**version 3.05 (10 October 2019)**

Проектирование и разработка веб-приложения для клиентов предприятия "Сумские телекомсистемы"

Перший розділ містить огляд існуючих рішень, за результатами якого формулюється (у вигляді останнього підрозділу даного розділу) постановка задачі та основні завдання необхідні для її виконання.

Другий розділ містить інформацію про метод рішення проблеми. Як окремі підрозділи виділяються: основні положення, визначення та концепція методу; математичні моделі; критерії.

Третій розділ містить опис основних етапів проектування і реалізації інформаційного і програмного забезпечення, що було розроблено студентом для розв’язання поставленої задачі. Як окремі підрозділи виділяють: формування вхідних даних; алгоритми, що були розроблені; короткий опис програмної реалізації; аналіз результатів роботи програми.

Висновок містить стисле викладення теоретичних і практичних результатів, отриманих автором роботи особисто в ході дослідження, а також обґрунтування перспектив проведення подальших досліджень у даній галузі (Посилання на інших авторів, їх цитування, а також наведення загальновідомих істин не допускаються).

1. Анализ существующих решений

<https://my.volia.com/kiev/ru/faq>

<http://www.datainlife.ru/to_clients/instructions/152.htm>

**У Вас есть свой сайт**. Он является хорошим инструментом продаж. Но сайт может быть также хорошим инструментом сопровождения бизнеса. Одной из задач сайта является помощь в сервисе, который оказывает компания своим клиентам. Сайт может не только помочь в оказании услуг клиентам, но и перевести сервис на более высокий уровень, снять нагрузку с телефонов компании, позволить не держать большой штат клиентских менеджеров, т.к. клиент самостоятельно в личном кабинете решает основную массу задач. Для этого случая как раз и подходит так называемый **личный кабинет** клиента.

<https://habrahabr.ru/company/Centrobit/blog/142215/>

<https://netcat.ru/developers/docs/modules/module-profile/>

# СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 3](#_Toc21531492)

[ВВЕДЕНИЕ 5](#_Toc21531493)

[1 ОБЗОР ИЗВЕСТНЫХ РЕШЕНИЙ 6](#_Toc21531494)

[1.1 Текущее решение предприятия 6](#_Toc21531495)

[1.2 Примеры приложений «Личный кабинет» 11](#_Toc21531496)

[1.3 Обзор СУБД 12](#_Toc21531497)

[1.3.1 Классификации СУБД¶ 13](#_Toc21531498)

[1.3.2 Примеры СУБД 16](#_Toc21531499)

[1.4 Обзор языков программирования 18](#_Toc21531500)

[1.4.1 Клиентская часть (Front-end) 18](#_Toc21531501)

[1.4.2 Серверная часть (Back-end) 18](#_Toc21531502)

[2 ВЫБОР МЕТОДА РЕШЕНИЯ 19](#_Toc21531503)

[2.1 Проектирование информационной системы 19](#_Toc21531504)

[2.1.1 Построение диаграмм потоков данных 28](#_Toc21531505)

[2.1.2 Логическая структура базы данных 34](#_Toc21531506)

[2.2 Физическая реализация базы данных 46](#_Toc21531507)

[2.2.1 Создание ключевых сущностей базы данных 46](#_Toc21531508)

[2.2.2 Реализация типичных запросов 55](#_Toc21531509)

[2.3 Проектирование приложения 55](#_Toc21531510)

[2.3.1 Описание архитектуры приложения 55](#_Toc21531511)

[2.3.2 Создание UML диаграмм 59](#_Toc21531512)

[3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ 65](#_Toc21531513)

[3.1 Создание и настройка приложения 65](#_Toc21531514)

[3.1.1 Выбор среды разработки для разработки приложения 65](#_Toc21531515)

[3.1.2 Настройка системы сборки Maven 68](#_Toc21531516)

[3.1.3 Использования системы контроля версий 71](#_Toc21531517)

[3.1.4 Настройка веб-сервера и развертывание приложения 78](#_Toc21531518)

[3.2 Тестирование приложения 81](#_Toc21531519)

[3.2.1 Описание графического интерфейса 81](#_Toc21531520)

[3.2.2 Тестирование приложения 87](#_Toc21531521)

[ВЫВОДЫ 88](#_Toc21531522)

[СПИСОК ЛИТЕРАТУРЫ 89](#_Toc21531523)

# ВВЕДЕНИЕ

Целью производственной практики было освоение под непосредственным руководством преподавателя практическими приемами и навыками проектирования информационной системы для предприятия, применения полученных в процессе обучения теоретических знаний в области «Информатика».

Прохождение практики осуществлялось в ООО «Сумские телекомсистемы», где приоритетным направлением развития в сфере предоставления услуг доступа к сети Интернет является построение высокоскоростных каналов связи. Нашей целью было усовершенствовать информационную систему предприятия, создав веб приложение для клиентов, с помощью которого они смогут осуществлять взаимодействие с системой предприятия: вносить изменения в предоставляемые телекоммуникационные услуги, просматривать историю платежей, осуществлять оплату счетов, получать информацию о компании и о предоставляемых ею услугах.

В результате производственной практики будет спроектирована основная часть информационной системы предприятия, необходимая для обеспечения работы веб приложения.

# 1 ОБЗОР ИЗВЕСТНЫХ РЕШЕНИЙ

## 1.1 Текущее решение предприятия

На данный момент на предприятии ООО «Сумские телекомсистемы» существует несколько видов взаимодействия клиентов с предприятием:

* веб сайт <http://tks.sumy.ua>;
* телефоны технической поддержки;
* электронная почта;
* сервисный центр.

Веб сайт <http://tks.sumy.ua> является сайтом-визиткой предприятия.



Рисунок 1.1 Сайт предприятия

Основные задачи сайта:

* привлечение потенциальных клиентов из сети Интернет;
* предоставление информации об услугах, тарифах и акциях компании;
* предоставление контактной информации для непосредственной связи с компанией.

На сайте также находятся ссылки на нормативные документы предприятия и необходимая информация для частных клиентов, для бизнес клиентов и для строительных организаций.

Основные преимущества сайта визитки следующие:

* относительно низкая стоимость создания и разработки сайта;
* сайт не требует особых затрат на обслуживание и постоянное обновления контента;
* достаточно простой сайт, предоставляет только самую важную информацию;
* возможно модернизировать сайт в будущем.

Основные недостатки сайта визитки:

* невозможно автоматизировать процессы, поэтому требуется дополнительный штат сотрудников для работы с клиентами.

Таким образом сайт визитка является простым и удобным решением для представления компании в сети Интернет, но его функциональность довольно сильно ограничена.

Телефоны технической поддержки существуют для предоставления разнообразной информациии о предприятии и его услугах. Во-первых пользователи могут подключать, модифицировать или отключать услуги. Во-вторых с помощью телефонной связи можно решить технические проблемы или проблемы с системой оплаты счетов.

Один из телефонных номеров, указанных на сайте, является многоканальным. Многоканальный номер — это номер, позволяющий принимать несколько звонков одновременно. Звонки, поступающие на этот номер, переадресовываются на различные телефоны. Как правило, создается список номеров для переадресации, а также приоритетность соединения с тем или иным номером. В любом случае, на поступивший звонок отвечает один из свободных на данный момент операторов.

Компания опубликовала данный телефонный номер в открытом доступе, а на звонки отвечают специально обученные операторы call-центров. Благодаря многоканальному номеру клиентам компании не приходится подолгу ожидать соединения или слышать сигнал «занято». Количество одновременных соединений может быть устанавлено в зависимости от потребностей фирмы и текущей нагрузки.

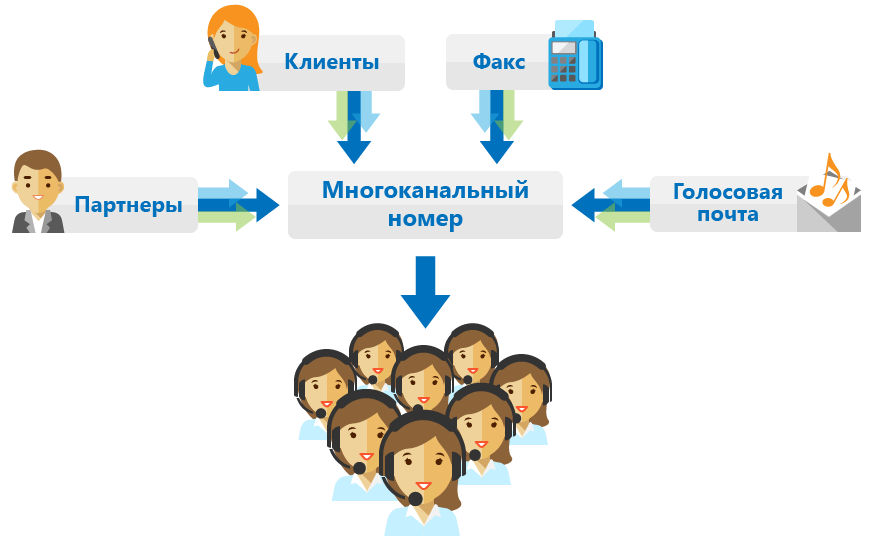


Рисунок 1.2 Схема телефонной связи при помощи многоканального номера

Одним из альтернативных путей коммуникации пользователей с предприятием является электронная почта. На сегодняшний день корпоративная электронная почта — неотъемлемая часть бизнес-культуры и бизнес-коммуникаций в любой компании. Невозможно даже представить себе современную жизнь и бизнес без электронной почты. Электронная почта служит средством связи, рекламы, продвижения товара или услуг и средством внутрикорпоративного взаимодействия. Однако, создание и поддержка сервера корпоративной электронной почты довольно сложный и трудоемкий процесс для системного администратора компании. Ведь необходимо детально продумать и реализовать аппаратное и программное обеспечение, обеспечить антивирусную и антиспам-защиту будущего почтового сервера, а также не забыть об удобстве пользователей электронной почты.

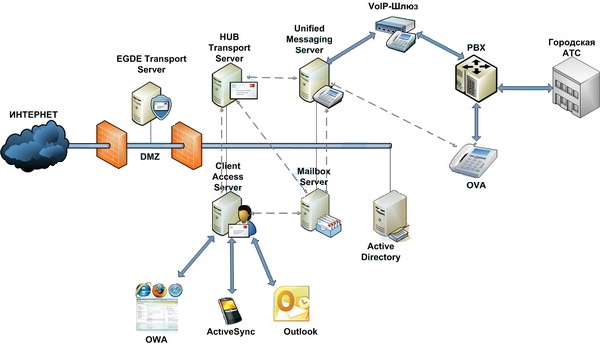


Рисунок 1.3 Схема организации корпоративной почты

По электронной почте пользователи могут присылать свои вопросы или запросы на изменение персональных данных или услуг. На данный момент существует 2 электронные адреса предприятия по которым можно обратиться:

* общие вопросы: info@tks.sumy.ua;
* служба поддержки: help@tks.sumy.ua.

Также у клиентов есть возможность обратиться напрямую в сервисный центр чтобы решить все необходимые вопросы.

В данном случае сервисный центр – это организация, занимающаяся оказанием услуг по сервисной поддержке, обслуживанию телекоммуникационного оборудования и предоставлению консультаций по предоставляемым продуктам.

Адресс и график работы сервисного центра можно найти на сайте компании или же уточнить по телефону.

Таким образом, чтобы удовлетворить потребности клиентов, предприятие должно содержать упомянутые выше средства коммуникации. В результате этого увеличивается количество сотрудников, необходимых для выполнения этих задач. Следовательно растут и расходы предприятия.

Идея данной работы в том, чтобы автоматизировать часть этих функций создав веб приложение для клиентов, с помощью которого они смогут контролировать состояние своего счета, осуществлять оплату, изменять личные данные, а также получать необходимую информацию о предоставляемых услугах и управлять ними.

Веб приложение называется «Личный кабинет». Личный кабинет может не только помочь в предоставлении услуг клиентам, но и перевести сервис на более высокий уровень.

Таким образом данное решение позволит снять нагрузку с телефонных линий компании и сделать коммуникацию между пользователями и предприятием более эффективной и удобной. В результате успешного внеднения данного приложения компания может сократить расходы на содержание большого штата обслуживающего персонала, так как клиент самостоятельно в личном кабинете решает основную массу задач.

## 1.2 Примеры приложений «Личный кабинет»

TDB

***Vodafone***

***Kyivstar***

***Volya***

***Koodo Mobile***

***Shaw Communications***

## 1.3 Обзор СУБД

Система управления базами данных (СУБД) — совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных.



Рисунок № Схема работы СУБД

Основные функции СУБД

* управление данными во внешней памяти (на дисках);
* управление данными в оперативной памяти с использованием дискового кэша;
* журнализация изменений, резервное копирование и восстановление базы данных после сбоев;
* поддержка языков БД (язык определения данных, язык манипулирования данными).

Обычно современная СУБД содержит следующие компоненты:

* ядро, которое отвечает за управление данными во внешней и оперативной памяти и журнализацию,
* процессор языка базы данных, обеспечивающий оптимизацию запросов на извлечение и изменение данных и создание, как правило, машинно-независимого исполняемого внутреннего кода,
* подсистему поддержки времени исполнения, которая интерпретирует программы манипуляции данными, создающие пользовательский интерфейс с СУБД
* сервисные программы (внешние утилиты), обеспечивающие ряд дополнительных возможностей по обслуживанию информационной системы.

### 1.3.1 Классификации СУБД¶

**По модели данных**

***Иерархические***

Используется представление базы данных в виде древовидной (иерархической) структуры, состоящей из объектов (данных) различных уровней.

Между объектами существуют связи, каждый объект может включать в себя несколько объектов более низкого уровня. Такие объекты находятся в отношении предка (объект более близкий к корню) к потомку (объект более низкого уровня), при этом возможна ситуация, когда объект-предок не имеет потомков или имеет их несколько, тогда как у объекта-потомка обязательно только один предок. Объекты, имеющие общего предка, называются близнецами (в программировании применительно к структуре данных дерево устоялось название братья).

Иерархической базой данных является файловая система, состоящая из корневого каталога, в котором имеется иерархия подкаталогов и файлов.

***Сетевые***

Сетевые базы данных подобны иерархическим, за исключением того, что в них имеются указатели в обоих направлениях, которые соединяют родственную информацию.

***Реляционные***

Практически все разработчики современных приложений, предусматривающих связь с системами баз данных, ориентируются на реляционные СУБД. По оценке Gartner в 2013 году рынок реляционных СУБД составлял 26 млрд долларов с годовым приростом около 9%, а к 2018 году рынок реляционных СУБД достигнет 40 млрд долларов. В настоящее время абсолютными лидерами рынка СУБД являются компании Oracle, IBM и Microsoft, с общей совокупной долей рынка около 90%, поставляя такие системы как Oracle Database, IBM DB2 и Microsoft SQL Server.

***Объектно-ориентированные***

Управляют базами данных, в которых данные моделируются в виде объектов, их атрибутов, методов и классов.

Этот вид СУБД позволяет работать с объектами баз данных так же, как с объектами в программировании в объектно-ориентированных языках программирования. ООСУБД расширяет языки программирования, прозрачно вводя долговременные данные, управление параллелизмом, восстановление данных, ассоциированные запросы и другие возможности.

***Объектно-реляционные***

Этот тип СУБД позволяет через расширенные структуры баз данных и язык запросов использовать возможности объектно-ориентированного подхода: бъекты, классы и наследование.

Зачастую все те СУБД, которые называются реляционными, являются, по факту, объектно-реляционными.

В данном курсе мы будем, в первую очередь, гооврить об этом виде СУБД.

**По степени распределённости**

* Локальные СУБД (все части локальной СУБД размещаются на одном компьютере)
* Распределённые СУБД (части СУБД могут размещаться на двух и более компьютерах).

**По способу доступа к БД**

***Файл-серверные***

В файл-серверных СУБД файлы данных располагаются централизованно на файл-сервере. СУБД располагается на каждом клиентском компьютере (рабочей станции). Доступ СУБД к данным осуществляется через локальную сеть. Синхронизация чтений и обновлений осуществляется посредством файловых блокировок. Преимуществом этой архитектуры является низкая нагрузка на процессор файлового сервера. Недостатки: потенциально высокая загрузка локальной сети; затруднённость или невозможность централизованного управления; затруднённость или невозможность обеспечения таких важных характеристик как высокая надёжность, высокая доступность и высокая безопасность. Применяются чаще всего в локальных приложениях, которые используют функции управления БД; в системах с низкой интенсивностью обработки данных и низкими пиковыми нагрузками на БД.

На данный момент файл-серверная технология считается устаревшей, а её использование в крупных информационных системах — недостатком.

***Клиент-серверные***

Клиент-серверная СУБД располагается на сервере вместе с БД и осуществляет доступ к БД непосредственно, в монопольном режиме. Все клиентские запросы на обработку данных обрабатываются клиент-серверной СУБД централизованно. Недостаток клиент-серверных СУБД состоит в повышенных требованиях к серверу. Достоинства: потенциально более низкая загрузка локальной сети; удобство централизованного управления; удобство обеспечения таких важных характеристик как высокая надёжность, высокая доступность и высокая безопасность.

***Встраиваемые***

Встраиваемая СУБД — СУБД, которая может поставляться как составная часть некоторого программного продукта, не требуя процедуры самостоятельной установки. Встраиваемая СУБД предназначена для локального хранения данных своего приложения и не рассчитана на коллективное использование в сети. Физически встраиваемая СУБД чаще всего реализована в виде подключаемой библиотеки. Доступ к данным со стороны приложения может происходить через SQL либо через специальные программные интерфейсы (API).

### 1.3.2 Примеры СУБД

Теория что такое СУБД + красивые диаграммки с сайта:

<https://www.mindmeister.com/ru/1153476789/_>

***MySQL***

***Oracle***

***PostgreSQL***

***SQLite***

***NoSQL решения***

<https://en.wikipedia.org/wiki/NoSQL>

<https://bigdata-madesimple.com/18-free-and-widely-used-open-source-nosql-databases/>

* MongoDB
* Redis
* Couch DB
* REVENDB
* MemcacheDB

## 1.4 Обзор языков программирования

<https://www.cleverism.com/programming-languages-web-development/>

<https://www.upwork.com/blog/2017/11/top-web-development-languages-2018/>

<https://medium.com/swlh/best-10-programming-languages-to-learn-in-2018-2d6cbc5ffc2a>

<https://blog.programminghub.io/blog/2017/10/04/top-programming-languages-used-web-development/>

TDB

### 1.4.1 Клиентская часть (Front-end)

TDB

### 1.4.2 Серверная часть (Back-end)

TDB

# 2 ВЫБОР МЕТОДА РЕШЕНИЯ

## 2.1 Проектирование информационной системы

Для обеспечения функционирования веб приложения «Личный кабинет» нам необходимо разработать информационную систему.

Информационная система (ИС) представляет собой система сбора, хранения, обработки, преобразования, передачи и обновления информации с использованием компьютерной и другой техники. Элементами этой системы являются не материальные объекты, а те или иные виды данных (информации), которые взаимодействуют и превращаются в процессе ее функционирования.

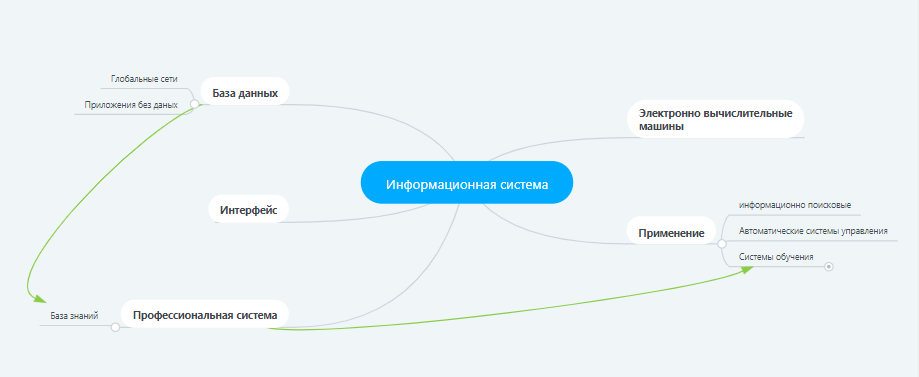


Рисунок 2.1 Компоненты информационной системы

Сайт с диаграммами: <https://www.mindmeister.com/ru/1153476789/_>

***Основные задачи информационных систем***

Современные информационные системы решают следующие основные задачи.

1. Осуществление поиска, обработки и хранения информации, которая накапливается в течение большого периода времени, имеет большую ценность.

ИС предназначены для более быстрой и надёжной обработки информации, чтобы люди не тратили время, чтобы избежать свойственных человеку случайных ошибок, чтобы сэкономить расходы, чтобы сделать жизнь людей более комфортной.

2. Хранение данных разной структуры. Не существует развитой ИС, работающей с одним однородным файлом данных. Более того, разумным требованием к информационной системе является то, чтобы она могла развиваться.

Могут появиться новые функции, для выполнения которых требуются дополнительные данные с новой структурой. При этом вся накопленная ранее информация должна остаться сохранной. Теоретически можно решить эту задачу путём использования нескольких файлов внешней памяти, каждый из которых хранит данные с фиксированной структурой. В зависимости от способа организации используемой системы управления файлами эта структура может быть структурой записи файла или поддерживаться отдельной библиотечной функцией, написанной специально для данной ИС. Известны примеры реально функционирующих ИС, в которых хранилище данных планировалось основывать на файлах. В результате развития большинства таких систем в них выделился отдельный компонент, который представляет собой разновидность системы управления базами данных (СУБД).

3. Анализ и прогнозирование потоков информации различных видов и типов, перемещающихся в обществе. Изучаются потоки с целью их минимизации, стандартизации и приспособления для эффективной обработки на вычислительных машинах, а также особенности потоков информации, протекающей через различные каналы распространения информации.

4. Исследование способов представления и хранения информации, создание специальных языков для формального описания информации различной природы, разработка специальных приёмов сжатия и кодирования информации, аннотирования объёмных документов и реферирования их. В рамках этого направления развиваются работы по созданию банков данных большого объёма, хранящих информацию из различных областей знаний в форме, доступной для вычислительных машин.

5. Построение процедур и технических средств для их реализации, с помощью которых можно автоматизировать процесс извлечения информации из документов, не предназначенных для вычислительных машин, а ориентированных на восприятие их человеком.

6. Создание информационно-поисковых систем, способных воспринимать запросы к информационным хранилищам, сформулированные на естественном языке, а также специальных языках запросов для систем такого типа.

7. Создание сетей хранения, обработки и передачи информации, в состав которых входят информационные банки данных, терминалы, обрабатывающие центры и средства связи.

Конкретные задачи, которые должны решаться информационной системой, зависят от той прикладной области, для которой предназначена система. Области применения информационных приложений разнообразны: банковское дело, управление производством, медицина, транспорт, образование, юриспруденция и т.д.

***Основные свойства и процессы в информационных системах***

Информационная система определяется следующими свойствами.

1. Структура ИС, её функциональное назначение должны соответствовать поставленным целям.

2. ИС предназначена для производства достоверной, надёжной, своевременной и систематизированной информации, основанной на использовании БД, экспертных систем и баз знаний. Так как любая ИС предназначена для сбора, хранения и обработки информации, то в основе любой ИС лежит среда хранения и доступа к данным. Среда должна обеспечивать уровень надёжности хранения и эффективность доступа, которые соответствуют области применения ИС.

3. ИС должна контролироваться людьми, ими пониматься и использоваться в соответствии с основными принципами, реализованными в виде стандарта организации на ИС. Интерфейс пользователя ИС должен быть легко понимаем на интуитивном уровне.

4. Любая информационная система может быть подвергнута анализу, построена и управляема на основе общих принципов построения систем.

5. Любая ИС является динамичной и развивающейся.

6. При построении ИС используются сети передачи данных. Процессы, обеспечивающие работу информационной системы любого назначения, условно можно представить в виде схемы (рис. 2.4.1), состоящей из блоков:

– ввод информации из внешних или внутренних источников;

– обработка входной информации и представление её в удобном виде;

– вывод информации для представления потребителям или передачи в другую систему;

– обратная связь – это информация, переработанная людьми данной организации для коррекции входной информации.

***Классификации информационных систем***

Информационные системы могут значительно различаться по типам объектов, характером и объемом решаемых задач и рядом других признаков.

Общепринятой классификации ИС до сих пор не существует, поэтому их можно классифицировать по разным признаками, что вызвало существование нескольких различных классификаций ИС.

***Классификация по архитектуре***

По степени распределённости отличают:

* настольные, или локальные ИС, в которых все компоненты (БД, СУБД, клиентские приложения) находятся на одном компьютере;
* распределённые ИС, в которых компоненты распределены по нескольким компьютерам.

Распределённые ИС, в свою очередь, разделяют на:

* файл-серверные ИС (ИС с архитектурой «файл-сервер»);
* клиент-серверные ИС (ИС с архитектурой «клиент-сервер»).

В файл-серверных ИС база данных находится на файловом сервере, а СУБД и клиентские приложения находятся на рабочих станциях.

В клиент-серверных ИС база данных и СУБД находятся на сервере, а на рабочих станциях находятся только клиентские приложения.

В свою очередь, клиент-серверные ИС разделяют на двухзвенные и многозвенные.

В двухзвенных ИС всего два типа «звеньев»: сервер базы данных, на котором находятся БД и СУБД, и рабочие станции, на которых находятся клиентские приложения. Клиентские приложения обращаются к СУБД напрямую.

В многозвенных ИС добавляются промежуточные «звенья»: серверы приложений. Пользовательские клиентские приложения не обращаются к СУБД напрямую, они взаимодействуют с промежуточными звеньями. Типичный пример применения трёхзвенной архитектуры — современные веб-приложения, использующие базы данных. В таких приложениях помимо звена СУБД и клиентского звена, выполняющегося в веб-браузере, имеется как минимум одно промежуточное звено — веб-сервер с соответствующим серверным программным обеспечением.

***Классификация по степени автоматизации***

По степени автоматизации ИС делятся на:

* автоматизированные: информационные системы, в которых автоматизация может быть неполной (то есть требуется постоянное вмешательство персонала);
* автоматические: информационные системы, в которых автоматизация является полной, то есть вмешательство персонала не требуется или требуется только эпизодически.

«Ручные ИС» («без компьютера») существовать не могут, поскольку существующие определения предписывают обязательное наличие в составе ИС аппаратно-программных средств. Вследствие этого понятия «автоматизированная информационная система», «компьютерная информационная система» и просто «информационная система» являются синонимами.

***Классификация по характеру обработки данных***

По характеру обработки данных ИС делятся на:

* информационно-справочные, или информационно-поисковые ИС, в которых нет сложных алгоритмов обработки данных, а целью системы является поиск и выдача информации в удобном виде;
* ИС обработки данных, или решающие ИС, в которых данные подвергаются обработке по сложным алгоритмам. К таким системам в первую очередь относят автоматизированные системы управления и системы поддержки принятия решений.

***Классификация по охвату задач (масштабности)***

По охвату задач (масштабности) ИС делятся на:

* персональная ИС предназначена для решения некоторого круга задач одного человека.
* групповая ИС ориентирована на коллективное использование информации членами рабочей группы или подразделения.
* корпоративная ИС автоматизирует все бизнес-процессы целого предприятия (организации) или их значительную часть, достигая их полной информационной согласованности, безызбыточности и прозрачности. Такие системы иногда называют информационными системами предприятия и системами комплексной автоматизации предприятия.

Разрабатываемая нами информационная система предназначена для обеспечения функций веб-приложения «Личный кабинет». Основные пользователи данной системы это клиенты предприятия ООО «Сумские телекомсистемы».

Основной перечень функций, доступных пользователям личного кабинета:

1. управление телекоммуникационными услугами;
2. управление платежами (просмотр истории платежей и оплата счетов);
3. поиск информации о предприятии и его услугах.

Рассмотрим каждую из этих функций более подробно.

Управление телекоммуникационными услугами представляет собой взаимодействие с такими сервисами как высокоскоростной интернет, телевидение и телефонная связь. Каждый сервис имеет свой набор тарифов, характеристик и параметров, которые могут быть изменены.

В личном кабинете пользователь может заказать новый сервис, отключить или внести определенные модификации в существующий.

В любой момент времени в веб-приложении отображается перечень установленных услуг и список доступных сервисов для подключения. При отключении существующей услуги пользователю нужно сообщить причину, по которой он отказывается от данной услуги. При заказе новой услуги он должен указать определенные параметры (скорость, оборудование, количество каналов, телефонный номер, способ оплаты и т.д.).

Также у пользователя есть возможность в зависимости от типа сервиса изменять некоторые параметры. Это могут быть как общие параметры (тарифный план), так и более специфичные для каждого сервиса (скорость и тип интернета, количество телевизионных каналов, телефонный номер и наличие голосовой почты).

Управление платежами включает в себя просмотр истории платежей, осуществление оплаты текущего счета и настройку автоматического платежа.

При помощи истории платежей пользователь может проследить динамику расходов на услуги в течении определенного периода времени, а также объем использованных данных (интернет трафик, количество телефонных минут и т.д.).

Кроме этого, доступная функция оплаты счета. Это может быть единоразовый платеж или же настройка автоматической оплаты в случае фиксированного тарифа. Для этого пользователь должен предоставить данные банковского счета или кредитной карты. Функция автоматической оплаты позволит не пропустить дату платежа и сделает ведение учета более простым и удобным для клиента.

Поиск информации о предприятии и его услугах состоит из нескольких составляющих.

Первым составляющим является получение контактных данных предприятия, графика работы сервисных центров. Контактные данные могут в себя включать ссылки на интернет ресурсы, номера телефонов, адреса сервисных центров. В перспективе возможна интеграция в разрабатываемое веб-приложение онлайн поддержки в виде чата.

Данная информация поможет пользователю наиболее удобным для него путем решить возникнувшие вопросы и технические проблемы.

Второй составляющей частью является доска обявлений. При помощи доски объявлений пользователь имеет возможность видеть у себя на странице полезную информацию о компании и ее услугах. Это могут быть объявления о технических работах, описание акционных предложений и скидок, новости о нововедениях предприятия, уведомления об изменениях в тарифных планах. Уведомления могут быть как общими для всех пользователей, так и индивидуальными, нацеленными на конкретные интересы потребителя.

Еще одной немаловажной составляющей является наличие раздела наиболее часто задаваемых вопросов (FAQ, Frequently Asked Questions). При помощи данного раздела пользователь сможет найти ответы на интересующие его вопросы без необходимости привлечении и персонала поддержки компании. В данной разделе могут быть размещены ответы на технические вопросы и вопросы оплаты, инструкции по настройке оборудования и другая полезная информация.

Все указанные выше фунции можно изобразить с помощью диаграмм потоков данных. Такие диаграммы является основным средством моделирования функциональных требований к проектируемой системе.

### 2.1.1 Построение диаграмм потоков данных

Одним из важнейших этапов проектирования ИС является построение диаграммы потоков данных. Диаграммы потоков данных (DFD) являются основным средством моделирования функциональных требований проектируемой системы. С их помощью эти требования разбиваются на функциональные компоненты (процессы) и представляются в виде сети, связанной потоками данных. Главная цель таких средств - продемонстрировать, как каждый процесс преобразует свои входные данные в выходные, а также выявить отношения между этими процессами.

Для изображения DFD традиционно используются две различные нотации: Йордона (Yourdon) и Гейна-Сарсона (Gane-Sarson).

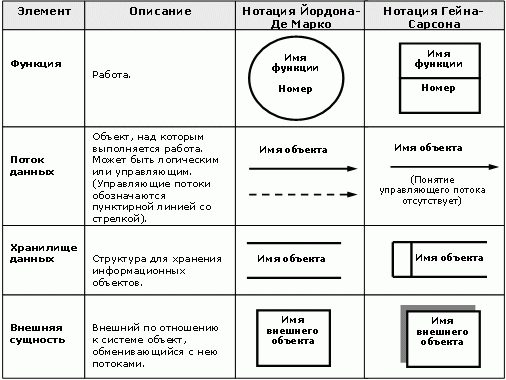


Рисунок 2.2 Сравнение типов диаграмм Йордона и Гейна-Сарсона

Далее при построении примеров будет использоваться нотация Йордона, все исключения будут предварительно оговариваться.

В основе данной методологии лежит построение модели анализируемой ИС - проектируемой или реально существующей. В соответствии с методологией модель системы определяется как иерархия диаграмм потоков данных, описывающих асинхронный процесс преобразования информации от ее ввода в систему до выдачи пользователю. Диаграммы верхних уровней иерархии (контекстные диаграммы) определяют основные процессы или подсистемы ИС с внешними входами и выходами. Они детализируются при помощи диаграмм нижнего уровня. Такая декомпозиция продолжается, создавая многоуровневую иерархию диаграмм, до тех пор, пока не будет достигнут такой уровень декомпозиции, на котором процесс становятся элементарными и детализировать их далее невозможно.

Первым шагом при построении иерархии DFD является построение контекстных диаграмм 0-го уровня. Обычно при проектировании относительно простых ИС строится единственная контекстная диаграмма со звездообразной топологией, в центре которой находится так называемый главный процесс (сама ИС), соединенный с приемниками и источниками информации, посредством которых с системой взаимодействуют пользователи и другие внешние системы.

Источники информации (внешние сущности) порождают информационные потоки (потоки данных), переносящие информацию к подсистемам или процессам. Те в свою очередь преобразуют информацию и порождают новые потоки, которые переносят информацию к другим процессам или подсистемам, накопителям данных или внешним сущностям - потребителям информации. Таким образом, основными компонентами диаграмм потоков данных являются:

* внешние сущности;
* системы/подсистемы;
* процессы;
* накопители данных;
* потоки данных.

Если же для сложной системы ограничиться единственной контекстной диаграммой, то она будет содержать слишком большое количество источников и приемников информации, которые трудно расположить на листе бумаги нормального формата, и кроме того, единственный главный процесс не раскрывает структуры распределенной системы.

Для сложных ИС строится иерархия контекстных диаграмм. При этом контекстная диаграмма 1-го уровня содержит набор подсистем, соединенных потоками данных. Контекстные диаграммы следующего уровня детализируют контекст и структуру подсистем до уровня отдельных задач.

Иерархия контекстных диаграмм определяет взаимодействие основных функциональных подсистем проектируемой ИС как между собой, так и с внешними входными и выходными потоками данных и внешними объектами (источниками и приемниками информации), с которыми взаимодействует ИС.

Разработка контекстных диаграмм решает проблему строгого определения функциональной структуры ИС на самой ранней стадии ее проектирования, что особенно важно для сложных многофункциональных систем, в разработке которых участвуют разные организации и коллективы разработчиков.

Для каждой подсистемы, присутствующей на контекстных диаграммах, выполняется ее детализация при помощи диаграммы DFD. Каждый процесс, в свою очередь, может быть детализирован при помощи отдельной диаграммы или миниспецификации.

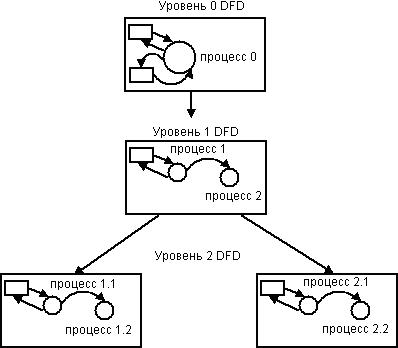


Рисунок 2.3. Уровни DFD диаграмм

При детализации должны выполняться следующие правила:

* правило балансировки — при детализации процесса дочерняя диаграмма в качестве внешних источников/приемников данных может иметь только те компоненты (подсистемы, процессы, внешние сущности, накопители данных), с которыми имеет информационную связь соответствующий процесс на родительской диаграмме;
* правило нумерации — при детализации процессов должна поддерживаться их иерархическая нумерация.
* правило семи — для того, чтобы диаграмма легко читалась, количество функций на диаграмме не должно быть больше семи.

Таким образом в зависимости от степени детализации процессов существует несколько уровней DF диаграмм. Для создания личного кабинета спроектируем Data Flow диаграммы 0-го и 1-го уровней (рис 2.1.1 и рис 2.1.2).

***Построение диаграммы потоков данных 0-го уровня***

На рисунке 2.1.1 мы изобразили главный процесс - «Управление телекоммуникационными услугами и счетами».. Этот процесс представляет собой преобразование входных потоков данных в выходные в соответствии с определенным алгоритмом и бизнес логикой предприятия. Он описывает основные функции проектируемой информационной системы.

Две внешние сущности «Клиент» и «Предприятие» представляют собой объекты, являющиеся источником или приемником системных данных.

Именуемыми стрелками мы проиллюстрировали потоки данных, которые являются в нашем макете абстракциями, используемыми для передачи информации из одной части системы в другую.

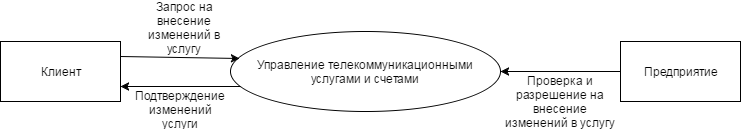


Рисунок 2.1.1. Data Flow диаграмма 0-го уровня

***Построение диаграммы потоков данных 1-го уровня***

Основной процесс можно детализировать, разбив его на несколько отдельных специфических подпроцессов. Для этого мы создали Data Flow диаграмму 1 уровня (рис. 2.1.2).

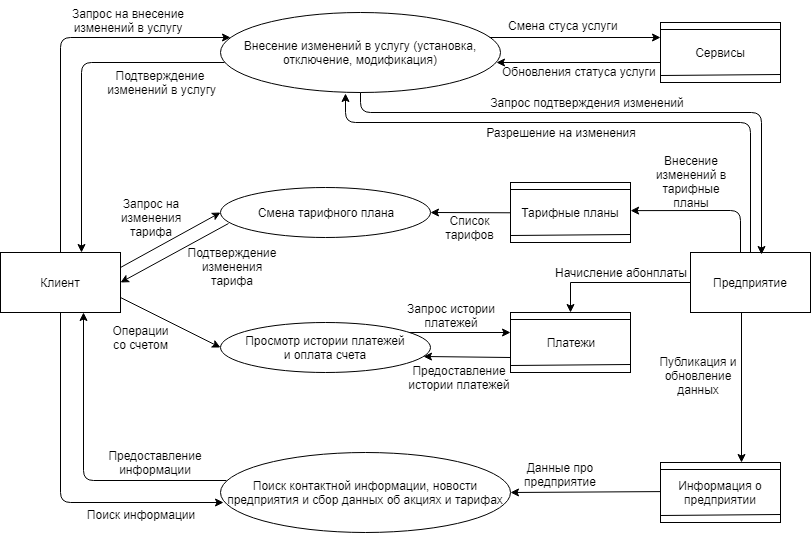


Рисунок 2.1.2. Диаграмма потоков данных 1-го уровня

На диаграмме потоков данных диаграмме 1-го уровня мы более подробно описали функции информационной системы. Данные функции были изображены на диаграмме в виде дополнительных процессов:

* внесение изенений в услугу (установка, отключение, модификация);
* смена тарифного плана;
* просмотр истории платежей и оплата счета;
* поиск контактной информации, новости предприятия и сбор данных об акциях и тарифах.

Согласно правилам построения DF диаграмм мы выделили хранилища данных «Сервисы», «Тарифные планы», «Платежи» и «Информация о предприятии», которые позволят нам на указанных участках определять данные, которые будут храниться в памяти между процессами. В этих абстрактных сущностях хранится информация, необходимая для функционирования проектируемой информационной системы.

Связь между упомянутыми выше элементами диаграммы будет осуществляться с помощью потоков данных, изображенных на диаграме в виде именуемых стрелок.

### 2.1.2 Логическая структура базы данных

Для поддержания БД в устойчивом состоянии используется ряд механизмов, которые получили обобщенное название средств поддержки целостности. Приведение структуры БД в соответствие этим ограничениям - это и есть нормализация.

В целом суть этих ограничений весьма проста: каждый факт, хранимый в БД, должен храниться один-единственный раз, поскольку дублирование может привести к несогласованности между копиями одной и той же информации. Следует избегать любых неоднозначностей, а также избыточности хранимой информации.

Схемой базы данных называется структура связей между полями и таблицами.

Нормализацией схемы базы данных называется процедура, производимая над базой данных с целью удаления в ней избыточности.

Выделяются шесть нормальных форм, пять из которых так и называются: первая, вторая, третья, четвертая, пятая нормальная форма, а также нормальная форма Бойса-Кодда, лежащая между третьей и четвертой.

Для реляционной модели данных разработано несколько нормализованных форм, три из которых являются основными.

База данных считается нормализованной, если ее таблицы представлены как минимум в третьей нормальной форме. Часто многие таблицы нормализуются до четвертой нормальной формы, иногда, наоборот, производится денормализация. Использования таблиц в пятой нормальной форме в реальных базах данных встречается редко.

Нормализация отношений – это формальный аппарат ограничений на формирование отношений, который позволяет устранить дублирование и потенциальную противоречивость хранимых данных, уменьшает трудозатраты на ведение БД. Процесс нормализации заключается в декомпозиции исходных отношений на более простые отношения. Цель нормализации – получение такого проекта БД, в котором «каждый факт появляется лишь в одном месте».

Теория нормализации основана на наличии зависимостей между атрибутами отношения. Основными видами зависимостей являются:

* функциональные;
* многозначные;
* транзитивные.

Базовым является понятие функциональной зависимости, поскольку на его основе формируются определения всех остальных видов зависимостей. Атрибут В функционально зависит от атрибута А, если каждому значению А соответствует в точности одно значение В. Математически функциональную зависимость В от А обозначают А ’ В. Это означает, что во всех кортежах с одинаковым значением атрибута А атрибут В будет иметь также одно и то же значение. При этом А и В могут быть составными, то есть состоять из двух и более атрибутов.

Зависимость, при которой каждый неключевой атрибут зависит от всего составного ключа и не зависит от его частей, называется полной функциональной зависимостью. Если атрибут А зависит от атрибута В, а атрибут В зависит от атрибута С (С ’ В ’ А), но обратная зависимость отсутствует, то зависимость А от С называется транзитивной.

Многозначная зависимость. Говорят, что один атрибут отношения многозначно определяет другой атрибут того же отношения, если для каждого значения первого атрибута существует множество соответствующих значений второго атрибута. Многозначные зависимости могут быть:

* один-ко-многим (1:М);
* многие-к-одному (М:1);
* многие-ко-многим (М:М).

Каждая ступень процесса нормализации приводит схему отношений в последовательные нормальные формы. Для каждой ступени имеются наборы ограничений. Выделяют следующую последовательность нормальных форм:

* первая нормальная форма (1НФ);
* вторая нормальная форма (2НФ);
* третья нормальная форма (3НФ);
* усиленная 3НФ или нормальная форма Бойса-Кодда (БКНФ);
* четвертая нормальная форма (4НФ);
* пятая нормальная форма (5НФ).

Для устранения потенциальной противоречивости и избыточности данных в отношениях, выявленных на этапе построения концептуальной модели приведём их к третьей нормальной форме. Нормальные формы определяются как совокупности требований, которыми должны удовлетворять отношения.

Переменная отношения находится в первой нормальной форме (1НФ) тогда и только тогда, когда в любом допустимом значении отношения каждый его кортеж содержит только одно значение для каждого из атрибутов.

Переменная отношения находится во второй нормальной форме (2НФ) тогда и только тогда, когда она находится в 1НФ, и каждый не ключевой атрибут неприводимо (функционально полно) зависит от ее потенциального ключа.

Переменная отношения находится в третьей нормальной форме (3НФ) тогда и только тогда, когда она находится во 2НФ, и отсутствуют транзитивные функциональные зависимости не ключевых атрибутов от ключевых.

Алгоритм приведения ненормализованных схем в 3НФ показан на рис. 18. На практике построение 3НФ в большинстве случаев является достаточным и приведением к ней процесс построения реляционной БД заканчивается.

Рисунок 18. Алгоритм приведения ненормализованных схем в 3НФ

Нормальные формы высших порядков (4НФ и 5НФ) представляют больший интерес для теоретических исследований, чем для практики проектирования БД.

Изобразим полученные отношения и их связи при помощи ER-диаграмм. (см. рис. 2.3 и рис 2.4).

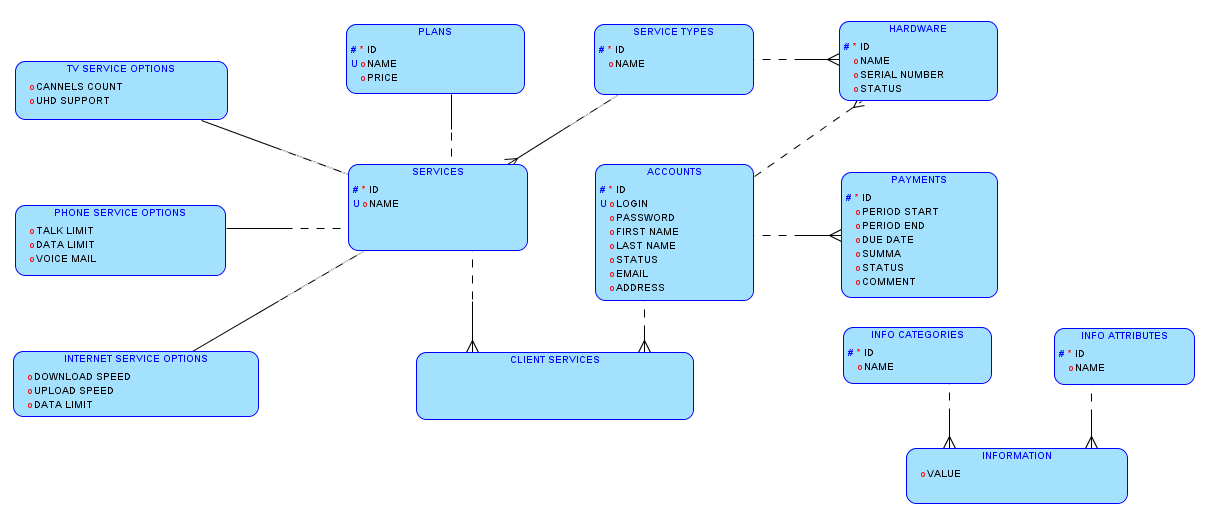


Рисунок 2.3 ER-диаграмма. Логическая модель.

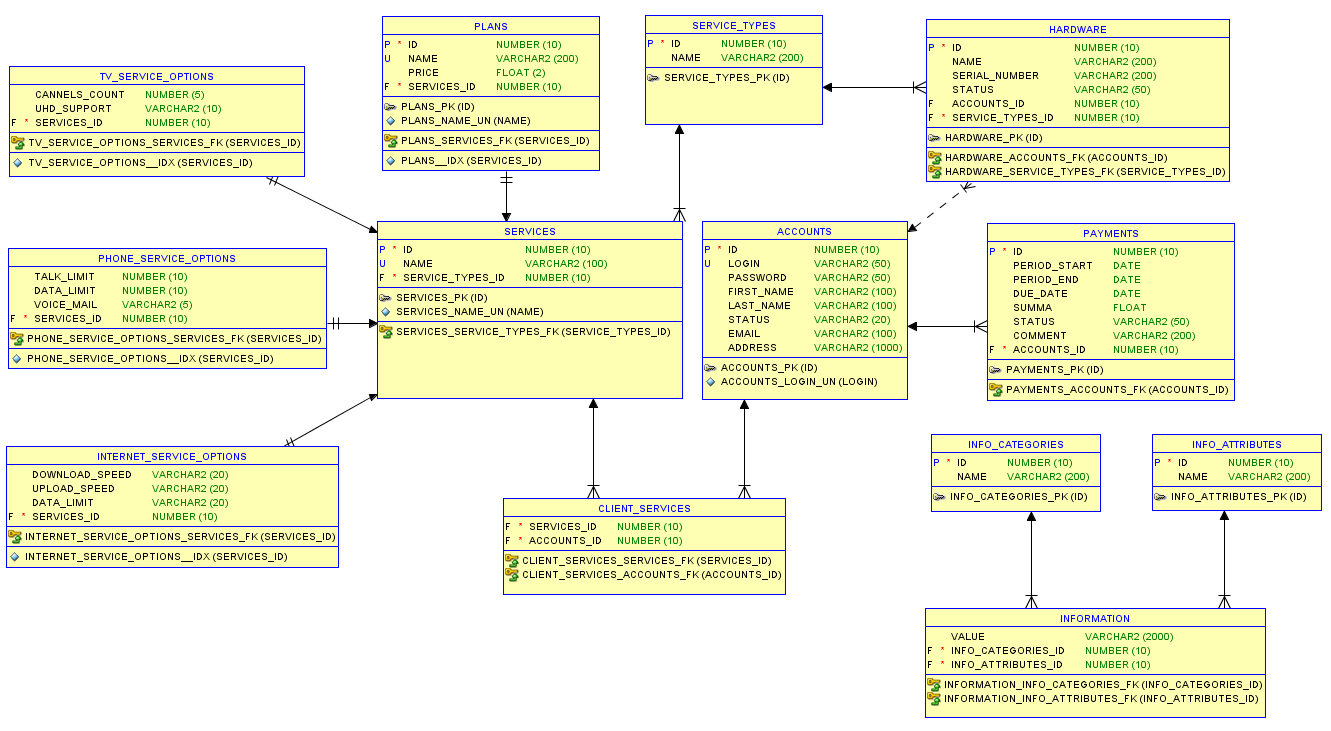


Рисунок 2.4 ER-диаграмма. Модель отношений.

Рассмотрим атрибуты, используемые в построенных сущностях, их тип и назначение (см. табл. 2.1, 2.2, 2.3, 2.4, 2.5).

Таблица 2.1 Атрибуты сущности Аккаунты (Accounts)

|  |  |
| --- | --- |
| Атрибут | Описание |
| id | Уникальный номер аккаунта пользователя, первичный ключ |
| login | Логин пользователя |
| password | Пароль пользователя |
| first\_name | Имя пользователя |
| last\_name | Фамилия пользователя |
| status | Текущий статус аккаунта |
| email | Адрес электроннной почты пользователя |
| address | Физический адрес пользователя |

Отношению «Аккаунты» соответствует полная 3НФ: id -> login, password, first\_name, last\_name, status, email, address.

TDB

Таблица 2.2 Атрибуты сущности «Тарифные Планы» (Plans)

|  |  |
| --- | --- |
| Атрибут | Описание |
| id | Уникальный номер тарифного плана, первичный ключ |
| name | Название тарифного плана |
| price | Месячная абонплата (грн.) |

Таблица 2.3 Атрибуты сущности «Типы Услуг» (Service Types)

|  |  |
| --- | --- |
| Атрибут | Описание |
| id | Уникальный номер типа сервиса, первичный ключ |
| name | Название типа сервиса (интернет, телевидение, телефония) |

Таблица 2.4 Атрибуты сущности «Сервисы» (Services)

|  |  |
| --- | --- |
| Атрибут | Описание |
| id | Уникальный номер сервиса, первичный ключ |
| name | Название сервиса (интернет, телевидение, телефония) |
| type\_id | Номер типа сервиса, внешний ключ |
| plan\_id | Номер тарифного плана, внешний ключ |

Таблица 2.5 Атрибуты сущности «Подключенные Сервисы Клиента» (Client Services)

|  |  |
| --- | --- |
| Атрибут | Описание |
| account\_id | Номер аккаунта пользователя, для которого подлючен сервис, внешний ключ |
| service\_id | Номер подключенного сервиса, внешний ключ |

Таблица 2.6 Атрибуты сущности «Платежи» (Payments)

|  |  |
| --- | --- |
| Атрибут | Описание |
| id | Уникальный номер платежа, первичный ключ |
| account\_id | Номер аккаунта пользователя, для которого предназначен платеж , внешний ключ |
| period\_start | Дата начала платежного периода |
| period\_end | Дата конца платежного периода |
| due\_date | Конечная дата платежа |
| summa | Сумма к оплате (грн.) |
| status | Статус платежа (оплачен/не оплачен) |

Таблица 2.7 Атрибуты сущности «Оборудование» (Hardware)

|  |  |
| --- | --- |
| Атрибут | Описание |
| id | Уникальный номер платежа, первичный ключ |
| account\_id | Номер аккаунта пользователя, к которому привязано оборудование, внешний ключ |
| service\_type\_id | Номер типа сервиса, для которого предназначено оборудование, внешний ключ |
| name | Название оборудования |
| serial\_number | Серийный номер оборудования |
| status | Статус оборудования (активный/неактивный). |

Таблица 2.9 Атрибуты сущности «Параметры Сервиса Интернет» (Internet Service Options)

|  |  |
| --- | --- |
| Атрибут | Описание |
| service\_id | Номер интернет сервиса, внешний ключ |
| download\_speed | Скорость скачивания данных (Мб/с) |
| upload\_speed | Скорость зарузки данных (Мб/с) |
| data\_limit | Суммарный месячный лимит трафика (ГБ) |

Таблица 2.10 Атрибуты сущности «Параметры Сервиса Телевидение» (TV Service Options)

|  |  |
| --- | --- |
| Атрибут | Описание |
| service\_id | Номер телевизионного сервиса, внешний ключ |
| channels\_count | Количество подключенных каналов |
| uhd\_support | Поддержка UHD (стандарт сверхвысокой чёткости) |

Таблица 2.11 Атрибуты сущности «Параметры Сервиса Телефония» (Phone Service Options)

|  |  |
| --- | --- |
| Атрибут | Описание |
| service\_id | Номер аккаунта пользователя, к которому привязано оборудование, внешний ключ |
| talk\_limit | Месячный лимит длительности разговора (мин) |
| data\_limit | Месячный лимит данных для мобильного интернет (ГБ) |
| voice\_mail | Наличие голосовой почты |

Таблица 2.8 Атрибуты сущности «Категории Информации» (Info Сategories)

|  |  |
| --- | --- |
| Атрибут | Описание |
| id | Уникальный номер категории информации, первичный ключ |
| name | Название категории информации |

Таблица 2.8 Атрибуты сущности «Параметры Информации» (Info Parameters)

|  |  |
| --- | --- |
| Атрибут | Описание |
| id | Уникальный номер параметра информации, первичный ключ |
| name | Имя параметра |

Таблица 2.8 Атрибуты сущности «Информация» (Information)

|  |  |
| --- | --- |
| Атрибут | Описание |
| attr\_id | Уникальный номер параметра, внешний ключ |
| info\_category\_id | Номер категории информации, внешний ключ |
| value | Значение параметра |

Проанализировав сущности, используемые в модели ИС, перейдём к реализации структуры БД. Для этого представим имена необходимых таблиц, атрибутов, типов, их назначения и ограничения (см. табл. 2.13).

Таблица 2.13 Структура БД

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Таблица** | **Столбец** | **Содержимое** | **Тип данных** | **Ключевой** | **Ограничения** |
| accounts | id | Уникальный номер аккаунта пользователя | NUMBER(10) | PK | NOT NULL |
| login | Логин пользователя | VARCHAR2(100) |  | NOT NULL |
| password | Пароль пользователя | VARCHAR2(100) |  |  |
| first\_name | Имя пользователя | VARCHAR2(100) |  |  |
| last\_name | Фамилия пользователя | VARCHAR2(100) |  |  |
| status | Текущий статус аккаунта | VARCHAR2(100) |  | NOT NULL, возможные значения: 'Active', 'Inactive' |
| email | Адрес электроннной почты пользователя | VARCHAR2(100) |  |  |
| address | Физический адрес пользователя | VARCHAR2(1000) |  |  |
| plans | id | Уникальный номер тарифного плана | NUMBER(10) | PK | NOT NULL |
| name | Название тарифного плана | VARCHAR2(100) |  | NOT NULL |
| price | Месячная абонплата (грн.) | FLOAT |  | Значение >= 0 |
| service\_types | id | Уникальный номер типа сервиса | NUMBER(10) | PK | NOT NULL |
| name | Название типа сервиса (интернет, телевидение, телефония) | VARCHAR2(200) |  | NOT NULL |
| services | id | Уникальный номер сервиса | NUMBER(10) | PK | NOT NULL |
| name | Название сервиса (интернет, телевидение, телефония) | VARCHAR2(200) |  | NOT NULL |
| type\_id | Номер типа сервиса | NUMBER(10) | PFK | NOT NULL |
| plan\_id | Номер тарифного плана | NUMBER(10) | PFK | NOT NULL |
| client\_services | account\_id | Номер аккаунта пользователя, для которого подлючен сервис | NUMBER(10) | PFK | NOT NULL |
| service\_id | Номер подключенного сервиса | NUMBER(10) | PFK | NOT NULL |
| payments | id | Уникальный номер платежа | NUMBER(10) | PK | NOT NULL |
| account\_id | Номер аккаунта пользователя, для которого предназначен платеж | NUMBER(10) | PFK | NOT NULL |
| period\_start | Дата начала платежного периода | DATE |  | NOT NULL,  period\_start <= period\_end |
| period\_end | Дата конца платежного периода | DATE |  | NOT NULL,  period\_start <= period\_end |
| due\_date | Конечная дата платежа | DATE |  | NOT NULL |
| summa | Сумма к оплате (грн.) | FLOAT |  | NOT NULL |
| status | Статус платежа (оплачен/не оплачен) | VARCHAR2(50) |  | NOT NULL, возможные значения: 'Paid', 'Not Paid' |
| hardware | id | Уникальный номер платежа | NUMBER(10) | PK | NOT NULL |
| account\_id | Номер аккаунта пользователя, к которому привязано оборудование | NUMBER(10) | FK |  |
| service\_type\_id | Номер типа сервиса, для которого предназначено оборудование | NUMBER(10) | FK | NOT NULL |
| name | Название оборудования | VARCHAR2(200) |  | NOT NULL |
| serial\_number | Серийный номер оборудования | VARCHAR2(200) |  | NOT NULL |
| status | Статус оборудования (активный/неактивный). | VARCHAR2(50) |  | NOT NULL,  возможные значения: 'Active', 'Inactive' |
| info\_categories | id | Уникальный номер категории информации | NUMBER(10) | PK | NOT NULL |
| name | Название категории информации | VARCHAR2(200) |  | NOT NULL |
| info\_attributes | id | Уникальный номер параметра информации | NUMBER(10) | PK | NOT NULL |
| name | Имя параметра | VARCHAR2(200) |  | NOT NULL |
| information | attr\_id | Номер параметра информации | NUMBER(10) | PFK | NOT NULL |
| category\_id | Номер категории информации | NUMBER(10) | PFK | NOT NULL |
| value | Значение параметра | VARCHAR2(2000) |  | NOT NULL |
| internet\_service\_options | service\_id | Номер интернет сервиса | NUMBER(10) | PFK | NOT NULL |
| download\_speed | Скорость скачивания данных (Мб/с) | VARCHAR2(20) |  | NOT NULL |
| upload\_speed | Скорость зарузки данных (Мб/с) | VARCHAR2(20) |  | NOT NULL |
| data\_limit | Суммарный месячный лимит трафика (ГБ) | VARCHAR2(20) |  | NOT NULL |
| tv\_service\_options | service\_id | Номер телевизионного сервиса | NUMBER(10) | PFK | NOT NULL |
| channels\_count | Количество подключенных каналов | NUMBER(10) |  | NOT NULL |
| uhd\_support | Поддержка UHD (стандарт сверхвысокой чёткости) | VARCHAR2(10) |  | NOT NULL,  возможные значения: 'On', 'Off' |
| phone\_service\_options | service\_id | Номер аккаунта пользователя, к которому привязано оборудование | NUMBER(10) | PFK | NOT NULL |
| talk\_limit | Месячный лимит длительности разговора (мин) | NUMBER(10) |  | NOT NULL |
| data\_limit | Месячный лимит данных для мобильного интернет (ГБ) | NUMBER(10) |  | NOT NULL |
| voice\_mail | Наличие голосовой почты | VARCHAR2(5) |  | NOT NULL,  возможные значения: 'On', 'Off' |

## 2.2 Физическая реализация базы данных

В качестве СУБД для базы данных было решено выбрать СУБД Oracle. Данное решение базируется на следующих факторах:

* высокая производительность;
* масштабируемость;
* поддержка 24/7;
* надежность (отлаженность процедур резервного копирования и восстановления);
* большое сообщество программистов и доступность документации.

Одним из существенных недостатков СУБД Oracle является то, что данный продукт в основном является платным. Но у Oracle также имеется бесплатная редакция СУБД Oracle Database. Это редакция eXpress Edition (XE). Она доступна для скачивания на сайте [https://www.oracle.com/](https://www.oracle.com/index.html) , абсолютна бесплатна для использования в бизнесе и имеет версии под Windows и Linux. Если бесплатная версия в будущем перестанет удовлетворять растущие потребности бизнеса, тогда есть возможность с минимальными затратами времени и ресурсов перейти на коммерческую версию Oracle СУБД, предлагающую более полный функционал.

### 2.2.1 Создание ключевых сущностей базы данных

***Создание таблиц***

Основными сущностями в БД являются таблицы. Ниже приведен SQL сценарий создания таблиц, инициализация ограничений и связей между таблицами.

Синтаксис создания таблицы в Oracle SQL указан ниже. Более подробную информацию по созданию таблиц и других объектов можно найти на сайте [Oracle](https://docs.oracle.com/).

CREATE TABLE table\_name

(

column1 datatype [ NULL | NOT NULL ],

column2 datatype [ NULL | NOT NULL ],

...

column\_n datatype [ NULL | NOT NULL ]

[ CONSTRAINT ]

);

Параметры и аргументы:

*table\_name* - имя создаваемой таблицы,

*column1, column2, ... column\_n* - имена столбцов в создаваемой таблице. Каждый столбец должен иметь тип данных (*datatype*). Столбец должен быть определен как «null» или «not null», и если это значение остается пустым, база данных предполагает «null» как значение по умолчанию.

Соответственно SQL код для создания таблиц для нашей БД будет выглядеть следующим образом:

CREATE TABLE accounts (

id NUMBER(10) NOT NULL,

login VARCHAR(50) NOT NULL,

password VARCHAR(50),

first\_name VARCHAR(100),

last\_name VARCHAR(100),

status VARCHAR(20) NOT NULL,

email VARCHAR(100),

address VARCHAR(1000)

, CONSTRAINT accounts\_pk PRIMARY KEY (id)

, CONSTRAINT accounts\_login\_uk UNIQUE (login)

, CONSTRAINT accounts\_status\_fixed\_values

CHECK (status IN ('Active', 'Inactive'))

);

CREATE TABLE plans (

id NUMBER(10) NOT NULL,

name VARCHAR(200) NOT NULL,

price FLOAT DEFAULT 0

, CONSTRAINT plans\_pk PRIMARY KEY (id)

, CONSTRAINT plans\_name\_uk UNIQUE (name)

, CONSTRAINT plans\_price\_not\_negative CHECK (price >= 0)

);

CREATE TABLE service\_types (

id NUMBER(10) NOT NULL,

name VARCHAR(200) NOT NULL

, CONSTRAINT service\_types\_pk PRIMARY KEY (id)

, CONSTRAINT service\_types\_name\_uk UNIQUE (name)

);

CREATE TABLE services (

id NUMBER(10) NOT NULL,

name VARCHAR(200) NOT NULL,

type\_id NUMBER(10),

plan\_id NUMBER(10)

, CONSTRAINT services\_pk PRIMARY KEY (id)

, CONSTRAINT services\_name\_uk UNIQUE (name)

, CONSTRAINT services\_type\_id\_fk

FOREIGN KEY (type\_id)

REFERENCES service\_types (id)

, CONSTRAINT services\_plan\_id\_fk

FOREIGN KEY (plan\_id)

REFERENCES plans (id)

);

CREATE TABLE client\_services (

account\_id NUMBER(10),

service\_id NUMBER(10)

, CONSTRAINT client\_services\_uk UNIQUE (account\_id, service\_id)

, CONSTRAINT client\_services\_acc\_id\_fk

FOREIGN KEY (account\_id)

REFERENCES accounts (id)

ON DELETE CASCADE

, CONSTRAINT client\_services\_serv\_id\_fk

FOREIGN KEY (service\_id)

REFERENCES services (id)

ON DELETE CASCADE

);

CREATE TABLE payments (

id NUMBER(10) NOT NULL,

account\_id NUMBER(10) NOT NULL,

period\_start DATE NOT NULL,

period\_end DATE NOT NULL,

due\_date DATE NOT NULL,

summa NUMBER(10) DEFAULT 0,

status VARCHAR2(50) DEFAULT 'Not Paid',

"COMMENT" VARCHAR2(200)

, CONSTRAINT payments\_pk PRIMARY KEY (id)

, CONSTRAINT payments\_account\_id\_fk

FOREIGN KEY (account\_id)

REFERENCES accounts (id)

, CONSTRAINT payments\_status\_values\_chk

CHECK (status IN ('Paid', 'Not Paid'))

, CONSTRAINT payments\_period\_chk

CHECK (period\_start <= period\_end)

);

CREATE TABLE hardware (

id NUMBER(10) PRIMARY KEY NOT NULL,

account\_id NUMBER(10) REFERENCES accounts (id),

service\_type\_id NUMBER(10) REFERENCES service\_types (id),

name VARCHAR2(200) NOT NULL,

serial\_number VARCHAR2(200) NOT NULL,

status VARCHAR2(50) DEFAULT 'Inactive'

, CONSTRAINT hw\_status\_values\_chk

CHECK (status in ('Active', 'Inactive'))

);

CREATE TABLE info\_categories (

id NUMBER(10) NOT NULL,

name VARCHAR(200) NOT NULL

, CONSTRAINT info\_categories\_pk PRIMARY KEY (id)

, CONSTRAINT info\_categories\_name\_uk UNIQUE (name)

);

CREATE TABLE info\_attributes (

id NUMBER(10) NOT NULL,

name VARCHAR(200) NOT NULL

, CONSTRAINT info\_attributes\_pk PRIMARY KEY (id)

, CONSTRAINT info\_attributes\_uk UNIQUE (name)

);

CREATE TABLE information (

attr\_id NUMBER(10) NOT NULL,

category\_id NUMBER(10) NOT NULL,

value VARCHAR2(500)

, CONSTRAINT information\_attr\_id\_fk

FOREIGN KEY (attr\_id)

REFERENCES info\_attributes (id)

, CONSTRAINT information\_cat\_id\_fk

FOREIGN KEY (category\_id)

REFERENCES info\_categories (id)

);

CREATE TABLE internet\_service\_options (

service\_id NUMBER(10) NOT NULL,

download\_speed VARCHAR(20) NOT NULL,

upload\_speed VARCHAR(20) NOT NULL,

data\_limit VARCHAR(20) NOT NULL

, CONSTRAINT int\_serv\_opt\_fk

FOREIGN KEY (service\_id)

REFERENCES services (id)

);

CREATE TABLE tv\_service\_options (

service\_id NUMBER(10) NOT NULL,

channels\_count NUMBER(5) NOT NULL,

uhd\_support VARCHAR(10) NOT NULL

, CONSTRAINT tv\_serv\_opt\_fk

FOREIGN KEY (service\_id)

REFERENCES services (id)

, CONSTRAINT tv\_serv\_opt\_uhd\_values

CHECK (uhd\_support IN ('On', 'Off'))

);

CREATE TABLE phone\_service\_options (

service\_id NUMBER(10) NOT NULL,

talk\_limit NUMBER(10) NOT NULL,

data\_limit NUMBER(10) NOT NULL,

voice\_mail VARCHAR2(5) DEFAULT 'Off'

, CONSTRAINT ph\_serv\_opt\_fk

FOREIGN KEY (service\_id)

REFERENCES services (id)

, CONSTRAINT ph\_serv\_opt\_vm\_values

CHECK (voice\_mail IN ('On', 'Off'))

);

***Создание последовательностей и триггеров для генерации первичных ключей***

Большинство из таблиц созданных выше имеют первичный ключ, который должен герерится автоматически при добавлении записи в таблицу. Для этого в Oracle SQL используются последовательности. Синтаксис создания последовательности приведен ниже.

CREATE SEQUENCE sequence\_name

MINVALUE value1

MAXVALUE value2

START WITH value3

INCREMENT BY value4

CACHE value5;

Параметры и аргументы:

*sequence\_name* - имя создаваемой последовательности,

*value1* – обозначает минимальное значение последовательности,

*value2* – обозначает максимальное значение последовательности,

*value3* – обозначает значение, с которого начинается отсчет последовательности,

*value4* – обозначает интервал между последующими номерами. Это целочисленное значение может быть любым положительным или отрицательным целым числом, но не может быть 0,

*value5* – обозначает сколько значений последовательности база данных предварительно распределяет и сохраняет в памяти для более быстрого доступа.

После создания последовательности, очередное ее значение можно получить следующим оператором:

sequence\_name.NEXTVAL;

Для того чтобы при вставке записей в таблицу этот операто вызывался автоматически, создадим триггер при помощи диалекта Oracle PL/SQL.

Cуществует несколько типов триггеров в зависимости от действия, по которому данный триггер срабатывает:

* BEFORE INSERT
* AFTER INSERT
* BEFORE UPDATE
* AFTER UPDATE
* BEFORE DELETE
* AFTER DELETE

Для отслеживания вставки записей в таблицу и генерации нового первичного ключа мы будем использовать триггер типа *BEFORE INSERT*.

Его синтаксис описан ниже:

CREATE [ OR REPLACE ] TRIGGER trigger\_name

BEFORE INSERT

ON table\_name

[ FOR EACH ROW ]

DECLARE

-- variable declarations

BEGIN

-- trigger code

EXCEPTION

WHEN ...

-- exception handling

END;

Параметры и аргументы:

*trigger\_name* - имя создаваемого триггера,

*BEFORE INSERT* - этот параметр указывает что триггер сработает перед операцией вставки данных в таблицу,

*table\_name* - имя таблицы, для которой создан триггер.

В качестве вызова кода в триггере мы будем вызывать следующий код, который возвращает очередное значение последовательности отдельно для каждой таблицы:

SELECT sequence\_name. NEXTVAL

INTO :new.id

FROM dual;

Таким образом полная версия скрипта для создания последовательностей и триггеров будет иметь вид:

CREATE SEQUENCE accounts\_seq

START WITH 1

INCREMENT BY 1

CACHE 100;

/

CREATE OR REPLACE TRIGGER accounts\_insert\_trigger

BEFORE INSERT ON accounts

FOR EACH ROW

BEGIN

SELECT accounts\_seq.nextval

INTO :new.id

FROM dual;

END;

/

CREATE SEQUENCE service\_types\_seq

START WITH 10

INCREMENT BY 1

CACHE 100;

/

CREATE OR REPLACE TRIGGER service\_types\_insert\_trigger

BEFORE INSERT ON service\_types

FOR EACH ROW

BEGIN

SELECT service\_types\_seq.nextval

INTO :new.id

FROM dual;

END;

/

CREATE SEQUENCE services\_seq

START WITH 100

INCREMENT BY 1

CACHE 100;

/

CREATE OR REPLACE TRIGGER services\_insert\_trigger

BEFORE INSERT ON services

FOR EACH ROW

BEGIN

SELECT services\_seq.nextval

INTO :new.id

FROM dual;

END;

/

CREATE SEQUENCE payments\_seq

START WITH 1000

INCREMENT BY 1

CACHE 100;

/

CREATE OR REPLACE TRIGGER payments\_insert\_trigger

BEFORE INSERT ON payments

FOR EACH ROW

BEGIN

SELECT payments\_seq.nextval

INTO :new.id

FROM dual;

END;

/

CREATE SEQUENCE hardware\_seq

START WITH 10000

INCREMENT BY 1

CACHE 100;

/

CREATE OR REPLACE TRIGGER hardware\_insert\_trigger

BEFORE INSERT ON hardware

FOR EACH ROW

BEGIN

SELECT hardware\_seq.nextval

INTO :new.id

FROM dual;

END;

/

CREATE SEQUENCE plans\_seq

START WITH 20000

INCREMENT BY 1

CACHE 100;

/

CREATE OR REPLACE TRIGGER plans\_insert\_trigger

BEFORE INSERT ON plans

FOR EACH ROW

BEGIN

SELECT plans\_seq.nextval

INTO :new.id

FROM dual;

END;

/

CREATE SEQUENCE info\_cat\_seq

START WITH 1

INCREMENT BY 1

CACHE 100;

/

CREATE OR REPLACE TRIGGER info\_cat\_insert\_trigger

BEFORE INSERT ON info\_categories

FOR EACH ROW

BEGIN

SELECT info\_cat\_seq.nextval

INTO :new.id

FROM dual;

END;

/

CREATE SEQUENCE info\_attr\_seq

START WITH 10

INCREMENT BY 1

CACHE 100;

/

CREATE OR REPLACE TRIGGER info\_attr\_insert\_trigger

BEFORE INSERT ON info\_attributes

FOR EACH ROW

BEGIN

SELECT info\_attr\_seq.nextval

INTO :new.id

FROM dual;

END;

/

После того как ключевые сущности БД созданы, необходимо применить изменения выполнив команду *COMMIT.* Затем нужно заполнить созданные таблицы данными. Полный листинг скриптов для вставки данных приведен в приложении А.

### 2.2.2 Реализация типичных запросов

TDB

## 2.3 Проектирование приложения

### 2.3.1 Описание архитектуры приложения

Так как разрабатываемое нами приложение «Личный кабинет» является веб-приложением, то необходимо разобрать суть этого понятия и его ключевые моменты.

Веб-приложение — клиент-серверное приложение, в котором клиент взаимодействует с сервером при помощи браузера, а за сервер отвечает веб-сервер. Логика веб-приложения распределена между сервером и клиентом, хранение данных осуществляется, преимущественно, на сервере, обмен информацией происходит по сети. Одним из преимуществ такого подхода является тот факт, что клиенты не зависят от конкретной операционной системы пользователя, поэтому веб-приложения являются межплатформенными службами.

Веб-приложение состоит из клиентской и серверной частей, тем самым реализуя технологию «клиент-сервер».

Клиентская часть реализует пользовательский интерфейс, формирует запросы к серверу и обрабатывает ответы от него.

Серверная часть получает запрос от клиента, выполняет вычисления, после этого формирует веб-страницу и отправляет её клиенту по сети с использованием протокола HTTP.

Само веб-приложение может выступать в качестве клиента других служб, например, базы данных или другого веб-приложения, расположенного на другом сервере.

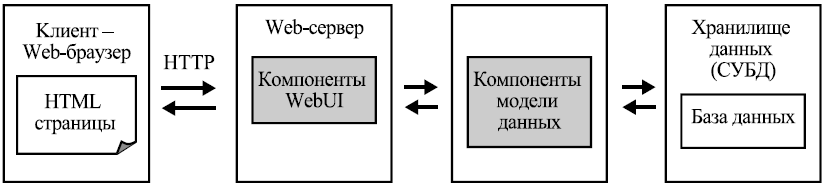


Рисунок 2.3.3 Схема функционирования веб-приложения

Для создания веб-приложений на стороне сервера используются разнообразные технологии и любые языки.

Основным языком программирования для реализации приложения «Личный кабинет» был выбран язык Java, который включает набор спецификаций Java Platform, Enterprise Edition (Java EE).

Платформа J2EE приспособлена для разработки многоуровневых Web-приложений. При работе с такими приложениями пользователь формирует свои запросы, заполняя HTML-формы в браузере, который упаковывает их в HTTP-сообщения и пересылает Web-серверу. Web-сервер передает эти сообщения Web-компонентам, выделяющим из них исходные запросы пользователя и передающим их для обработки компонентам EJB. Результаты работы EJB компонентов превращаются Web-компонентами в динамически генерируемые HTML-страницы, и отправляются обратно пользователю, представая перед ним в окне браузера. Аплеты используются там, где нужен более функциональный интерфейс, чем стандартные формы и страницы HTML.

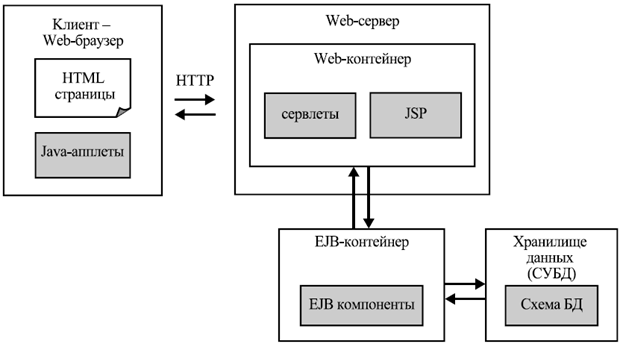


Рис. 13.1. Типовая архитектура J2EE приложения.

Выделены компоненты, разрабатываемые вручную. Таким образом, приложения на базе J2EE строятся с использованием трех основных архитектурных стилей:

***Многоуровневая система.***

Самые крупные подсистемы организованы как уровни, решающие различные задачи.

Интерфейс взаимодействия с внешней средой, включая пользователей, реализуется при помощи Web-компонентов.

Уровень бизнес-логики и модели данных реализуется при помощи EJB-компонентов.

Уровень управления ресурсами строится на основе коммерческих систем управления базами данных (СУБД). Можно также подключать другие виды ресурсов, для которых имеется реализация интерфейса поставщика служб J2EE (J2EE service provider interface, J2EE SPI).

***Независимые компоненты.***

Первые два уровня построены из отдельных компонентов, каждый из которых имеет собственную область ответственности, но может привлекать для решения частных задач другие компоненты.

***Данные–представление–обработка (MVC).***

Работа компонентов в рамках обработки группы тесно связанных запросов организуется по образцу MVC. При этом сервлеты и обработчики Web-событий служат обработчиками, компоненты JSP — представлением, а компоненты EJB — моделью данных.

Как указано выше, для реализации базового пользовательского интерфейса приложения использовались компоненты JSP.

JSP (JavaServer Pages) — технология, позволяющая веб-разработчикам создавать содержимое, которое имеет как статические, так и динамические компоненты. Страница JSP содержит текст двух типов: статические исходные данные, которые могут быть оформлены в одном из текстовых форматов HTML, SVG, WML, или XML, и JSP- элементы, которые конструируют динамическое содержимое. Кроме этого могут использоваться библиотеки JSP-тегов, а также EL (Expression Language), для внедрения Java-кода в статичное содержимое JSP-страниц.

Для того чтобы сделать интерфейс более гибким, функциональным и визуально привлекательным, использовалась front-end библиотека Bootstrap.

Bootstrap — свободный набор инструментов для создания сайтов и веб-приложений. Включает в себя HTML- и CSS-шаблоны оформления для типографики, веб-форм, кнопок, меток, блоков навигации и прочих компонентов веб-интерфейса, включая JavaScript-расширения. Данный фреймворк позволяет быстро прототипировать свои идеи или создайте свое приложение с помощью наших готовых переменных, гибкой сетчатой системы, обширных готовых компонентов и мощных плагинов, основанных на jQuery.

### 2.3.2 Создание UML диаграмм

Центральное место в ООАП занимает разработка логической модели системы в виде диаграммы классов. Для ее создания приятно использовать язык моделирования UML.

UML (Unified Modeling Language — унифицированный язык моделирования) — язык графического описания для объектного моделирования в области разработки программного обеспечения, моделирования бизнес-процессов, системного проектирования и отображения организационных структур.

Нотация UML предоставляет широкие возможности для отображения дополнительной информации (абстрактные операции и классы, стереотипы, общие и частные методы, детализированные интерфейсы, параметризованные классы). При этом возможно использование графических изображений для ассоциаций и их специфических свойств, таких как отношение агрегации, когда составными частями класса могут выступать другие классы.

Диаграмма классов (class diagram) служит для представления статической структуры модели системы в терминологии классов объектно-ориентированного программирования. Диаграмма классов может отражать, в частности, различные взаимосвязи между отдельными сущностями предметной области, такими как объекты и подсистемы, а также описывает их внутреннюю структуру и типы отношений. На данной диаграмме не указывается информация о временных аспектах функционирования системы. С этой точки зрения диаграмма классов является дальнейшим развитием концептуальной модели проектируемой системы.

Диаграмма классов представляет собой некоторый граф, вершинами которого являются элементы типа «классификатор», которые связаны различными типами структурных отношений. Следует заметить, что диаграмма классов может также содержать интерфейсы, пакеты, отношения и даже отдельные экземпляры, такие как объекты и связи. Когда говорят о данной диаграмме, имеют в виду статическую структурную модель проектируемой системы. Поэтому диаграмму классов принято считать графическим представленном таких структурных взаимосвязей логической модели системы, которые не зависят или инвариантны от времени.

Диаграмма классов состоит из множества элементов, которые в совокупности отражают декларативные знания о предметной области. Эти знания интерпретируются в базовых понятиях языка UML, таких как классы, интерфейсы и отношения между ними и их составляющими компонентами. При этом отдельные компоненты этой диаграммы могут образовывать пакеты для представления более общей модели системы. Если диаграмма классов является частью некоторого пакета, то ее компоненты должны соответствовать элементам этого пакета, включая возможные ссылки на элементы из других пакетов.

В общем случае пакет статической структурной модели может быть представлен в виде одной или нескольких диаграмм классов. Декомпозиция некоторого представления на отдельные диаграммы выполняется с целью удобства и графической визуализации структурных взаимосвязей предметной области. При этом компоненты диаграммы соответствуют элементам статической семантической модели. Модель системы, в свою очередь, должна быть согласована с внутренней структурой классов, которая описывается на языке UML.

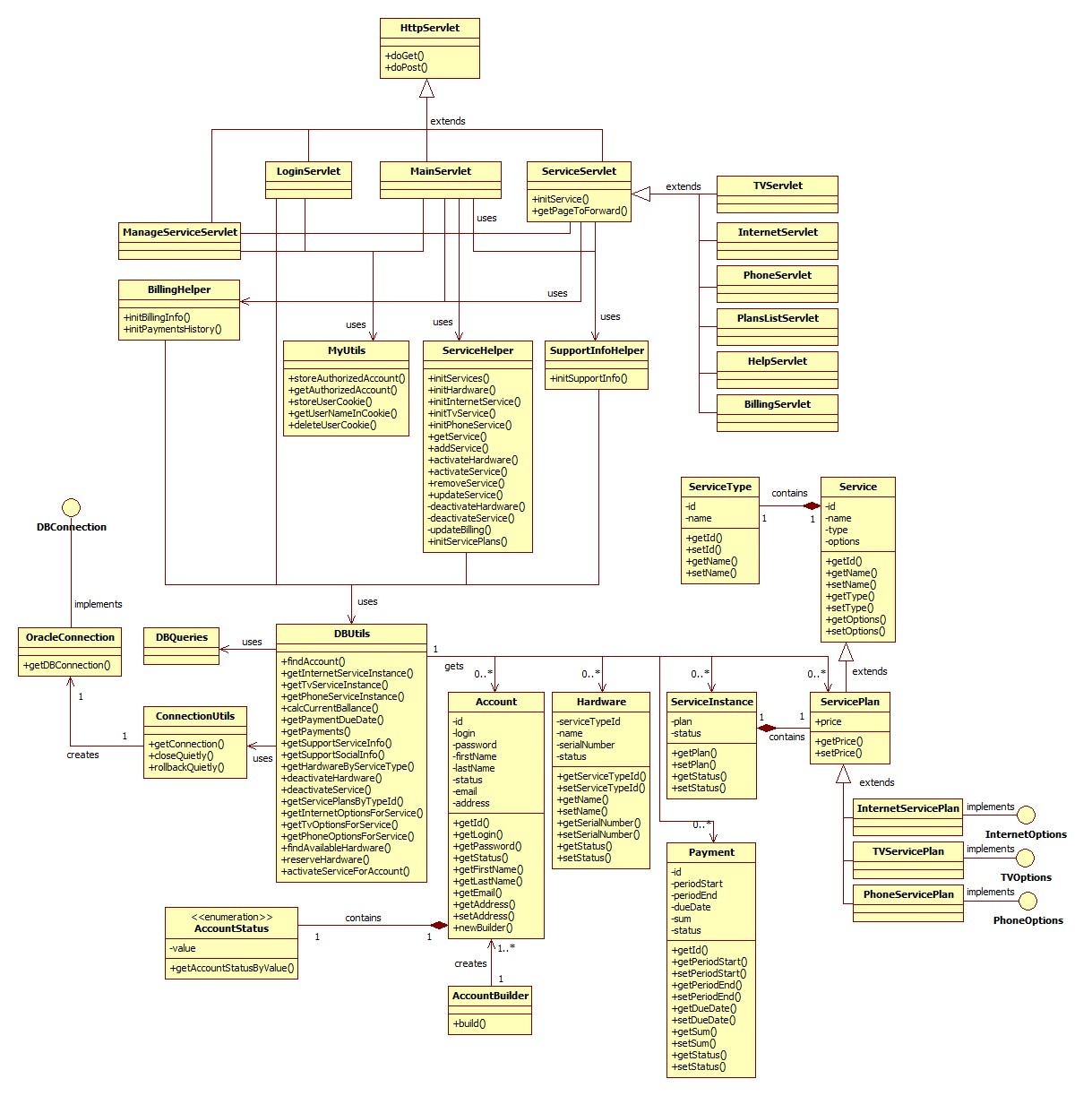


Рисунок # Диаграмма классов приложения «Личный кабинет»

Диаграмма последовательности — диаграмма, на которой для некоторого набора объектов на единой временной оси показан жизненный цикл какого-либо определённого объекта (создание-деятельность-уничтожение некой сущности) и взаимодействие актеров (действующих лиц) ИС в рамках какого-либо определённого прецедента (отправка запросов и получение ответов).

Основными элементами диаграммы последовательности являются обозначения объектов (прямоугольники с названиями объектов), вертикальные «линии жизни», отображающие течение времени, прямоугольники, отражающие деятельность объекта или исполнение им определенной функции (прямоугольники на пунктирной «линии жизни»), и стрелки, показывающие обмен сигналами или сообщениями между объектами. На данной диаграмме объекты располагаются слева направо.

На диаграмме ниже (рисунок) показан процесс управления сервисами. Когда клиент выбирает опцию установки/модификации или отключения услуги в приложении «Личный кабинет», посылается сообщение во внешние системы (External Inventory, Billing) для активации оборудования и расчета платежей соответственно. После завершения этих процессов и получения ответа от этих систем, текущий статус сервиса записывается в базу данных приложения и обновляется на странице пользователя.

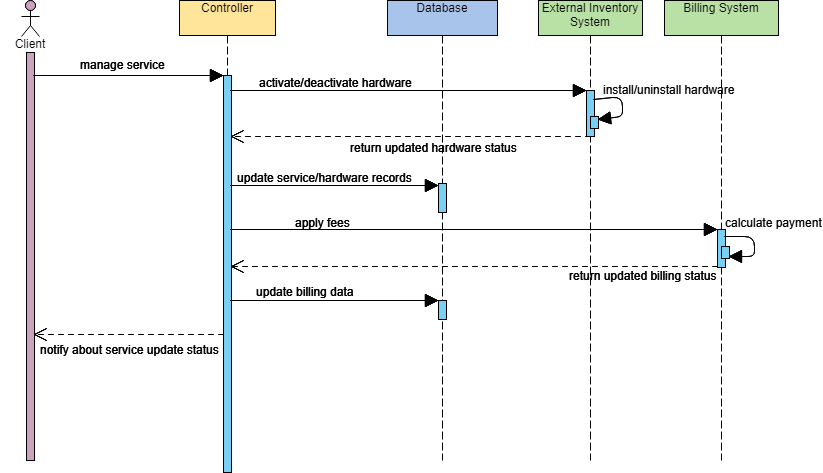


Рисунок № Диаграмма последовательности. Управление сервисами.

<https://online.visual-paradigm.com/>

На следующей диаграмме последовательности изображены основные функции, которые доступны пользователям приложениния: проверка статуса услуг, получение информации о балансе и истории платежей, получение информации о продуктах компании.

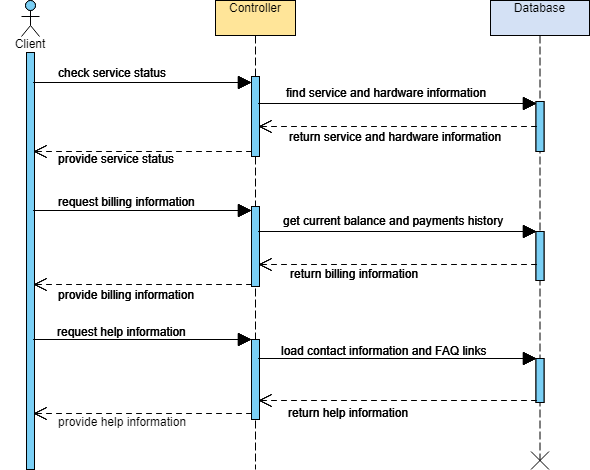


Рисунок № Диаграмма последовательности. Основные функции приложения.

# 3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

## 3.1 Создание и настройка приложения

TDB

### 3.1.1 Выбор среды разработки для разработки приложения

Интегрированная среда разработки (Integrated development environment, IDE) — это программа, которая содержит в себе инструменты для разработки программного обеспечения.

***Среда разработки для работы с базой данных***

Для работы с SQL скриптами и внесения изменений в БД была использована программа Oracle SQL Developer.

Oracle SQL Developer - это бесплатная интегрированная среда разработки, которая упрощает разработку и управление базами данных Oracle как в традиционном, так и в облачном развертывании. SQL Developer предлагает полную сквозную разработку PL/SQL-приложений, рабочий лист для запуска запросов и сценариев, консоль администратора баз данных для управления базой данных, интерфейс отчетов, полное решение для моделирования данных и платформу миграции для перемещения сторонних баз данных.

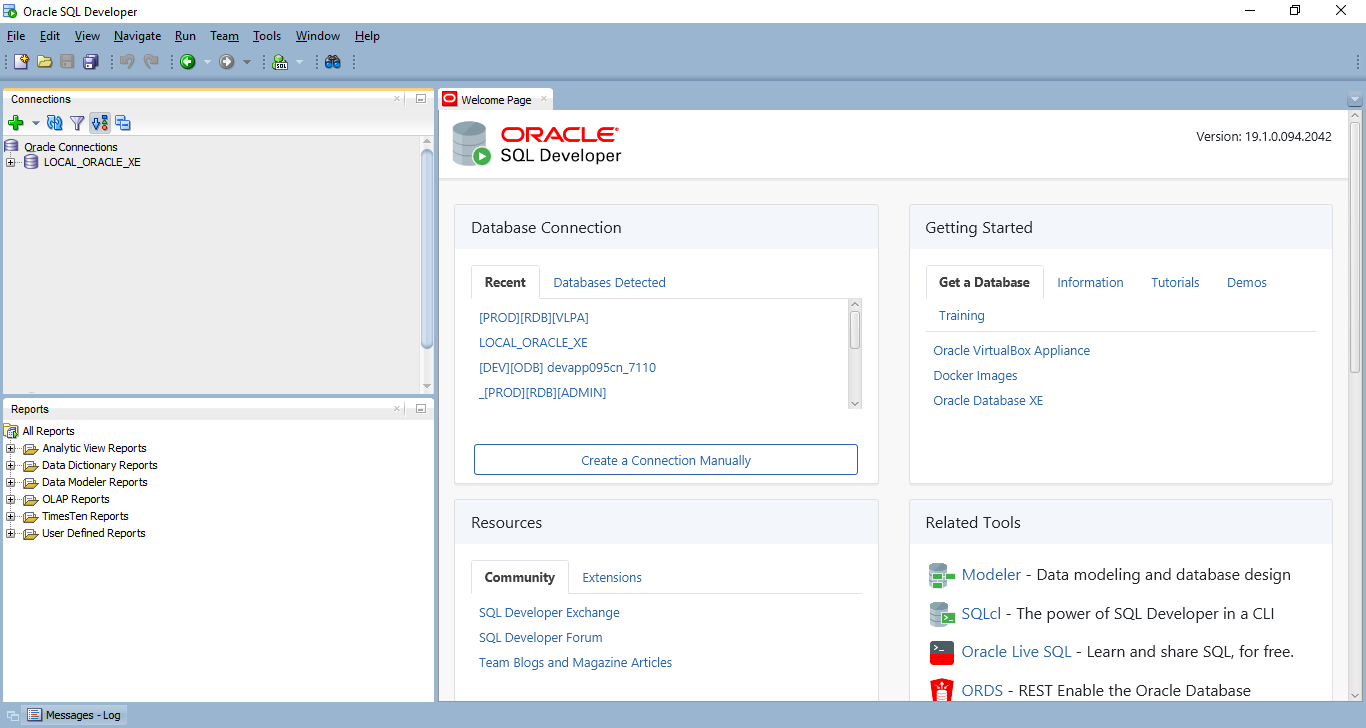


Рисунок 2.4 Интерфейс программы Oracle SQL Developer

Корпорация Oracle предоставляет продукт бесплатно. Сама среда написана на языке программирования Java, работает на всех платформах, где доступна среда выполнения Java SE.

***Среда разработки для Java***

Современная среда разработки на языке Java включает в себя:

* текстовый редактор с подсветкой кода;
* компилятор или интерпретатор;
* браузер классов, инспектор объектов и диаграмму иерархии классов;
* средства автоматизации сборки;
* отладчик;
* средства для интеграции с системами управления версиями;
* инструменты для упрощения конструирования графического интерфейса пользователя.

Основными IDE для написания кода на языке программирования Java являются IntelliJ IDEA, Eclipse, NetBeans, JDeveloper.

Наш выбор остановился на IntelliJ IDEA, так как она предоставляет множество инструметов для продуктивной работы и идеально подходит для создания корпоративных, мобильных и веб-приложений.

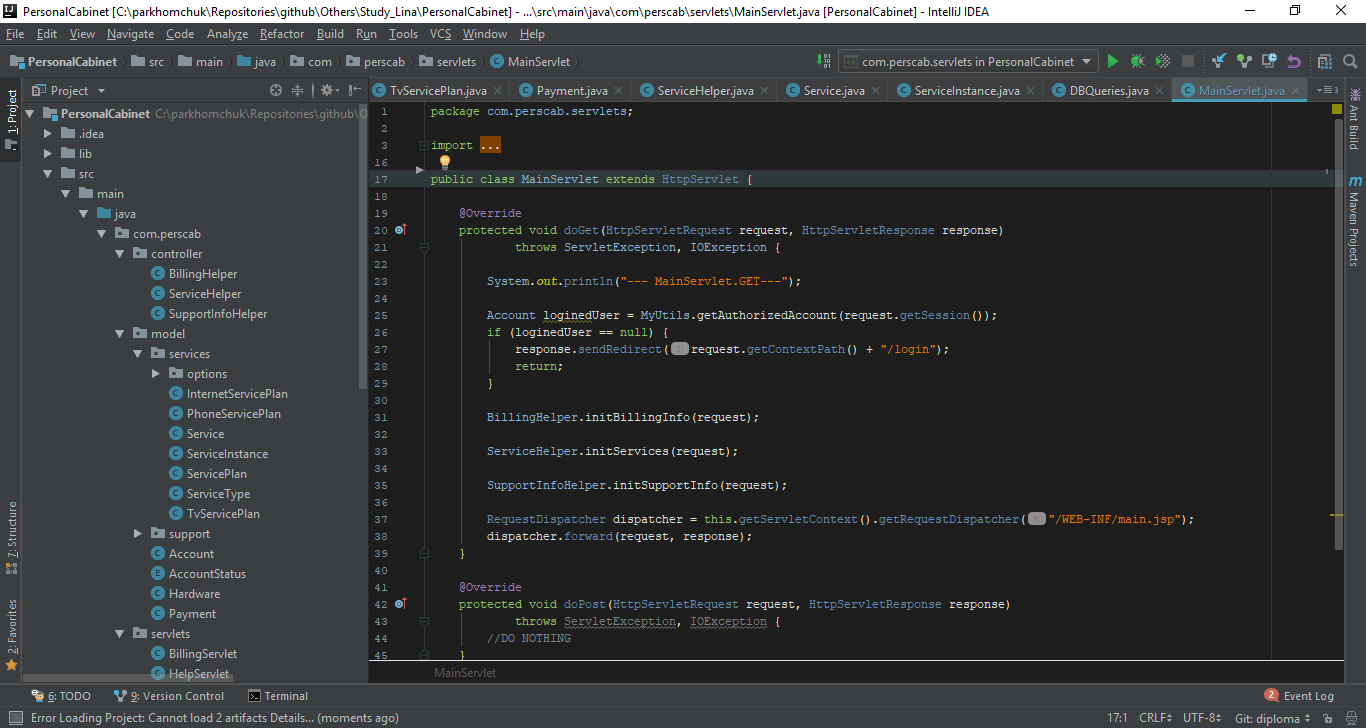


Рисунок № Интерфейс среды разработки IntelliJ IDEA

IntelliJ IDEA распространяется в двух версиях, одна из которых бесплатная - Free Community Edition. В платной же версии присутсвует поддержка фреймворков Spring (Spring MVC framework, Spring Security, Spring Boot, Spring Integration и т. д.), Node.js, Angular React, Grails, возможность использовать дополнительные языки (javascript, typescript, coffeescript) и взаимодействовать почти со семи популярными серверами (Tomcat, TomEE, GlassFish, JBoss, WildFly, Weblogic, WebSphere, Geronimo, Virgo и т. д.).

Для реализаци нашего приложения будет достаточно Free Community Edition версии для IntelliJ IDEA, так как она содержит все необходимые средства для разработки проекта.

### 3.1.2 Настройка системы сборки Maven

Maven — инструмент для автоматизации сборки проектов на основе описания их структуры в файлах на языке POM (англ. Project Object Model), являющемся подмножеством XML. Данный инструмент используется в разработке приложений на Java, хотя есть плагины для интеграции с C/C++, Ruby, Scala, PHP и другими языками.

Одна из главных особенностей фреймворка — декларативное описание проекта. Это значит, что разработчику не нужно уделять внимание каждому аспекту сборки — все необходимые параметры настроены по умолчанию. Изменения нужно вносить лишь в том объёме, в котором программист хочет отклониться от стандартных настроек.

Ещё одно достоинство проекта — гибкое управление зависимостями. Maven умеет подгружать в свой локальный репозиторий сторонние библиотеки, выбирать необходимую версию пакета, обрабатывать транзитивные зависимости.

Мы будем работать с Maven с помощью командной строки, однако этот фреймворк также интегрирован в Eclipse, IntelliJ IDEA, NetBeans и другие IDE.

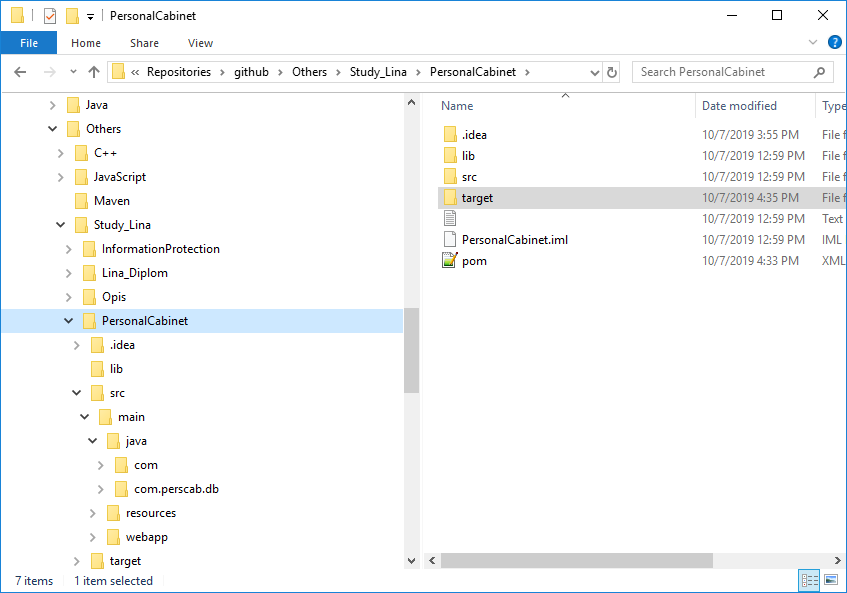


Рисунок № Структура Maven проекта приложения «Личный кабинет»

Конфигурационный POM файл для приложения «Личный кабинет» будет выглядеть следующим образом:

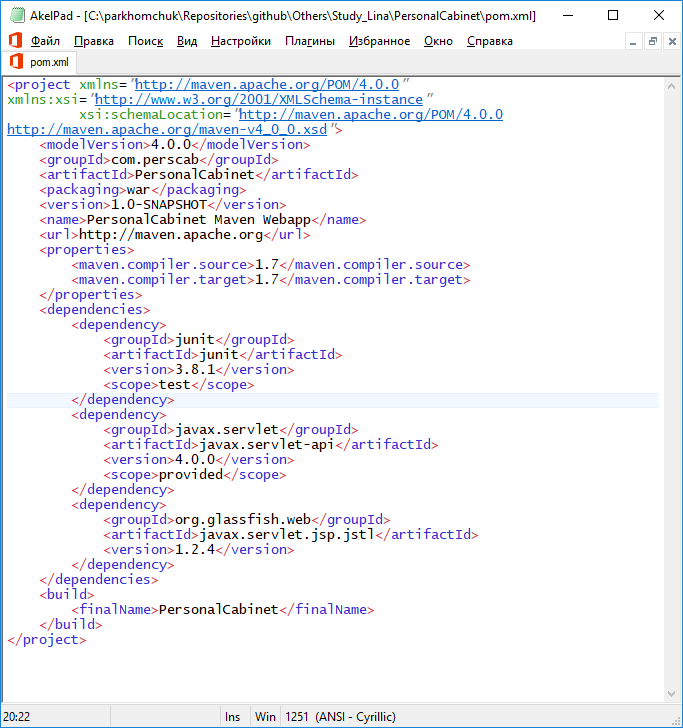


Рисунок № Конфигурационный POM файл

Запуск сборки проекта осуществляется командой “mvn clean install”. Описание жизненного цикла сборки, команд и доступных параметров можно найти на официально сайте [Apache Maven](http://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html).

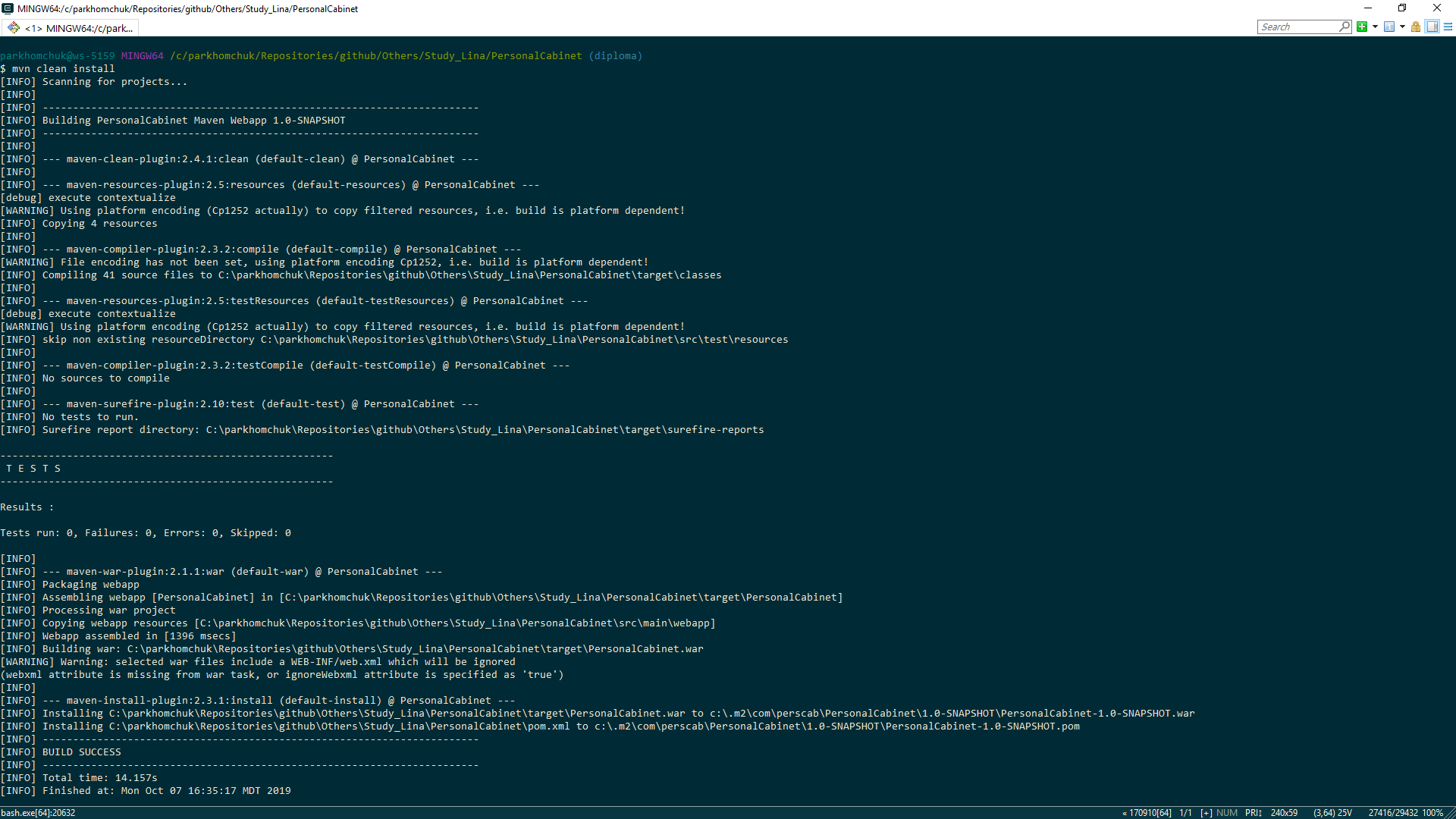


Рисунок № Результат сборки Maven проекта

После успешного завершения будет создан файл *PersonalCabinet.war*, который по сути является веб-архивом со всеми необходимыми ресурсами, который нужно будет скопировать на веб-сервер.

### 3.1.3 Использования системы контроля версий

Система контроля версий (от англ. Version Control System, VCS или Revision Control System) — программное обеспечение для облегчения работы с изменяющейся информацией. Система контроля версий (СКВ) позволяет хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение, и многое другое.

Такие системы наиболее широко используются при разработке программного обеспечения для хранения исходных кодов разрабатываемой программы. Однако они могут с успехом применяться и в других областях, в которых ведётся работа с большим количеством непрерывно изменяющихся электронных документов. Управление версиями используется в инструментах конфигурационного управления (Software Configuration Management Tools).

СКВ даёт возможность возвращать отдельные файлы к прежнему виду, возвращать к прежнему состоянию весь проект, просматривать происходящие со временем изменения, определять, кто и когда вносил изменения в модуль, кто и когда внёс в код какую-то ошибку и многое другое.

СКВ отслеживает изменения в файлах, предоставляет возможности для создания новых и слияние существующих ветвей проекта, производит контроль доступа пользователей к проекту, позволяет откатывать исправления и определять кто, когда и какие изменения вносил в проект.

***Локальные системы контроля версий***

Многие предпочитают контролировать версии, просто копируя файлы в другой каталог (как правило добавляя текущую дату к названию каталога). Такой подход очень распространён, потому что прост, но он и чаще даёт сбои. Очень легко забыть, что ты не в том каталоге, и случайно изменить не тот файл, либо скопировать файлы не туда, куда хотел, и затереть нужные файлы.

Чтобы решить эту проблему, были разработаны локальные СКВ с простой базой данных, в которой хранятся все изменения нужных файлов (рисунок 1-1).



Рисунок 1-1. Схема локальной СКВ.

Одной из наиболее популярных СКВ такого типа является rcs, которая до сих пор устанавливается на многие компьютеры. Даже в современной операционной системе Mac OS X утилита rcs устанавливается вместе с Developer Tools. Эта утилита основана на работе с наборами патчей между парами версий, которые хранятся в специальном формате на диске. Это позволяет пересоздать любой файл на любой момент времени, последовательно накладывая патчи.

***Централизованные системы контроля версий***

Следующей основной проблемой является необходимость сотрудничать с разработчиками за другими компьютерами. Чтобы решить её, были созданы централизованные системы контроля версий (ЦСКВ). В таких системах, например CVS, Subversion и Perforce, есть центральный сервер, на котором хранятся все файлы под версионным контролем, и ряд клиентов, которые получают копии файлов из него. Много лет это было стандартом для систем контроля версий (см. рис. 1-2).



Рисунок 1-2. Схема централизованного контроля версий.

Такой подход имеет множество преимуществ, особенно над локальными СКВ. К примеру, все знают, кто и чем занимается в проекте. У администраторов есть чёткий контроль над тем, кто и что может делать, и, конечно, администрировать ЦСКВ намного легче, чем локальные базы на каждом клиенте.

Однако при таком подходе есть и несколько серьёзных недостатков. Наиболее очевидный — централизованный сервер является уязвимым местом всей системы. Если сервер выключается на час, то в течение часа разработчики не могут взаимодействовать, и никто не может сохранить новой версии своей работы. Если же повреждается диск с центральной базой данных и нет резервной копии, вы теряете абсолютно всё — всю историю проекта, разве что за исключением нескольких рабочих версий, сохранившихся на рабочих машинах пользователей. Локальные системы контроля версий подвержены той же проблеме: если вся история проекта хранится в одном месте, вы рискуете потерять всё.

***Распределённые системы контроля версий***

Описанные выше проблемы решаются при помощи распределённых систем контроля версий (РСКВ). В таких системах как Git, Mercurial, Bazaar или Darcs клиенты не просто выгружают последние версии файлов, а полностью копируют весь репозиторий. Поэтому в случае, когда "умирает" сервер, через который шла работа, любой клиентский репозиторий может быть скопирован обратно на сервер, чтобы восстановить базу данных. Каждый раз, когда клиент забирает свежую версию файлов, он создаёт себе полную копию всех данных (рисунок 1-3).



Рисунок 1-3. Схема распределённой системы контроля версий.

Кроме того, в большей части этих систем можно работать с несколькими удалёнными репозиториями, таким образом, можно одновременно работать по-разному с разными группами людей в рамках одного проекта. Так, в одном проекте можно одновременно вести несколько типов рабочих процессов, что невозможно в централизованных системах.

Основным понятием РСКВ является репозиторий (*repository*) – специальное хранилище файлов и папок проекта, изменения в которых отслеживаются. В распоряжении разработчика имеется так называемая “рабочая копия” (*working copy*) проекта, с которой он непосредственно работает. Рабочую копию необходимо периодически синхронизировать с репозиторием, эта операция предполагает отправку в него изменений, которые пользователь внес в свою рабочую копию (такая операция называется *commit*) и актуализацию рабочей копии, в процессе которой к пользователю загружается последняя версия из репозитория (этот процесс носит название *update*).

Для приложения «Личный кабинет» будем использовать одну из самых популярных РСКВ – Git.

Git – распределенная система контроля версий, разработанная Линусом Торвальдсем для работы над ядром операционной системы Linux. Среди крупных проектов, в рамках которых используется Git, можно выделить ядро Linux, Qt, Android. Git свободен и распространяется под лицензией GNU GPL 2 и доступен практически на всех операционных системах. Благодаря ряду достоинств (высокая скорость работы, возможность интеграции с другими VCS, удобный интерфейс) и очень активному сообществу, сформировавшемуся вокруг этой системы, Git вышел в лидеры рынка распределенных систем контроля версий.

В качестве репозитория был выбран GitHub (рисунок). GitHub — крупнейший веб-сервис для хостинга IT-проектов и их совместной разработки. Веб-сервис основан на системе контроля версий Git и разработан на Ruby on Rails и Erlang компанией GitHub, Inc. Сервис бесплатен для проектов с открытым исходным кодом и (с 2019 года) небольших частных проектов, предоставляя им все возможности (включая SSL), а для крупных корпоративных проектов предлагаются различные платные тарифные планы.

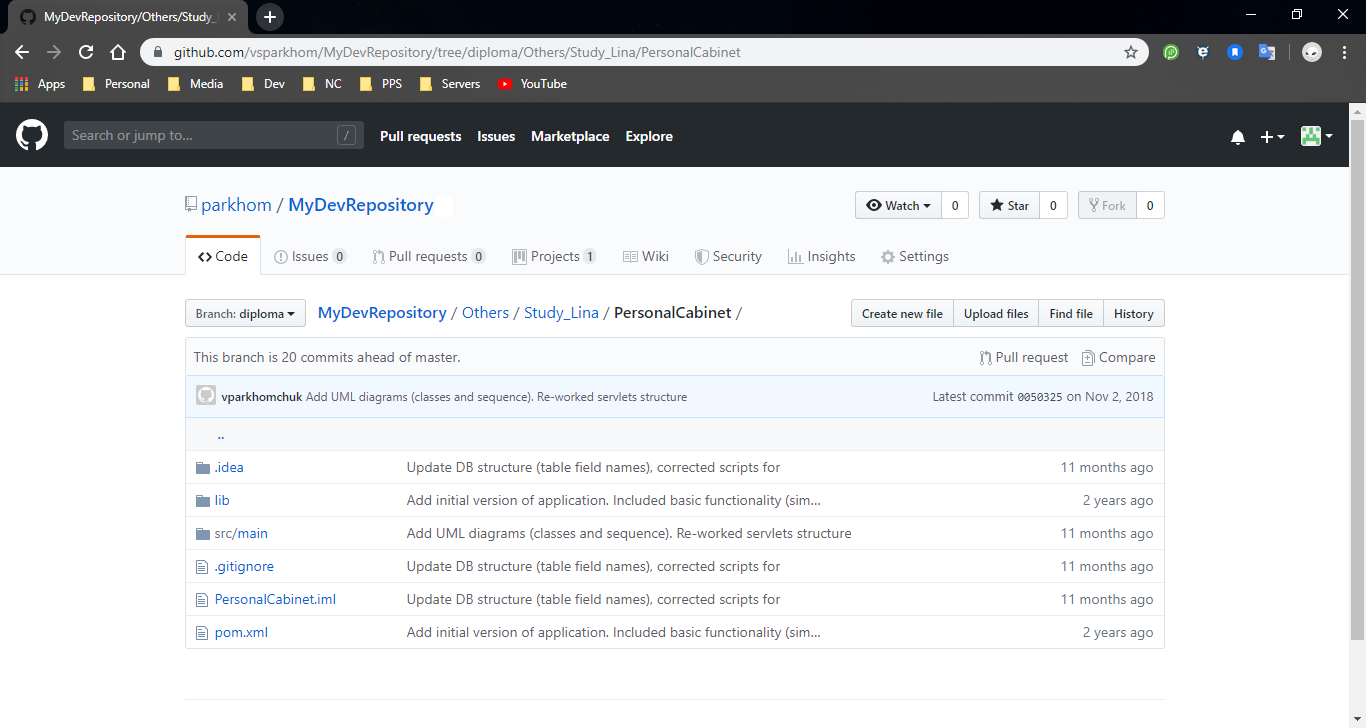


Рисунок 1-3. Репозиторий GitHub

Далее в нужно создать рабочую копию репозитория и создать отдельную ветку для изменений (рисунок). При последующей разработке проекта все изменения будут коммититься в эту ветку.

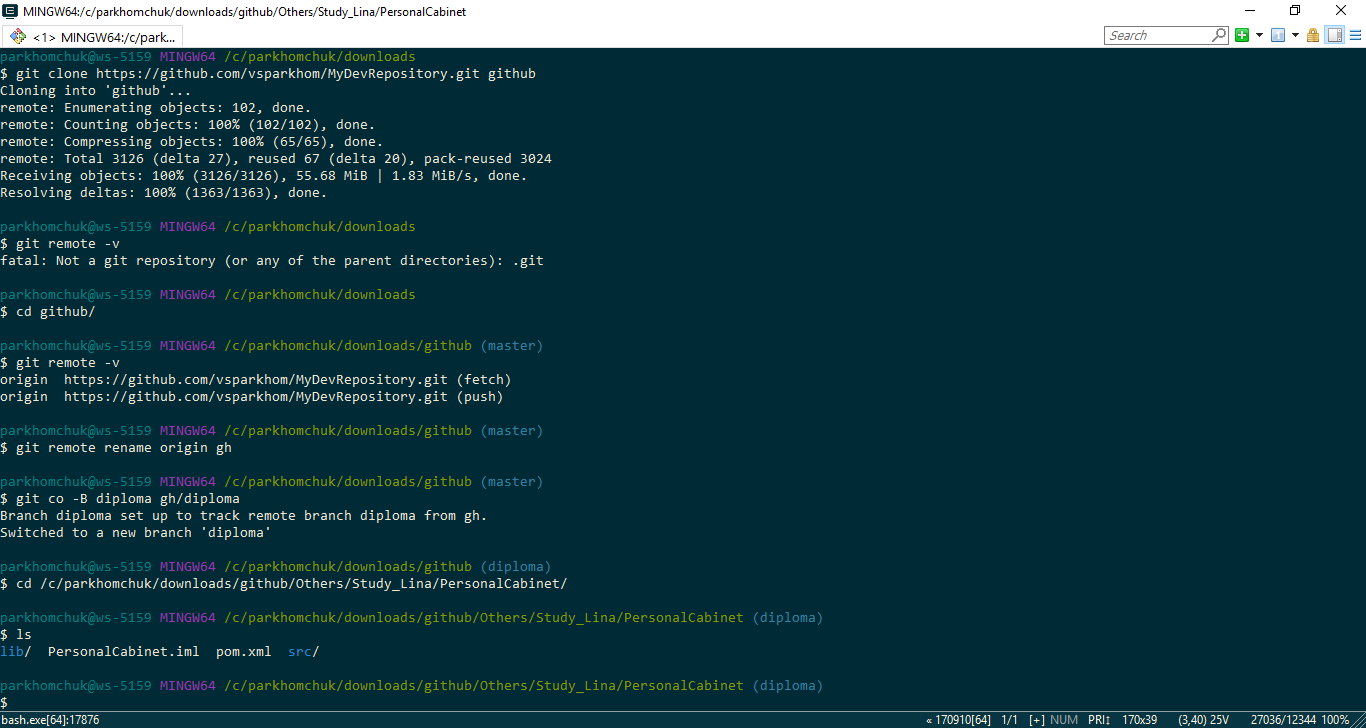


Рисунок 1-3. Создание и настройка рабочей копии репозитория

### 3.1.4 Настройка веб-сервера и развертывание приложения

После успешной сборки исходного кода необходимо развернуть приложение на тестовый веб-сервер для дальнейшего тестирования и отладки. В качестве веб-сервера будет выбран Apache Tomcat.

Apache Tomcat — контейнер сервлетов с открытым исходным кодом, разрабатываемый Apache Software Foundation. Реализует спецификацию сервлетов, спецификацию JavaServer Pages (JSP) и JavaServer Faces (JSF). Написан на языке Java. Он позволяет запускать веб-приложения и содержит ряд программ для самоконфигурирования. Tomcat используется в качестве самостоятельного веб-сервера, в качестве сервера контента в сочетании с веб-сервером Apache HTTP Server, а также в качестве контейнера сервлетов в серверах приложений JBoss и GlassFish.

Последнюю версию Apache Tomcat можно скачать на [официальном сайте](https://tomcat.apache.org/download-90.cgi) (рисунок).

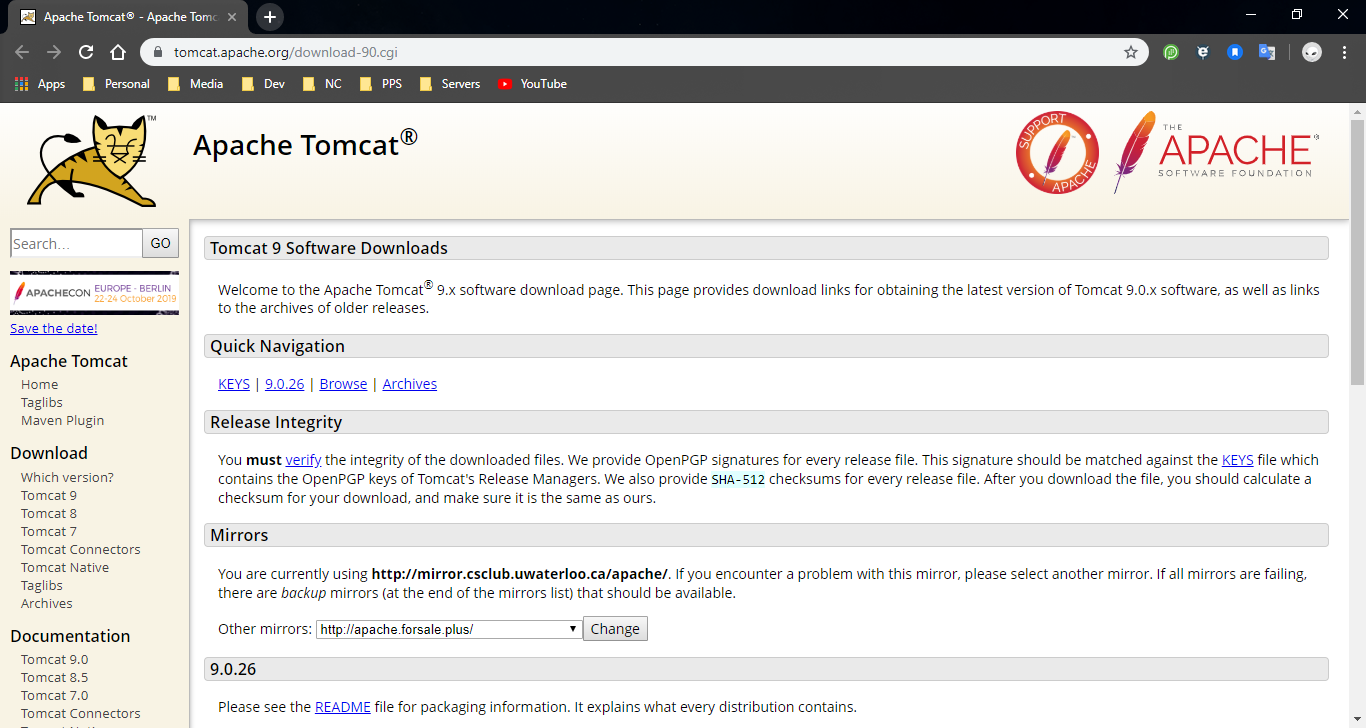


Рисунок № Официальный сайт Apache Tomcat

После настройки сервера (конфигурация портов, создание пользователей и соответствующих ролей) нужно запустить сервер командой *./startup.sh* (рисунок).



Рисунок № Запуск сервера Apache Tomcat

Затем нужно в браузере открыть адрес <http://localhost:8080/> и появится стандартное окно сервера (рисунок).

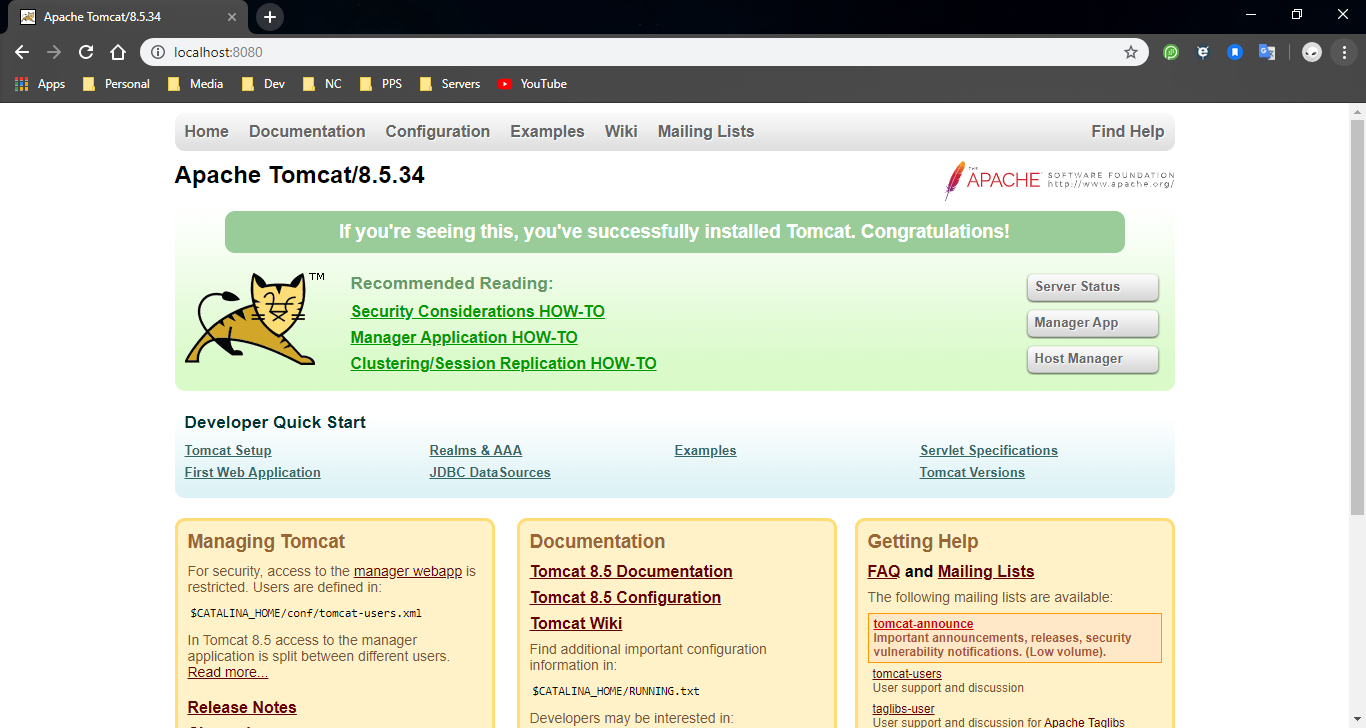


Рисунок № Стандартная страница Apache Tomcat

По кнопке «Server Status» можно посмотреть текущий статус сервера.

Нажав на ссылку «Manager App» можно перейти на страницу управления веб-приложениями (рисунок).

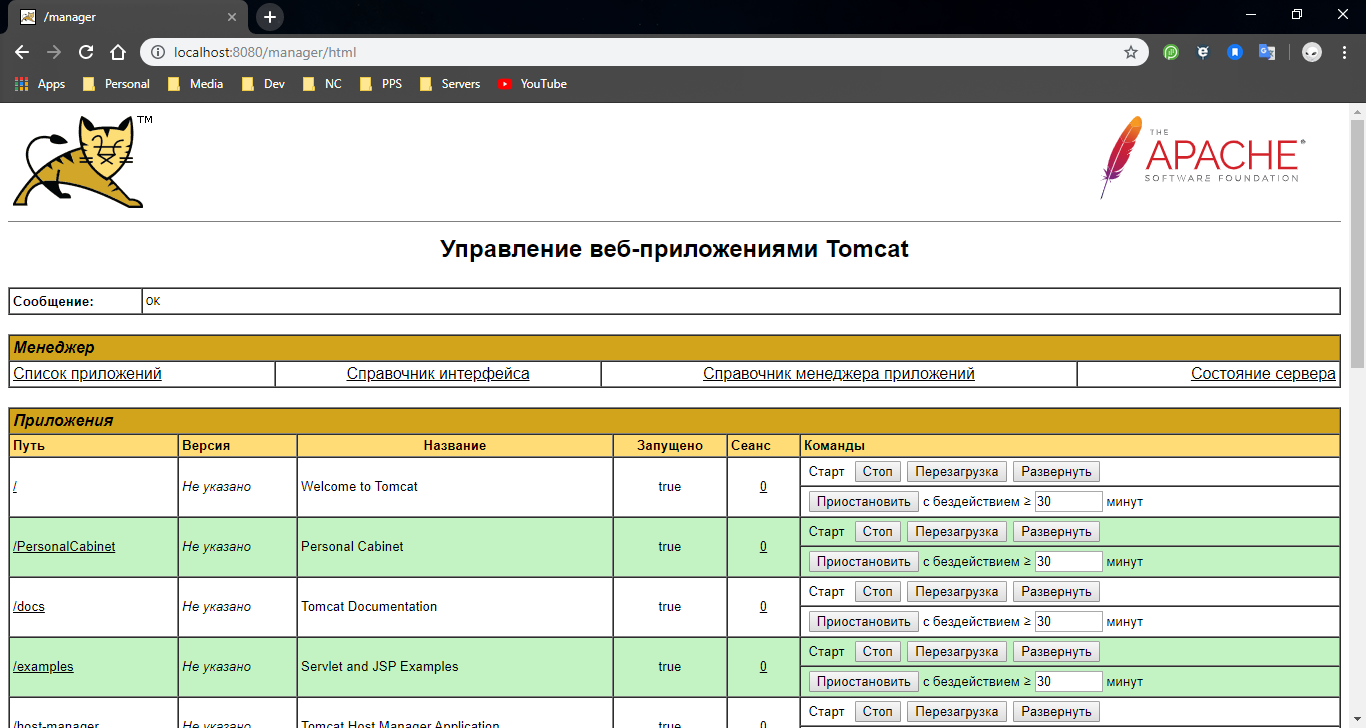


Рисунок № Управление веб-приложениями Apache Tomcat

На этой же странице доступны функции для старта, остановки и перезагрузки установленных приложений.

Также, здесь присутствует возможность развертывания приложения. Для этого необходимо указать основные параметы (веб-адрес в браузере и путь к WAR файлу) и нажать кнопку «Развернуть».

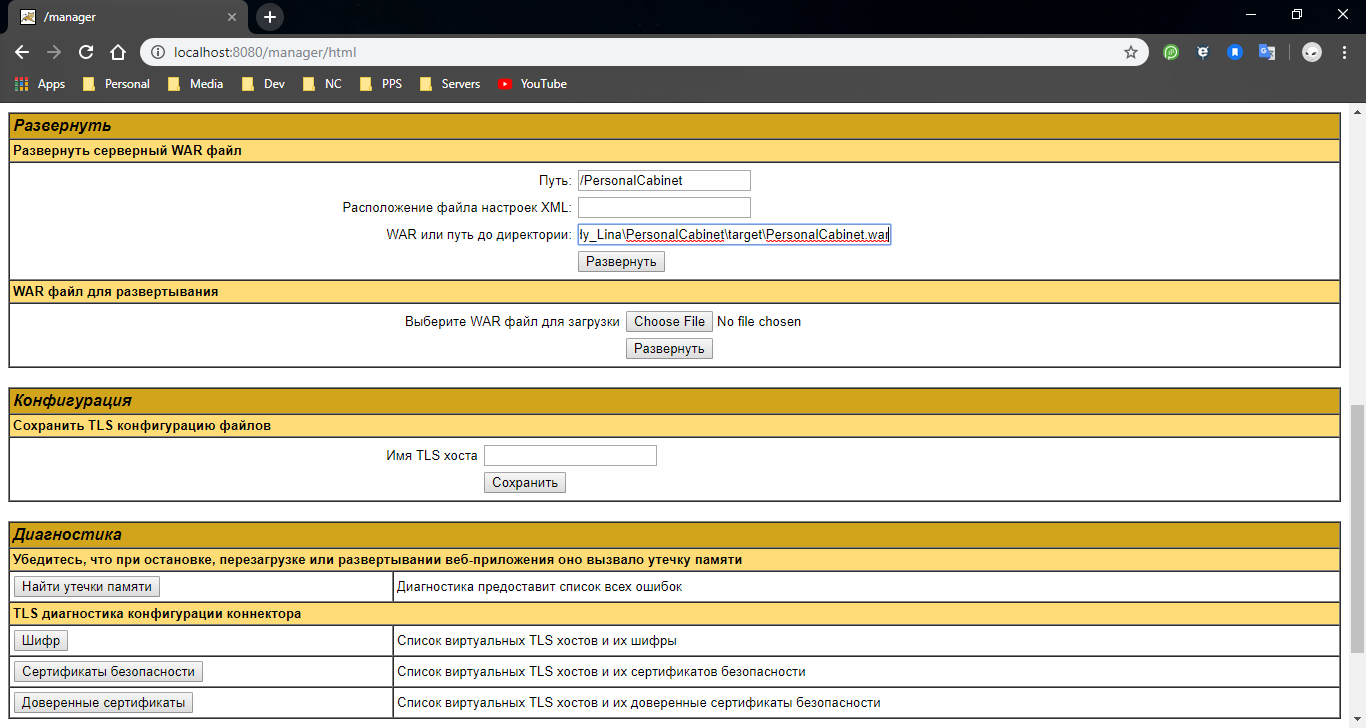


Рисунок № Страница для развертывания приложений

После завершения конфигурации приложения «Личный кабинет» и перезагрузки веб-сервера, приложение будет доступно по адресу <http://localhost:8080/PersonalCabinet/>

## 3.2 Тестирование приложения

### 3.2.1 Описание графического интерфейса

***Страница «Account information»***

На данной странице находятся основная информация с разных отделов (Billing, Services, Support). В секции Billing показан текущий баланс пользователя, дата до которой нужно оплатить счет и опция отправления электронных чеков. В секции Services размещены панели сервисов, где указаны статус сервиса, название тарифа и ссылки на доступные тарифные планы. Секция Support содержит контактную информацию предприятия (телефон, электронный адрес).

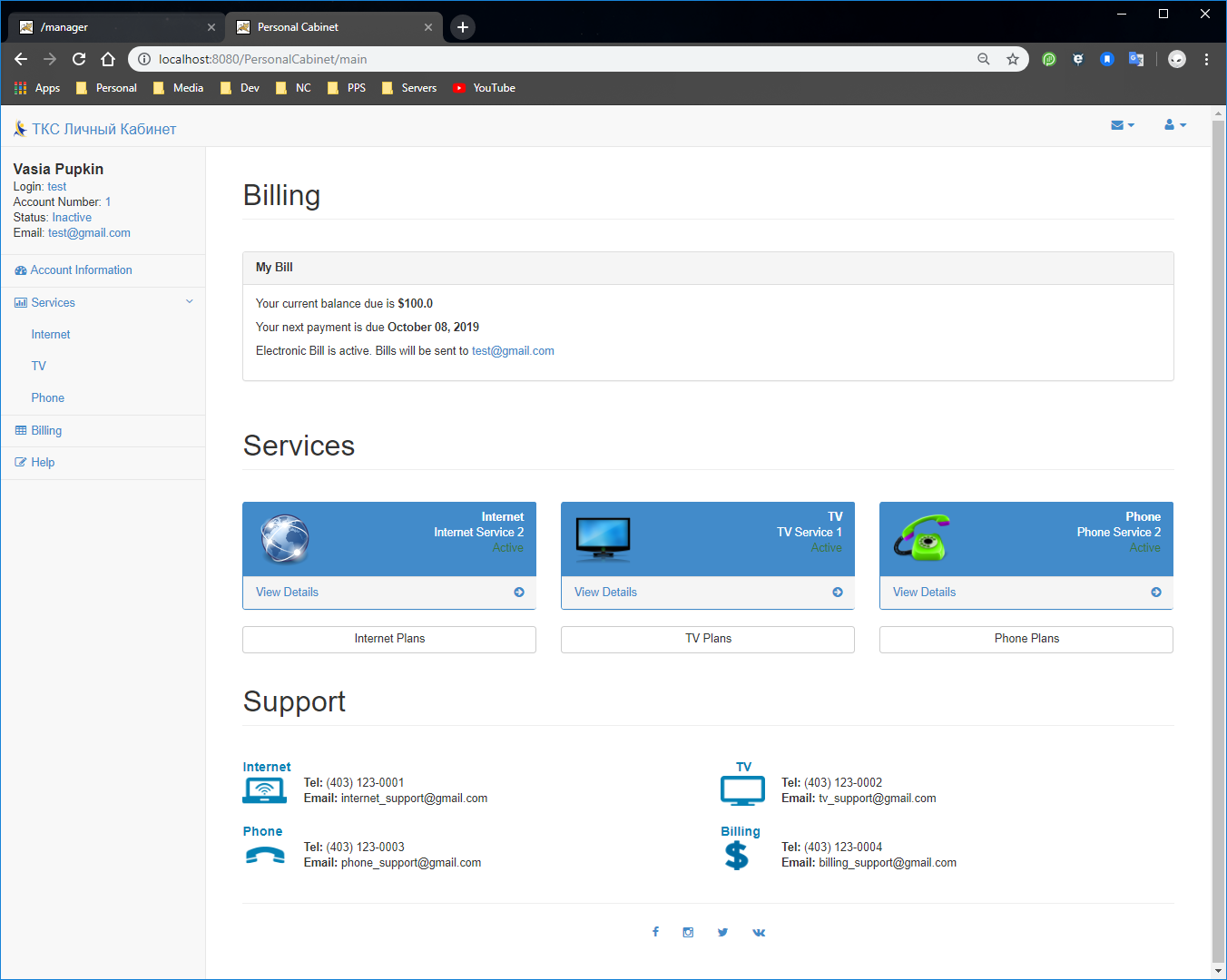


Рисунок № Страница «Account information»

***Страницы тарифных планов***

При переходе на страницу тарифных планов для каждого из сервисов отображаются доступные тарифные планы, их стоимость и технические характеристики. Если тарифный план уже установлен, то соответствующий управляющий элемент будет недоступен. В ином случае пользователь может применить выбраный план.

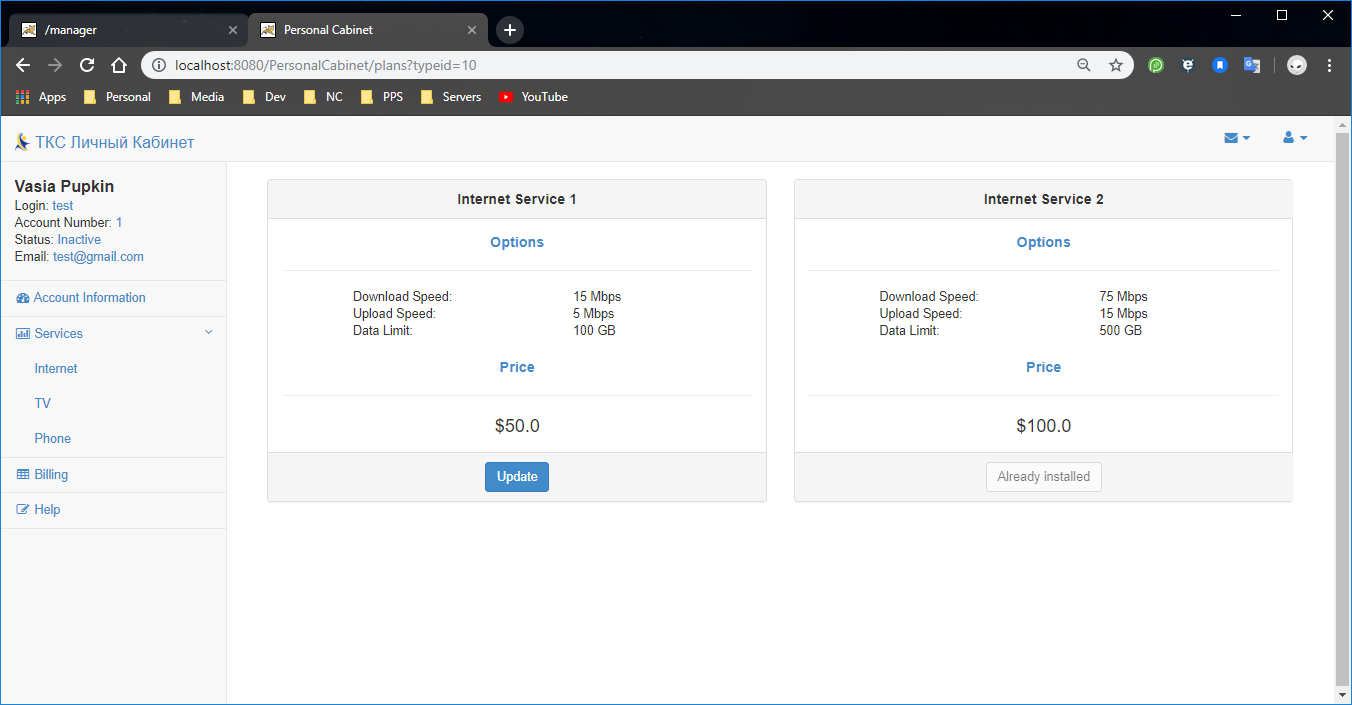


Рисунок № Тарифные планы Интернет сервиса

Для Интернет сервиса присутствуют следующие технические характеристики: скорость загрузки/выгрузки и ежемесячный лимит данных.

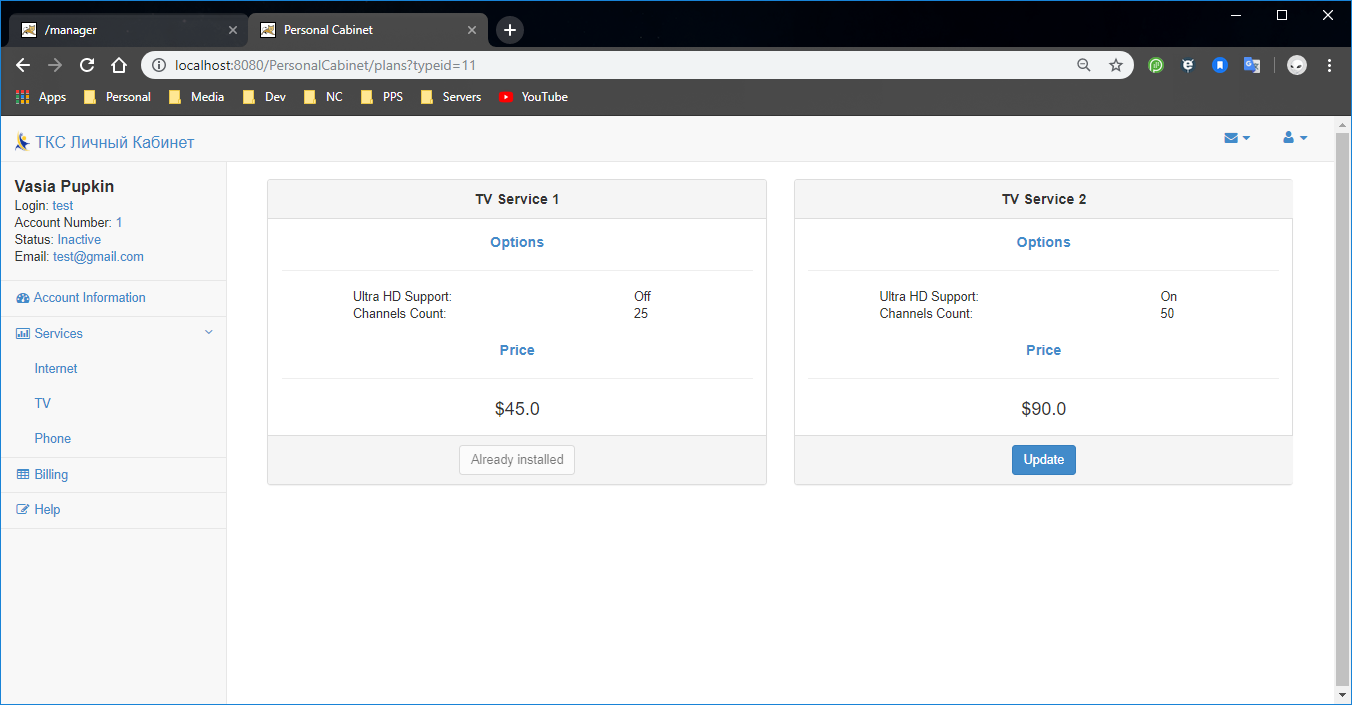


Рисунок № Тарифные планы ТВ сервиса

Для ТВ сервиса присутствуют следующие технические характеристики: количество каналов и поддержка сверхвысокой чёткости.

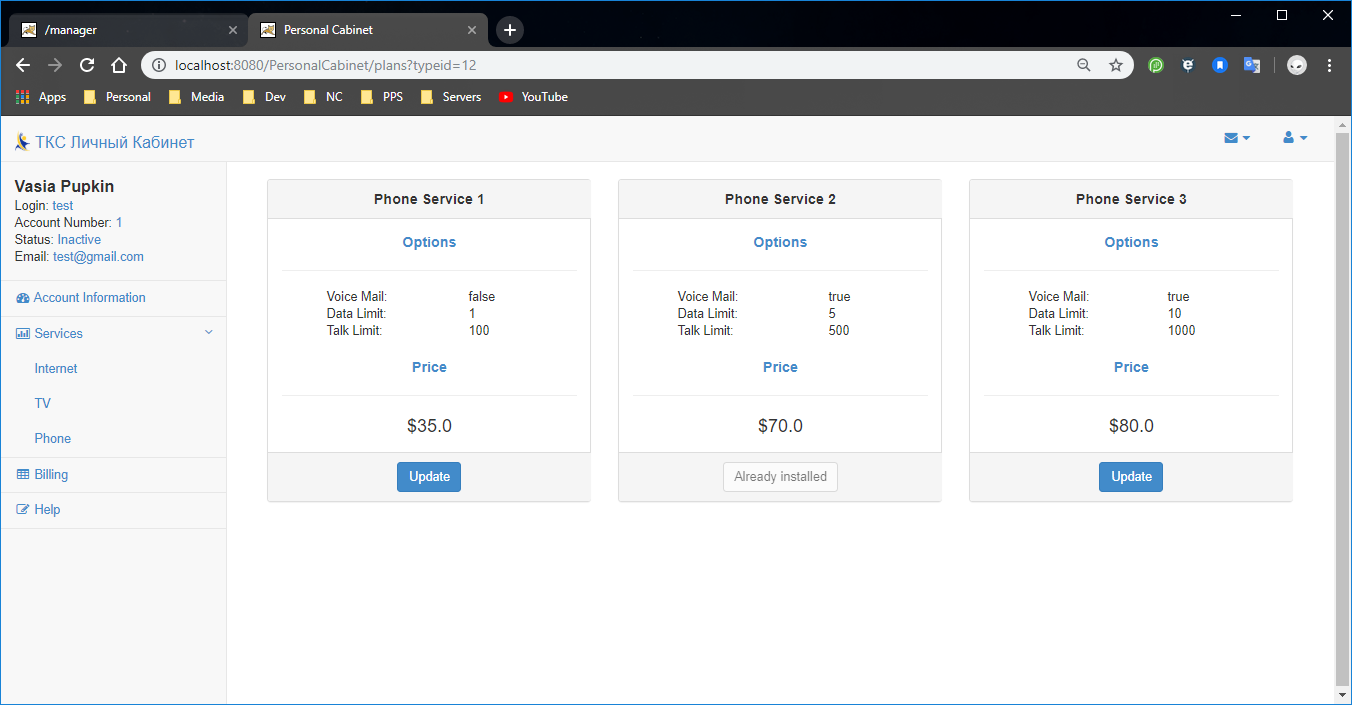


Рисунок № Тарифные планы сервиса Телефон

Для сервиса Телефон присутствуют следующие технические характеристики: наличие голосовой почты, ежемесячный объем мобильного интернета, количество бесплатных минут в месяц.

***Страница Интернет сервиса***

На странице Интернет сервиса отображаются технические характеристики тарифного плана, установленное оборудование (например модем), адреса электронной почты и количество использованого трафика.

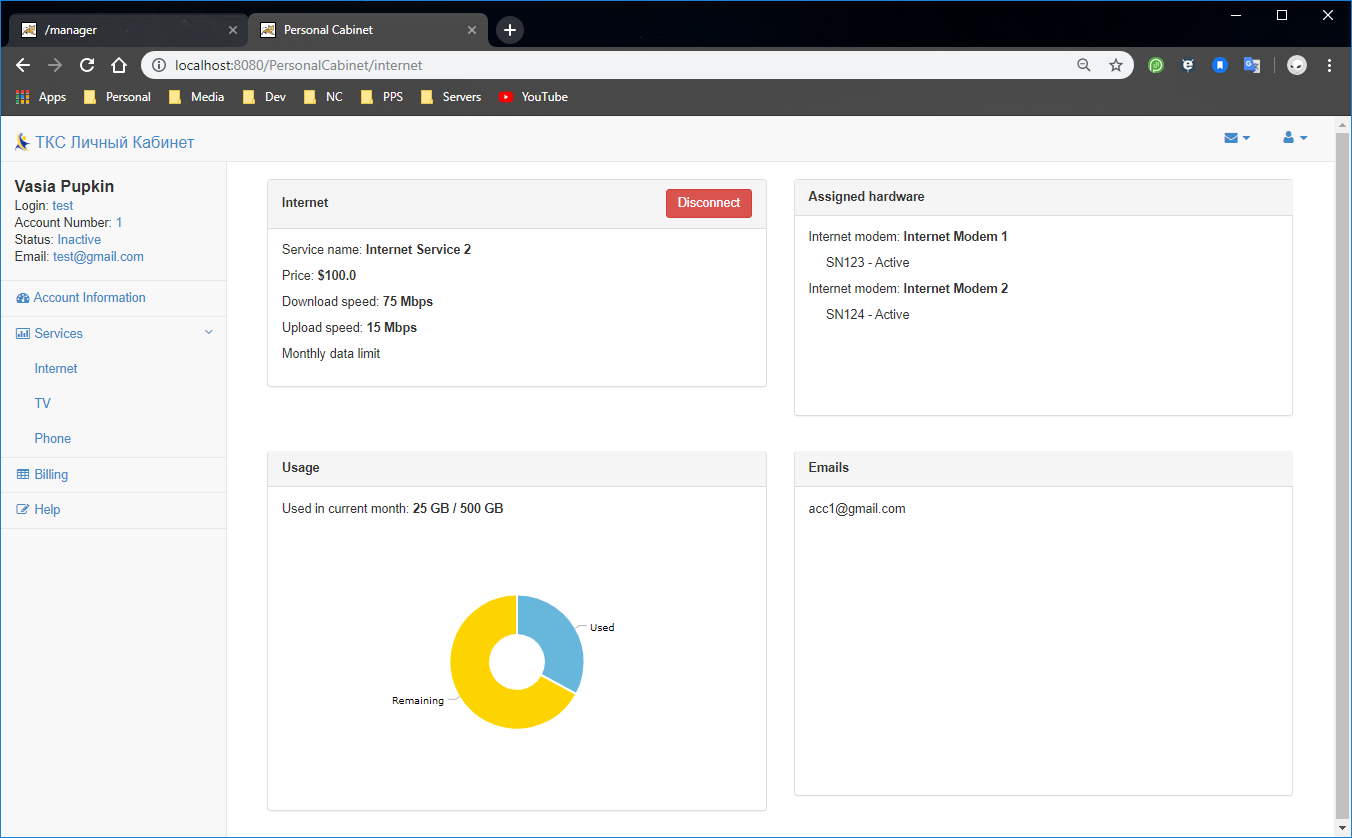


Рисунок № Страница Интернет сервиса

***Страница ТВ сервиса***

На странице ТВ сервиса отображаются технические характеристики тарифного плана и установленное оборудование

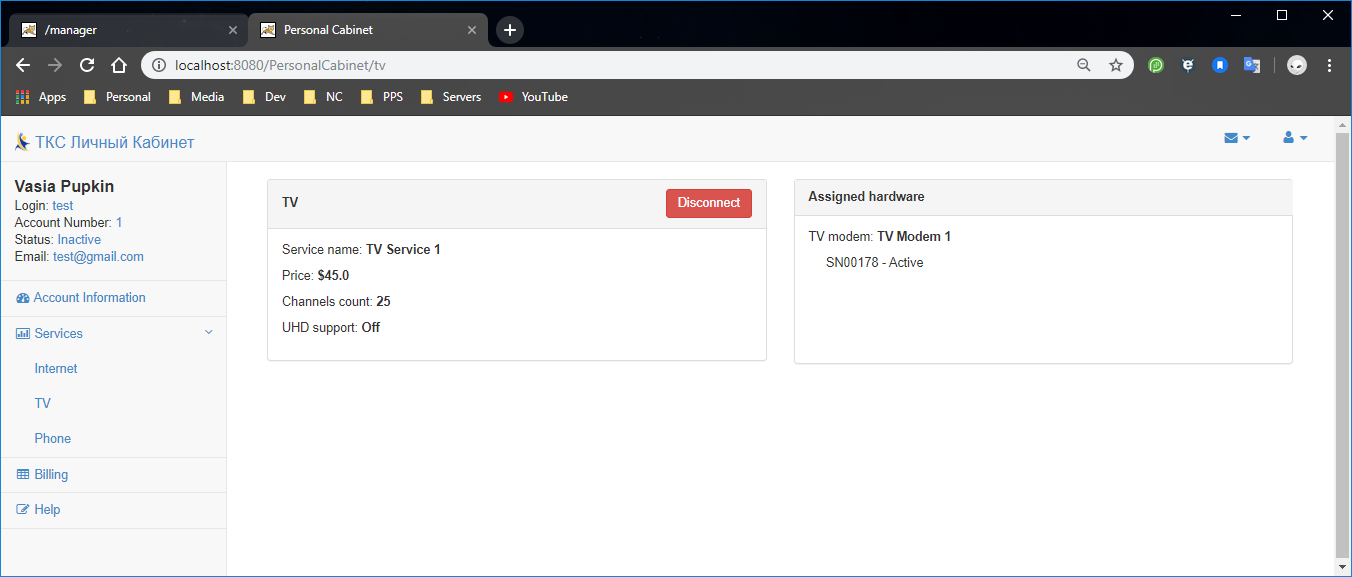


Рисунок № Страница ТВ сервиса

***Страница сервиса Телефон***

На странице Телефон сервиса отображаются технические характеристики тарифного плана, установленное оборудование.

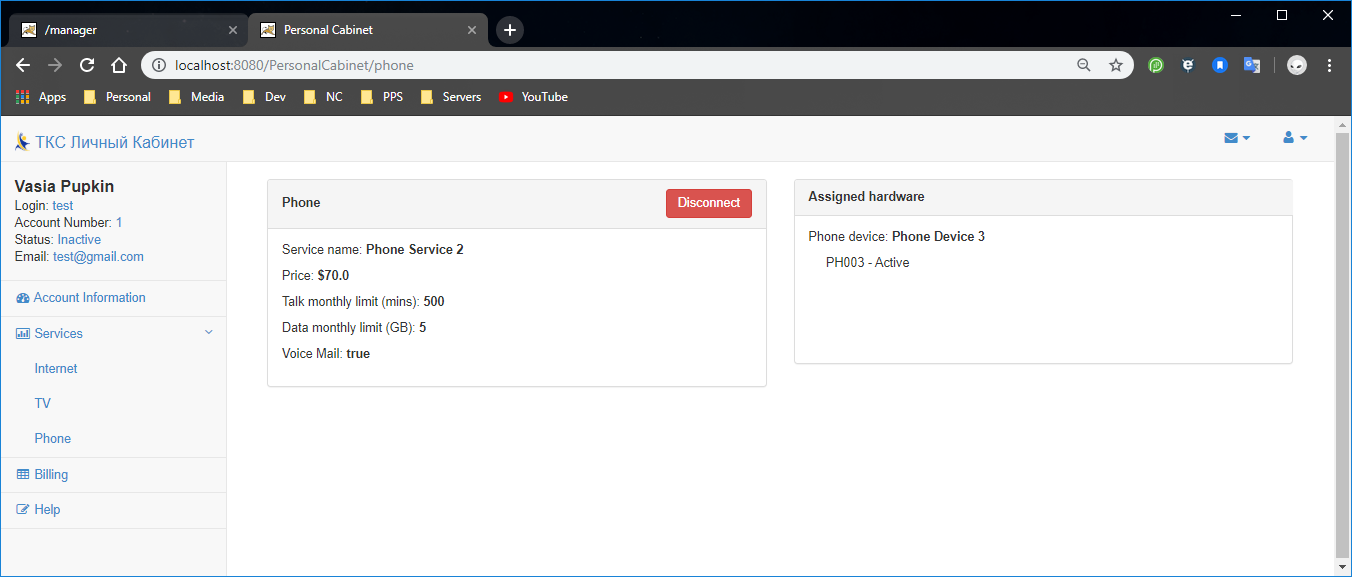
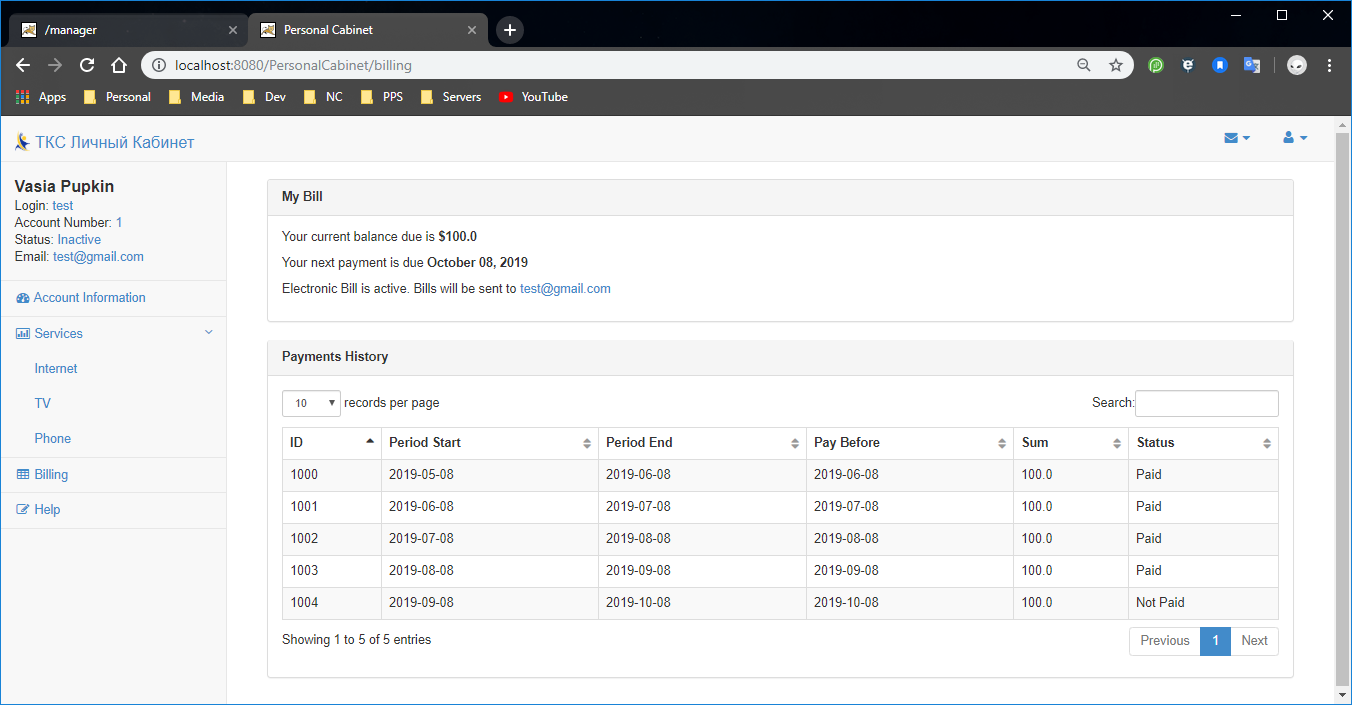


Рисунок № Страница сервиса Телефон

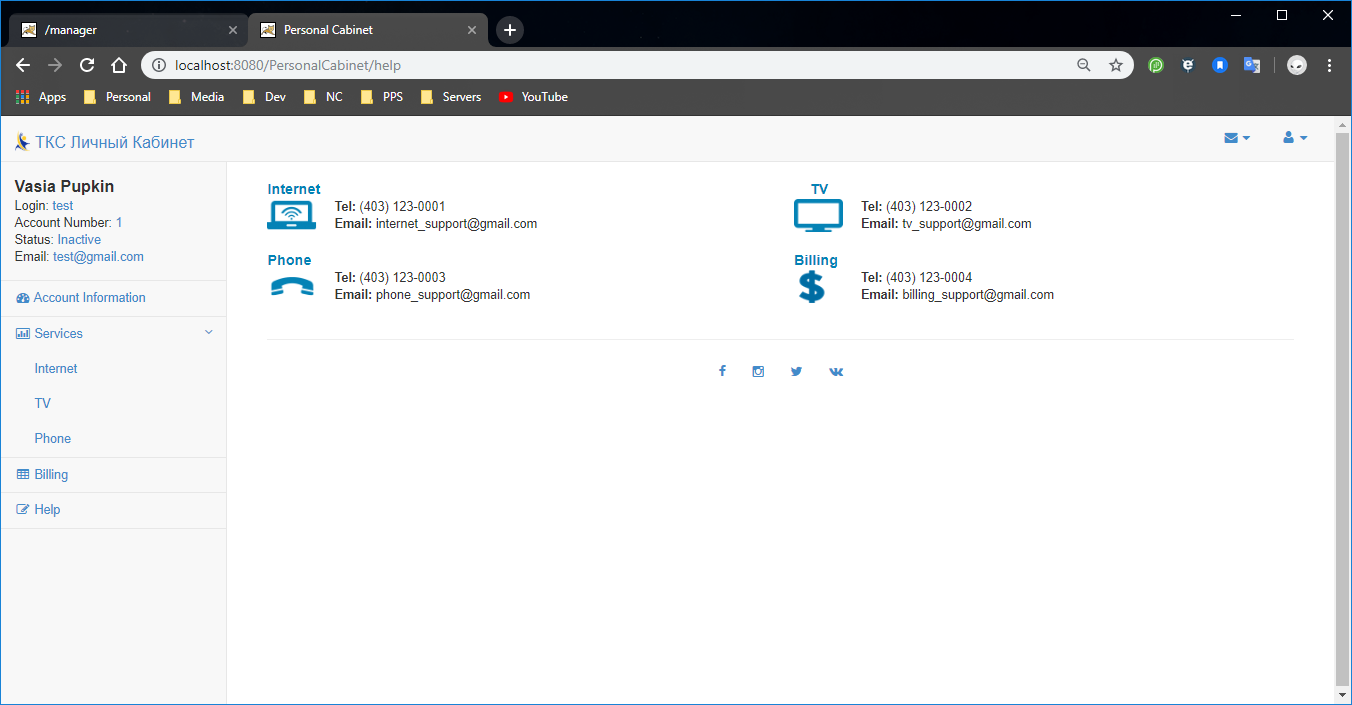
***Страница «*Billing*»***

На странице Billing сервиса отображается информация о текущем балансе и история платежей.



***Страница «Help»***

На странице Help сервиса указана контактная информация о предприятии. В перспективе здесь также можно разместить ссылку на онлайн чат для решения технических проблем.



### 3.2.2 Тестирование приложения

***Примеры подключения/модификации/отключения услуги***

***Просмотр платежей***

# ВЫВОДЫ

В ходе выполнении производственной практики была спроектирована информационная система (ИС) предприятия ООО «Сумские телекомсистемы». Данная ИС предназначена для расширения возможностей взаимодействия предприятия с клиентами, повышения качества обслуживания и снижения расходов на обслуживающий персонал.

Был проведен анализ основных параметров и функций, которые должны быть реализованы в данной ИС. При помощи DFD диаграмм мы построили и проиллюстрировали концептуальную модель системы, определили основные процессы, потоки и хранилища данных.

В дальнейшем на основе данной информационной системы будет реализовано веб-приложение «Личный кабинет».

# СПИСОК ЛИТЕРАТУРЫ

1. Слайды лекций дисциплины «Информационные системы и базы данных» [Электронный ресурс] – Режим доступа: dl.sumdu.edu.ua/e-pub/db, свободный.

2. Курс «Проектирование информационных систем» Национального Открытого Университета «ИНТУИТ» [Электронный ресурс] - Режим доступа: <http://www.intuit.ru/studies/courses/2195/55/info> , свободный— Загл. с экрана.

3. Сайт учебных материалов В. Анисимова [Электронный ресурс] : Проектирование информационных систем /автор В. Анисимов, доц. каф. "Информационные технологии и системы" Дальневосточного государственного университета путей сообщения – Режим доступа: https://sites.google.com/site/anisimovkhv/learning/pris/lecture, свободный.

4. Грекул В. И. Проектирование информационных систем / Грекул В. И., Денищенко Г. Н., Коровкина Н. Л. – ИНТУИТ, 2005 – С. 240.

5. Мацяшек Л. А., Анализ требований и проектирование систем. Разработка информационных систем с использованием UML / Мацяшек Л. А. – Вильямс, 2002 – С. 432.

6. Леоненков А. Самоучитель UML / Леоненков А. - БХВ-Петербург, 2004 – С.418.

7. Форум CITForum [Электронный ресурс] : Диаграммы потоков данных - Режим доступа: http://citforum.ru/programming/oop\_rsis/glava2\_6\_1.shtml, свободный— Загл. с экрана.

Г. Буч, Д. Рамбо, А. Джекобсон. Язык UML Руководство пользователя.

Леоненков А. Самоучитель UML.

Мартин Фаулер UML Основы. Третье издание

Новиков Ф.А. Анализ и проектирование на UML