и нажатии на кнопку на странице “кредитного калькулятора” будет выдан один из трех ответов на запрос:

* Согласие на предоставление кредита
* Отказ на предоставление кредита
* Отсутствие клиента в таблице бюро кредитной истории

Все эти события представлены на рисунках 7, 8, 9:

**

*Рисунок 7 (Согласие на предоставление кредита).*

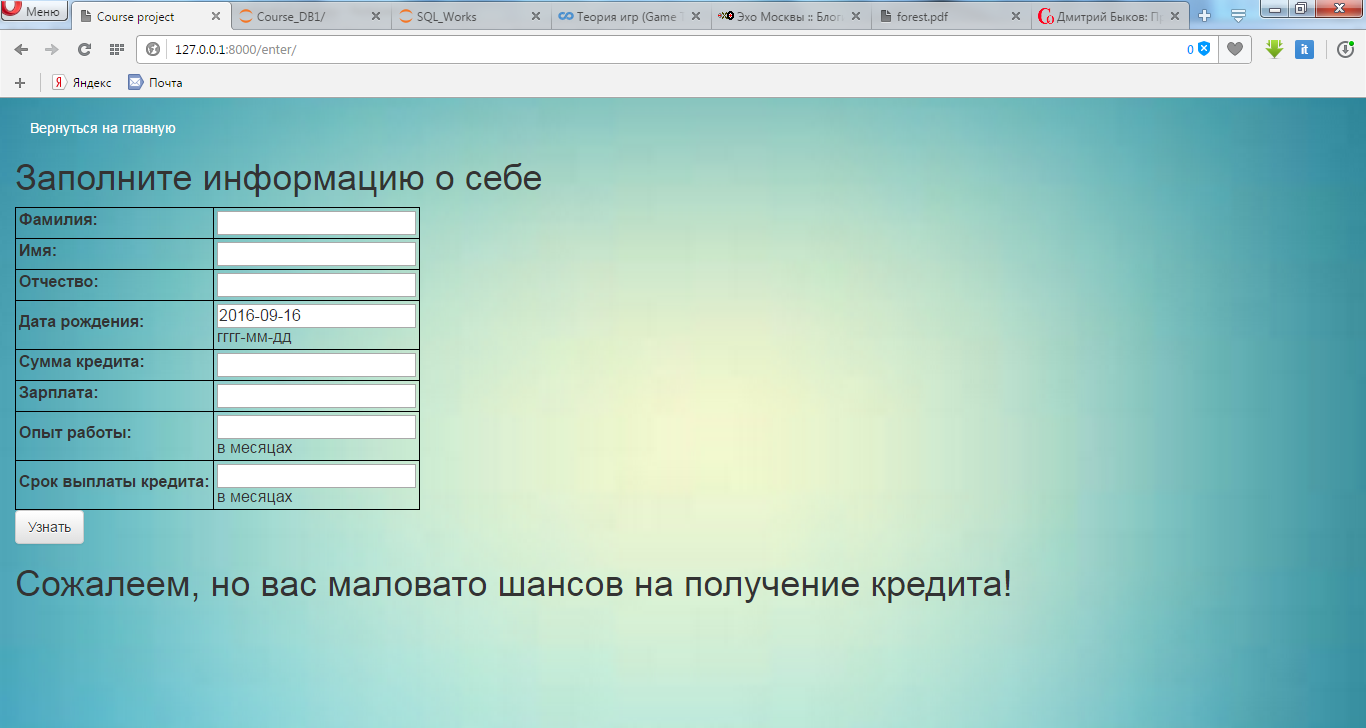
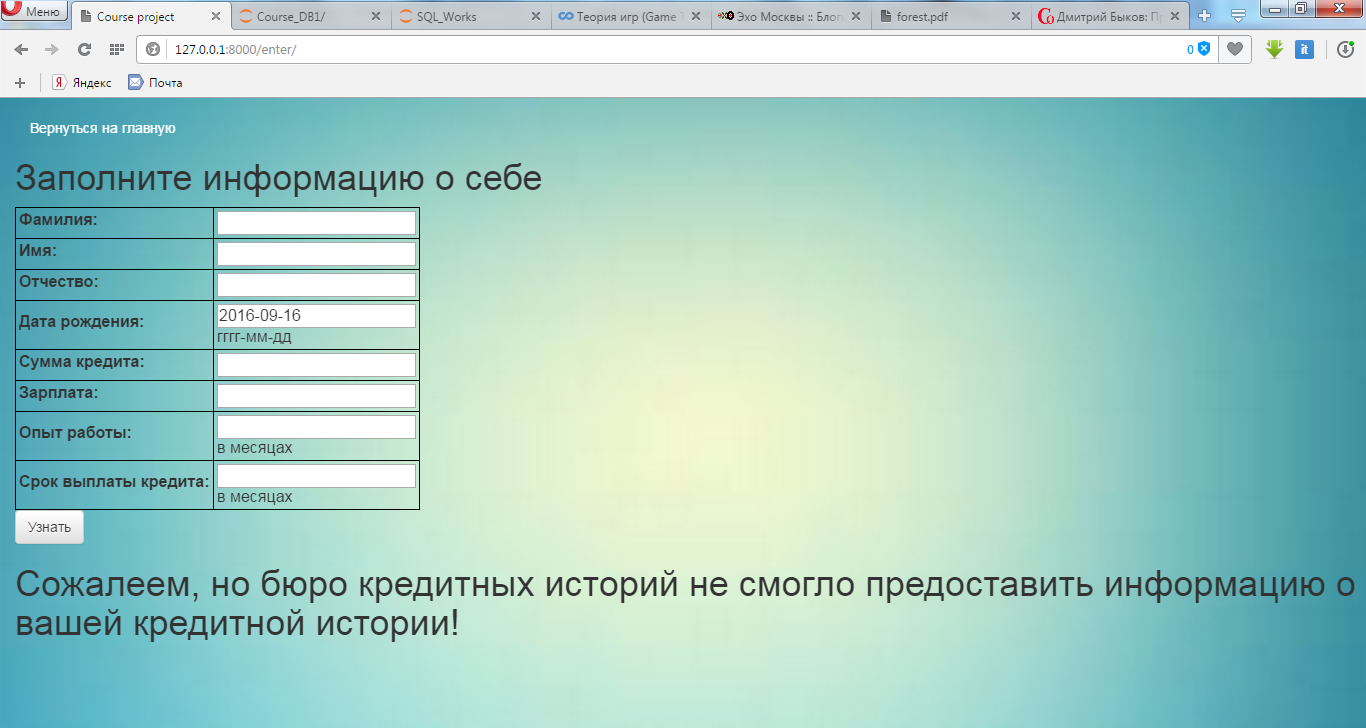
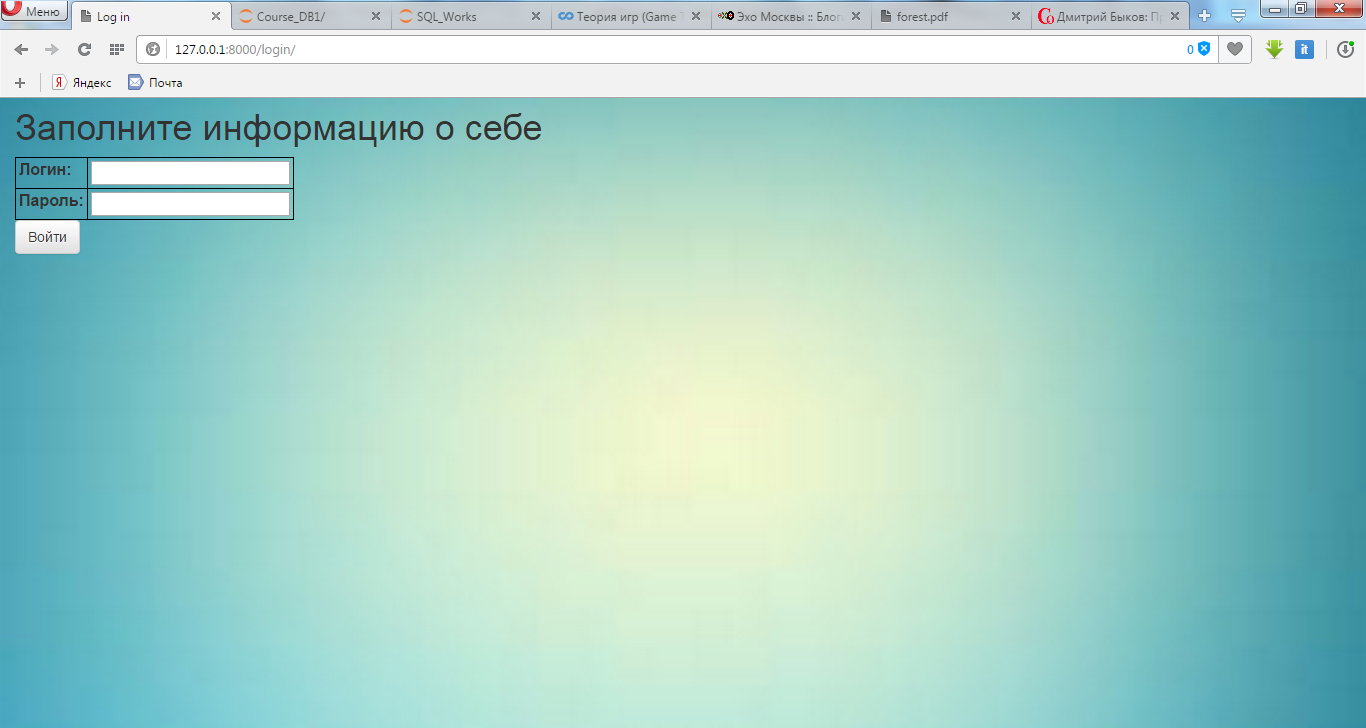


Рисунок 8 (Отказ на предоставление кредита).



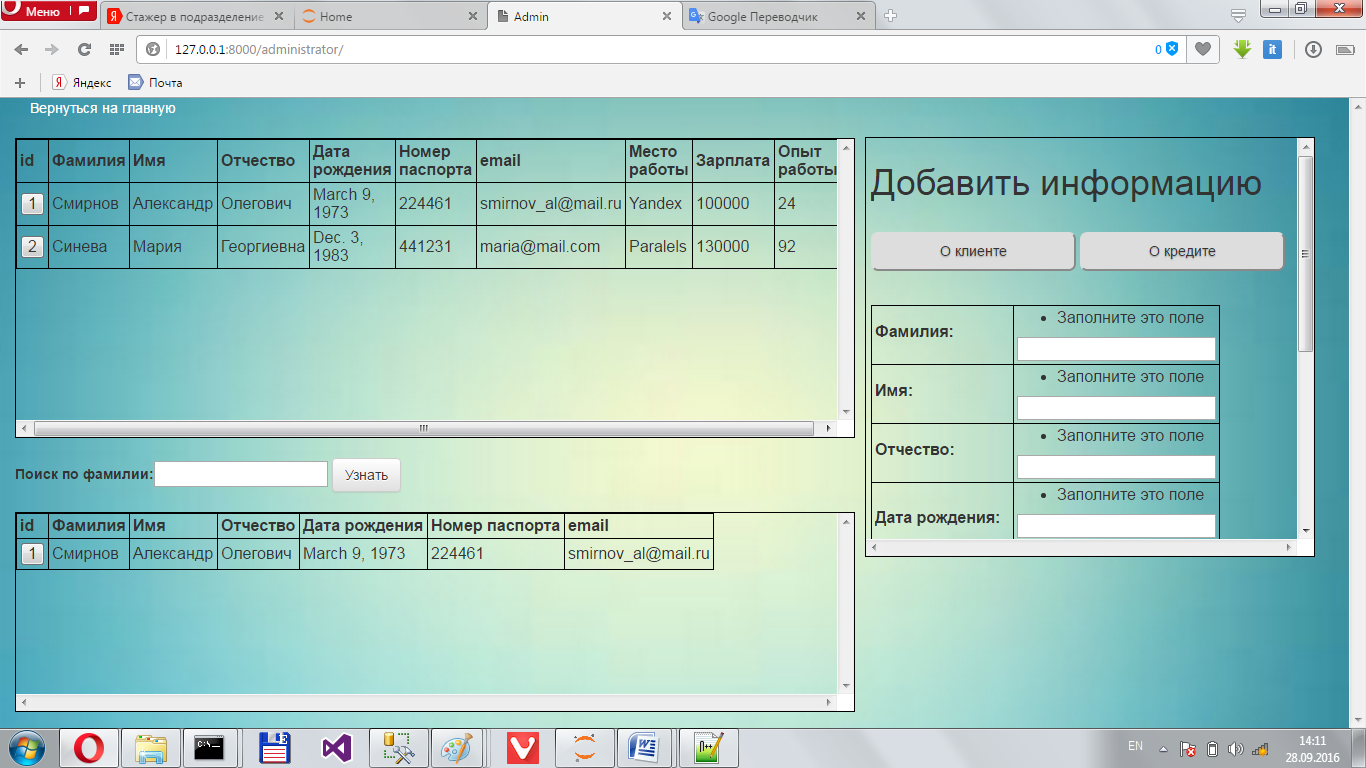
*Рисунок 9 (Отсутствие клиента в таблице бюро кредитной истории).*

На рисунке 10 представлен результат перехода по ссылке “зайти как амин”:



*Рис 10(переход по ссылке “зайти как админ”)*

Если пользователь не авторизовывался до этого, то ему придется ввести логин и пароль. При совпадении введенных значений, со значениями в базе данных, пользователь получит права администратора и возможность зайти на страницу администратора:



*Рис 11(страница администратора)*

На странице администратора можно записать в базу данных нового клиента, добавить информацию о кредите уже существующего клиента, или посмотреть список уже введенных клиентов. При верном заполнении администратором всех форм, новый клиент отобразится в таблице справа (рис 11)

Так же есть возможность узнать информацию о кредитной истории клиента из базы банка. Для этого достаточно нажать на id нужного клиента в таблице, и информация о его кредитной истории появится в виде таблицы (рис 12)



*Рис 12(информация о кредитах клиента)*

**8. Технические требования**

Для запуска приложения необходимо:

* Python 2.7
* Django 1.9.4

**Заключение**

В разработанном Web приложении реализована вся функциональность промо сайта, а именно: было реализовано средство оценивания рентабельности пользования услуг банка, подробная информацию об этом средстве, возможность внесения клиентов в базу, поиска клиетов и просмотра информации, об уже внесенных клиентах.

Также данное приложение имеет множество направлений для дальнейшего развития, например: сортировки клиентов в таблице для администратора, добавление информации о банке, комментарии пользователей и замена функционала предсказательной модели с бинарной классификации на оценивание вероятности получения кредита.

# **Список литературы**

* К. Дж. Дейт – Введение в системы баз данных
* Ицик Бен-Ган – Microsoft SQL Server 2012
* Марк Лутц – Программирования на pythonычвчвч
* http://djbook.ru/rel1.8/ - документация Django 1.8.
* <http://tutorial.djangogirls.org/ru/> - руководство Django Girls.
* http://python-3.ru/tag/sqlite%20python – использование SQLite в python