ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ "САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М. А. БОНЧ-БРУЕВИЧА"

Факультет инфокоммуникационных сетей и систем

Кафедра программной инженерии и вычислительной техники

ОТЧЕТ ПО

КУРСОВОЙ РАБОТЕ

по дисциплине

«Разработка Java-приложений управления телекоммуникациями»

Вариант №9

Выполнил:

студент 3 курса

группы ИКПИ-15

Ильницкий Иван

Принял:  
Леонов Александр Сергеевич

\_\_\_\_\_\_\_\_\_\_\_\_\_

Санкт-Петербург

2023 год

**РЕФЕРАТ**

Отчет 62с., 8 рис., 2 табл., 4 источн., 3 прил.

JAVA, СЕРВЛЕТ, ВЕБ-ПРИЛОЖЕНИЕ, ВЕБ-СТРАНИЦА, JAKARTA SERVLET API, MAVEN, CRUD, JDBC, ТЕСТИРОВАНИЕ, HTTP

Объектом исследования являются веб-приложения, разрабатываемые с помощью языка программирования Java и программного интерфейса Jakarta Servlet Api.

Цель работы – разработка веб-приложения для учета личных затрат.

В ходе выполнение курсовой работы были получены следующие результаты:

1. Определен современный программный инструментарий для разработки веб-приложений на языке программирования Java
2. Исследованы различные методы предоставления информации на веб-страницах, не препятствующие их доступности
3. Изучены различные способы работы с базами данных с помощью пользовательских программных интерфейсов, предоставляемых Java и другими ее компонентами
4. Разработан полноценный веб сервис для учета личных затрат

**ТЕРМИНЫ, ОПРЕДЕЛЕНИЯ И СОКРАЩЕНИЯ**

В настоящем отчете о курсовой работе применяют следующие термины, определения и сокращения:

*Веб-приложение* – программное приложение, которое пользователи запускают через веб-браузер на своем устройстве и взаимодействуют с ним посредством интернета.

*Веб-страница* – документ или информационный ресурс Всемирной паутины, доступ к которому осуществляется с помощью веб-браузера.

*База данных* – совокупность данных, организованных по определенным правилам, предусматривающим общие принципы описания, хранения и манипулирования данными, независимая от прикладных программ.

*Веб-интерфейс* – веб-страница или совокупность веб-страниц, предоставляющая пользовательский интерфейс для взаимодействия с сервисом или устройством посредством веб-браузера.

*Нормальная форма базы данных* – требование, предъявляемое к структуре таблиц в теории реляционных баз данных для устранения из базы избыточных функциональных зависимостей между атрибутами.

*URL* (Uniform Resource Locator c англ. – «единообразный указатель местонахождения ресурса») - адрес ресурса в сети Интернет.

*CRUD* (create, read, update, delete с англ – «создание, чтение, модификация, обновление) – акроним, обозначающий четыре базовые функции, используемые при работе с базами данных.

*ER-диаграмма* (от англ. Entity-Relationship diagram, модель «сущность-связь» – модель данных, позволяющая описывать отношения и взаимосвязи между объектами в данной предметной области.

*SQL* (от англ. Structured Query Language - «язык структурированных запросов») – декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных

*POJO* (англ. Plain Old Java Object – «старый добрый java-объект») – простой Java-объект, не унаследованный от какого-то специфического объекта и не реализующий никаких служебных интерфейсов сверх тех, которые нужны для бизнес-модели.

*API* (от англ. application programming interface – «интерфейс программирования приложения») - программный интерфейс, то есть описание способов взаимодействия одной компьютерной программы с другими.

*JDBC* (англ. Java DataBase Connectivity – «соединение с базами данных на Java») – платформенно независимый промышленный стандарт взаимодействия Java-приложений с различными СУБД.

**Содержание**

[**1.** **Постановка задачи** 4](#_Toc154098415)

[**1.1.** **Требования к функционалу программы** 5](#_Toc154098416)

[**1.2.** **Требования к веб-интерфейсу программы** 5](#_Toc154098417)

[**2.** **Среда разработки и используемые технологии** 6](#_Toc154098418)

[**3.** **Структура программы** 7](#_Toc154098419)

[**4.** **Отношения между таблицами в БД** 8](#_Toc154098420)

[**5.** **Разработка собственных классов** 10](#_Toc154098421)

[**5.1.** **Классы-сущности** 10](#_Toc154098422)

[**5.2.** **Классы-сервлеты** 10](#_Toc154098423)

[**5.3.** **Классы-сервисы** 11](#_Toc154098424)

[**5.4.** **Классы-репозитории** 12](#_Toc154098425)

[**6.** **Веб-страницы** 14](#_Toc154098426)

[**7.** **Скриншоты веб-страниц** 15](#_Toc154098427)

[**ЗАКЛЮЧЕНИЕ** 18](#_Toc154098428)

[**СПИСОК ЛИТЕРАТУРЫ** 19](#_Toc154098429)

[**ПРИЛОЖЕНИЕ** 20](#_Toc154098430)

[**ПРИЛОЖЕНИЕ A. Файлы .java** 21](#_Toc154098431)

[**ПРИЛОЖЕНИЕ Б. Файлы .jsp, .css и .js** 40](#_Toc154098432)

[**ПРИЛОЖЕНИЕ В. Файлы конфигурации** 60](#_Toc154098433)

# **Постановка задачи**

Разработать веб-приложение с использованием Jakarta EE. Приложение должно располагать возможностью создания, изменения, обновления и удаления данных. В процессе достижения цели необходимо решить ряд следующих задач:

* ознакомиться с современными технологиями и по возможности использовать их в своей разработке;
* определить программный инструментарий, применяемый для разработки веб-приложения;
* выявить и учесть методы и способы представления на веб-страницах различных видов информации, не препятствующие их доступности;
* ознакомиться с основными правилами и рекомендациями по разработке и созданию веб-приложений и принципиально следовать им в своей практике;
* определиться со структурой веб-страниц;

*Вариант 9.* Разработать веб-приложение для учета личных затрат. Система должна включать функционал по учету доходов и расходов, а также позволять генерировать диаграммы доходов и расходов по категориям.

## **Требования к функционалу программы**

Разрабатываемая программа должна иметь широкий и удобный функционал по работе с товарами, категориями товаров, поставками и продажами. В качестве ключевых функций можно выделить следующие:

* просмотр информации о категориях;
* просмотр информации о расходах
* создание записи о расходах
* изменение записи о расходах
* удаление записи о расходах
* просмотр информации о доходах
* создание записи о доходах
* изменение записи о доходах
* удаление записи о доходах
* просмотр информации о расходах пользователя по категориям
* просмотр информации о доходах пользователя по категориям
* авторизация в приложении
* регистрация в приложении
* генерация диаграмм о доходах пользователя по категориям
* генерация диаграмм о расходах пользователя по категориям

## **Требования к веб-интерфейсу программы**

Все веб-страницы должны быть оформлены в одном стиле: использование не более двух различных шрифтов, использование не более трех цветов, допустимо использовать различные оттенки этих цветов. Интерфейс веб-страницы следовать стандартам человеко-машинного взаимодействия.

Помимо вышеперечисленных требований сайт также должен быть адаптивным: расположение элементов должно оставаться удобным и понятным при просмотре с мобильных устройств с различными дисплеями.

# **Среда разработки и используемые технологии**

Программа разработана на языке Java (OpenJDK 17) в среде IntelliJ IDEA Ultimate 2023.2.4. в операционной системе MacOS. Также использовались следующие технологии и библиотеки:

* Jakarta Servlet API - фреймворк, предоставляющий средства для создания веб-сервлетов и JSP-страниц;
* JDBC – программный интерфейс java для взаимодействия с базами данных;
* PostgreSQL Driver - драйвер JDBC, предоставляющий средства подключения к базе данных PostgreSQL;
* Apache Tomcat - контейнер сервлетов, предназначенный для развертывания и выполнения веб-приложения;
* Maven – инструмент для сборки проектов, подключения зависимостей и плагинов;

# **Структура программы**

Структура программы разбита на несколько слоёв. Это упрощает процесс разработки и поддержки приложения. Для веб-приложений обычно выделяют следующие слои:

* слой представления;
* слой контроллера;
* сервисный слой;
* слой доступа к базе данных.

В качестве представления в разрабатываемом приложении выступают jsp-страницы. Они динамически генерируются на сервере и отображают пользователю все необходимые данные.

Веб-сервлеты представляют собой слой контроллера. Они обрабатывают пользовательские запросы и генерируют ответы.

Сервисный слой содержит всю логику работы приложения. В нем производятся различные вычисления, данные запросов проходят валидацию и обработку, отлавливаются и обрабатываются ошибки.

Слой доступа к данным представляет собой репозитории. В них осуществляется подключение к базе данных, выполняются CRUD-операции с сущностями.



Рисунок 1 – Структура приложения

# **Отношения между таблицами в БД**

Используемая база данных должна быть нормализована.

В базах данных, нормализация — это процесс организации данных с целью устранения избыточности и обеспечения целостности информации. Нормальные формы определяют правила для структурирования таблиц в базах данных, чтобы минимизировать дублирование данных и обеспечить эффективность и надежность операций. В подавляющем большинстве современных приложений базы данных находятся в третьей нормальной форме.

Основным принципом третьей нормальной формы является то, что неключевые столбцы должны зависеть только от первичного ключа, а не от других неключевых столбцов. То есть, если у нас есть неключевой столбец, который функционально зависит от другого неключевого столбца, то эти зависимости должны быть устранены. Преимуществом третьей нормальной формы является более логичное и четкое представление данных, однако, увеличивается сложность запросов.

Разработанная база данных удовлетворяет требованиям третьей нормальной формы. ER-диаграмма базы данных, используемой в приложении, представлена на рис. 2.

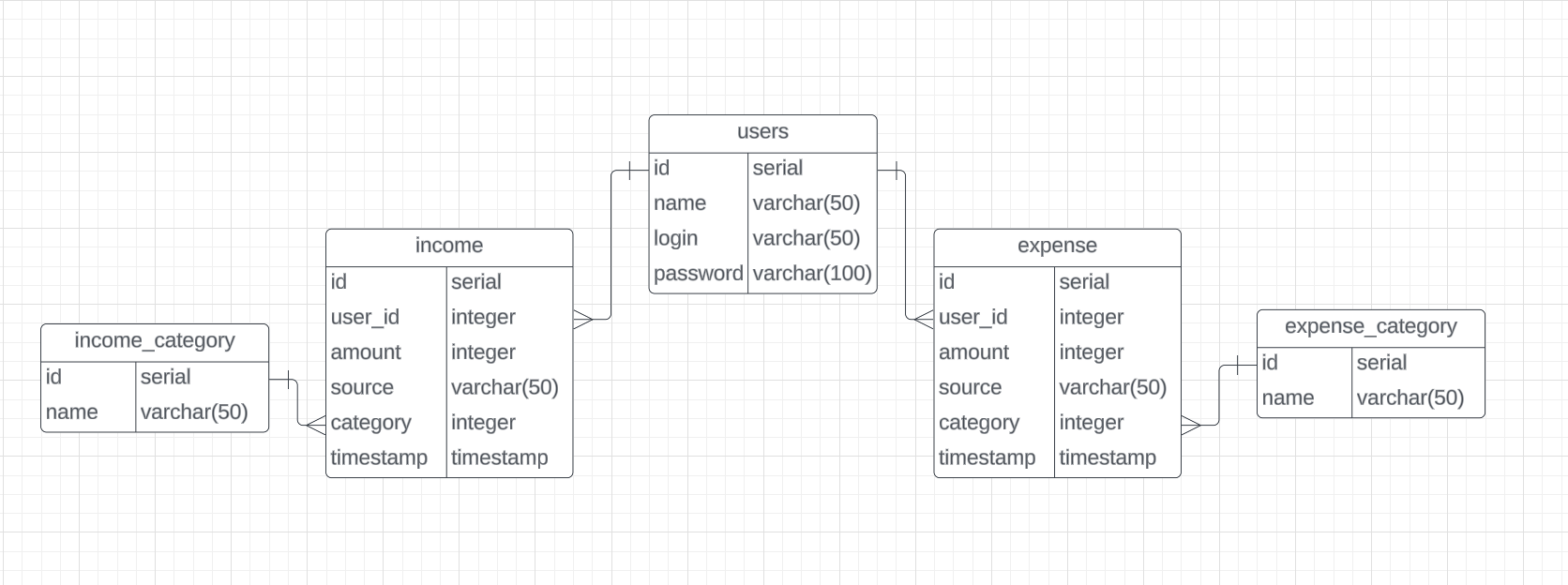


Рисунок 2 – ER-диаграмма

# **Разработка собственных классов**

В проекте разработанные классы разбиты по соответствующим пакетам:

* servlet – сервлеты;
* model – сущности, описывающие затраты, категории и так далее;
* service – сервисы;

## **Классы-сущности**

Таблица 1 – Классы, описывающие сущности

|  |  |
| --- | --- |
| **Класс** | **Поля класса** |
| Expense.java | int id – идентификатор затраты  int userId – идентификатор пользователя  int amount – сумма затраты  String source – описание затраты  int categoryId – идентификатор категории затраты  String category – название категории затраты  Timestamp timestamp – дата покупки  String dateFormatted – дата покупки в читабельном формате |
| Category.java | int id – идентификатор категории  String name – название категории |
| Income.java | int id – идентификатор дохода  int userId – идентификатор пользователя  int amount – сумма дохода  String source – название источника дохода  int categoryId – идентификатор категории дохода  String category – название категории дохода  Timestamp timestamp – дата получения дохода  String dateFormatted – дата получения дохода в читабельном формате |
| User.java | int id – идентификатор пользователя  String name – имя пользователя  String login – логин пользователя  String password – пароль пользователя |

## **Классы-сервлеты**

В приложении разработано 8 сервлетов AnalyticsServlet, ExpenseCategoryServlet, ExpenseServlet, IncomeCategoryServlet, IncomeServlet, LoginServlet, RegisterServlet и UserServlet обрабатывающих HTTP-запросы, связанные с операциями над товарами, категориями, продажами, поставками и отчетами соответственно. Каждый из сервлетов наследуется от HttpServlet и переопределяет два метода: doGet() и doPost(). Эти методы, в свою очередь, обрабатывают конкретные типы запросов, переадресуют пользователя на jsp-страницы или вызывают методы сервисных классов в зависимости от переданных в запросе параметров. s

Все сервлеты помечены аннотацией @WebServlet. Аннотация @WebServlet указывает контейнеру сервлетов, что этот класс является сервлетом и должен обрабатывать запросы, соответствующие указанным URL-шаблонам. Фрагмент исходного кода одного из сервлетов:

|  |
| --- |
| **LoginServlet.java** |
| @WebServlet("/login") public class LoginServlet extends HttpServlet {  private final LoginService loginService;   public LoginServlet() {  this.loginService = new LoginService();  }   @Override  protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  req.setAttribute("errorMessage", "");  req.getRequestDispatcher("/login.jsp").forward(req, resp);  resp.setContentType("text/html");  super.doGet(req, resp);  }  @Override  protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  String username = req.getParameter("username");  String password = req.getParameter("password");  int userId = loginService.auth(username, password);  if (userId != -1) {  req.getSession().setAttribute("user\_id", userId);  resp.sendRedirect( "/dashboard");  } else {  req.setAttribute("errorMessage", "error login or pass");  req.getRequestDispatcher("/login.jsp").forward(req, resp);  super.doPost(req, resp);  }  } } |

## **Классы-сервисы**

Для операций над сущностями предусмотрены сервисные классы DataBaseService, LoginService. Сервисные классы имеют следующие функции:

* выполнение подключения к базе данных
* авторизация и аутенфикация пользователя
* выполнение select-запросов к базе данных

Фрагмент исходного кода одного из сервисных классов:

|  |
| --- |
| **LoginService.java** |
| public class LoginService {  private final UserRepository userRepository;   public LoginService() {  this.userRepository = new UserRepository();  }   public int auth(String login, String password) {  String hashedPassword = hashPassword(password);  return userRepository.getUserIdByUsernameAndPassword(login,hashedPassword);  }  public int register(String name,String login,String password) {  if (userRepository.isUserExistsByLogin(login)) {  return -1;  }  String hashedPassword = hashPassword(password);  return userRepository.createUser(name,login,hashedPassword);  }  public String hashPassword(String password) {  MessageDigest digest = null;  try {  digest = MessageDigest.*getInstance*("SHA-256");  } catch (NoSuchAlgorithmException e) {  System.*out*.println(e.getMessage());  }  byte[] hashBytes = digest.digest(password.getBytes(StandardCharsets.*UTF\_8*));   BigInteger hashInt = new BigInteger(1, hashBytes);  return hashInt.toString(16);  } } |

## **Классы-репозитории**

Репозитории отвечают за взаимодействие с базой данных. Для каждой сущности в приложении предусмотрен свой репозиторий: ExpenseCategoryRepository, ExpenseRepository, IncomeCategoryRepository, IncomeRepository, UserRepository. Логика по подключению к источнику данных и получения соединения с базой данных вынесена в отдельный класс DatabaseService для избежания дублирования кода. Репозитории в разрабатываемом приложении выполняют следующие функции:

* Установление соединения с базой данных. Репозитории управляют установлением и разрывом соединения с базой данных. Для этого используются такие параметры БД, как URL, имя пользователя, пароль и так далее;
* Выполнение запросов. Репозитории отвечают за создание и выполнение SQL-запросов к базе данных;
* Обработка результатов запросов. После выполнения запроса репозиторий получает результирующие наборы и преобразует их в формат, понятный для приложения, например POJO.

Фрагмент исходного кода одного из репозиториев:

|  |
| --- |
| **ExpenseRepository.java** |
| public class ExpenseRepository {  private final DataBaseService dataBaseService;  static final String *SELECT\_BY\_USERID* = "SELECT \* from expense JOIN expense\_category" +  " on expense\_category.id=expense.category where expense.user\_id=? ORDER BY timestamp desc";  static final String *INSERT* = "INSERT INTO expense (user\_id,amount,source,category,timestamp) values (?,?,?,?,NOW())";  static final String *DELETE* = "DELETE FROM expense where id=?";  static final String *UPDATE* = "UPDATE expense SET amount=?,source=?,category=? where id=?";  static final String *SELECT\_SUM\_EXPENSES\_OF\_USER\_GROUP\_BY\_CATEGORY* =  "SELECT expense\_category.id, expense\_category.name, COALESCE(SUM(amount), 0) AS sum" +  " FROM expense\_category" +  " LEFT JOIN expense ON expense.category = expense\_category.id AND expense.user\_id=?" +  " GROUP BY expense\_category.id, expense\_category.name" +  " ORDER BY expense\_category.id";   public ExpenseRepository() {  this.dataBaseService = new DataBaseService();  }  public List<Expense> getExpensesByUserId(Integer userId) {  List<Expense> expenses = new ArrayList<>();  Connection conn = dataBaseService.getConnect();  try {  PreparedStatement statement = conn.prepareStatement(*SELECT\_BY\_USERID*);  statement.setInt(1, userId);  ResultSet resultSet = statement.executeQuery();  while (resultSet.next()) {  Expense expense = new Expense(  resultSet.getInt("id"),  resultSet.getInt("user\_id"),  resultSet.getInt("amount"),  resultSet.getString("source"),  resultSet.getInt("category"),  resultSet.getTimestamp("timestamp")  );  expense.setCategory(resultSet.getString("name"));  expenses.add(expense);  }  } catch (SQLException e) {  e.printStackTrace();  }  return expenses;  }  <…>  } |

# **Веб-страницы**

В приложении используются динамически генерируемые JSP страницы. Все страницы имеют схожее оформление: используется одинаковый текст, схожие цвета, одно и то же навигационное меню. Все используемые стили доступны в приложении Б.

Таблица 2 – Используемые веб-страницы

|  |  |
| --- | --- |
| **Веб-страница** | **Функционал** |
| main.html | Каркас веб-приложения. Включает в себя «шапку» сайта и навигационное меню |
| index.jsp | Веб-страница с приветствием и одной единственной кнопкой, по нажатию на которую пользователь попадает на login.jsp |
| userCabinet.jsp | Основная страница приложения. На ней отображается таблица со всеми расходами и карточки с затратами по категориям. Карточки кликабельны – при нажатии открывается модальное окно для создания записи о расходе в выбранной категории. |
| login.jsp | Страница авторизации в приложении |
| register.jsp | Страница регистрации в приложении |
| logout.jsp | Страница выхода из аккаунта |
| expense-history.jsp | Получение и изменение информации о расходах |
| income-history.jsp | Получение, создание и изменение информации о доходах |
| analytics.jsp | Отображение диаграмм доходов и расходов по категориям |

# **Скриншоты веб-страниц**

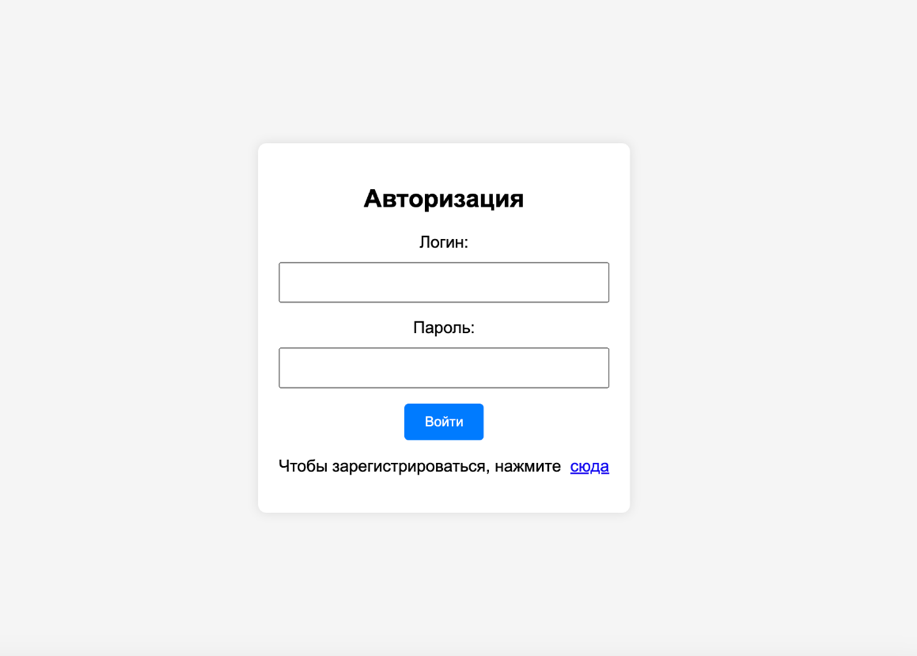


Рисунок 3 – Страница авторизации

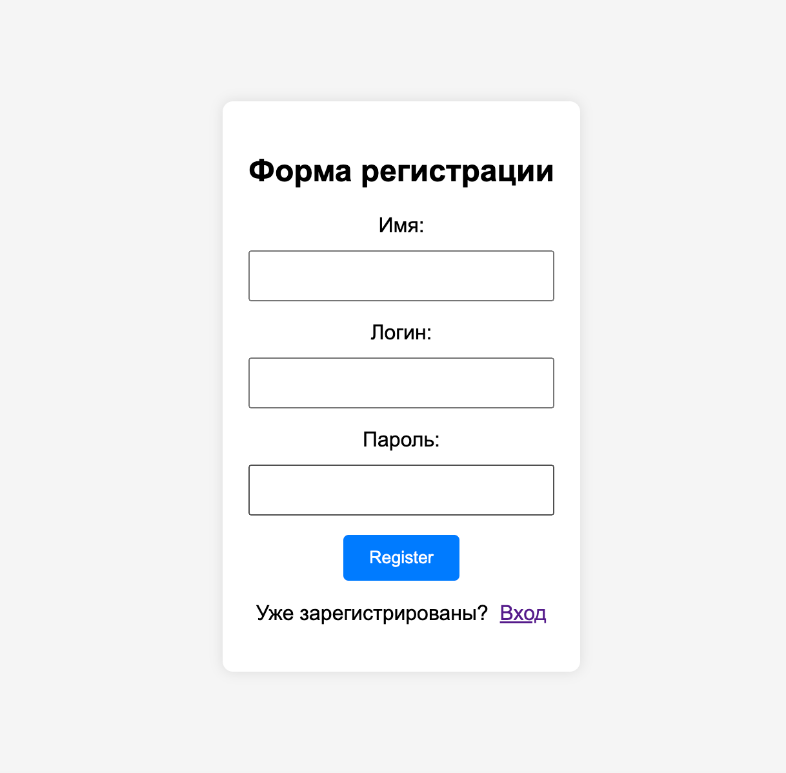


Рисунок 4 – Страница регистрации

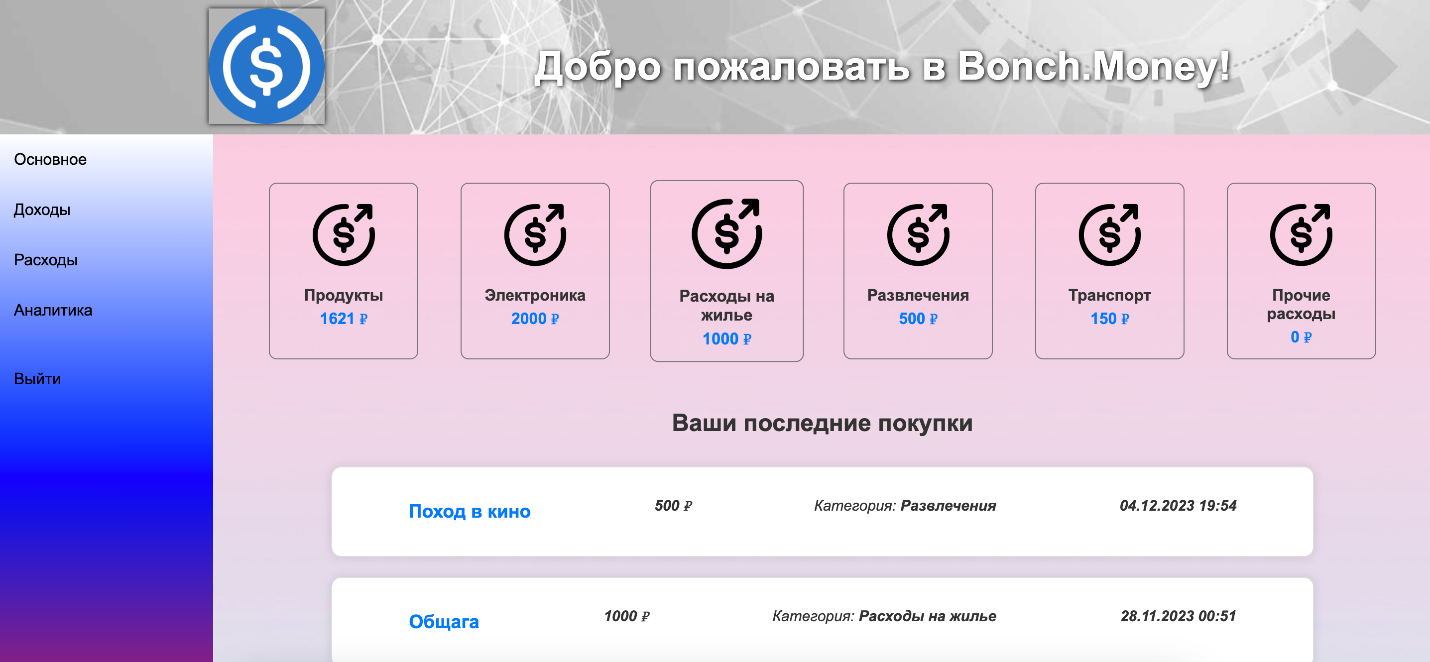


Рисунок 5 – Основная веб-страница

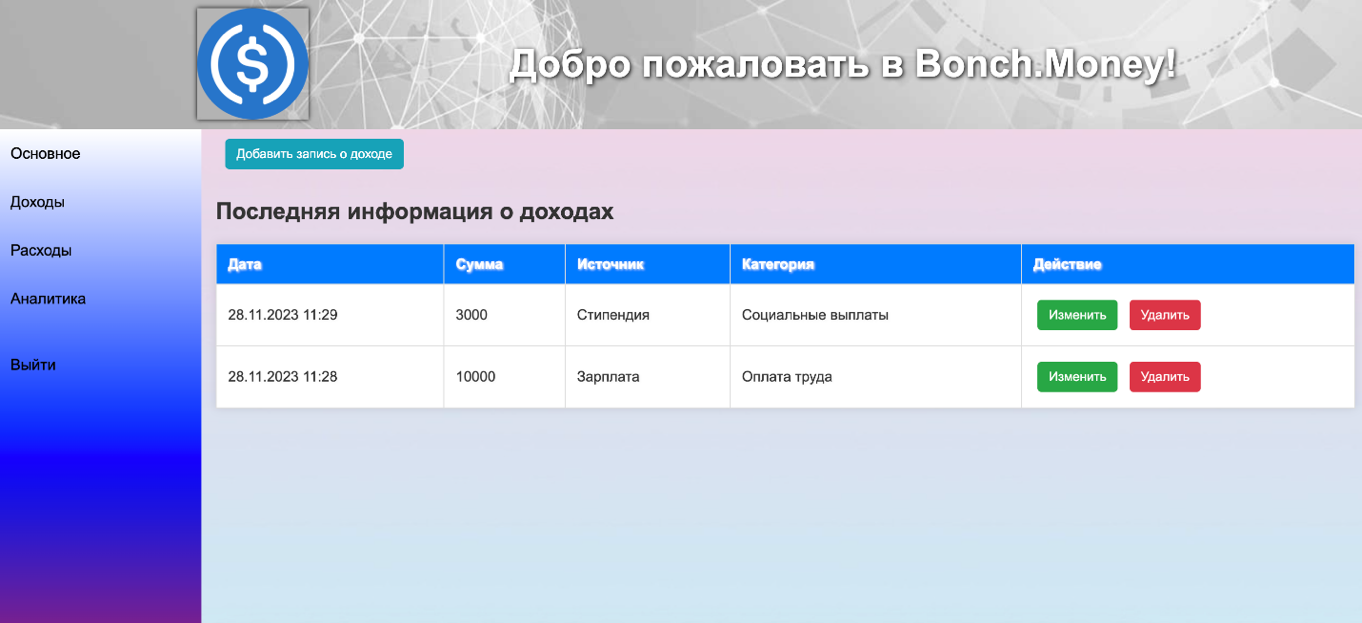


Рисунок 6 – Веб-страница для добавления записи о доходах

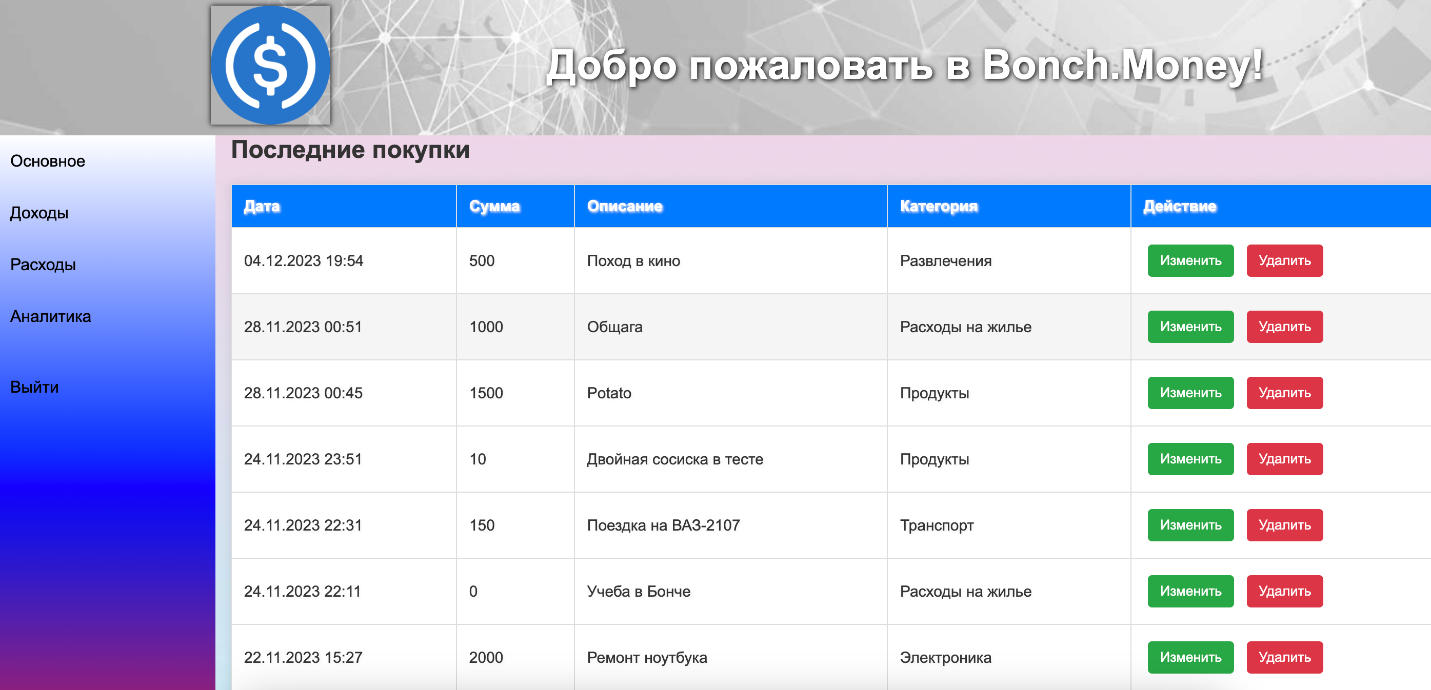


Рисунок 7 – Веб-страница для изменения записей о расходах.

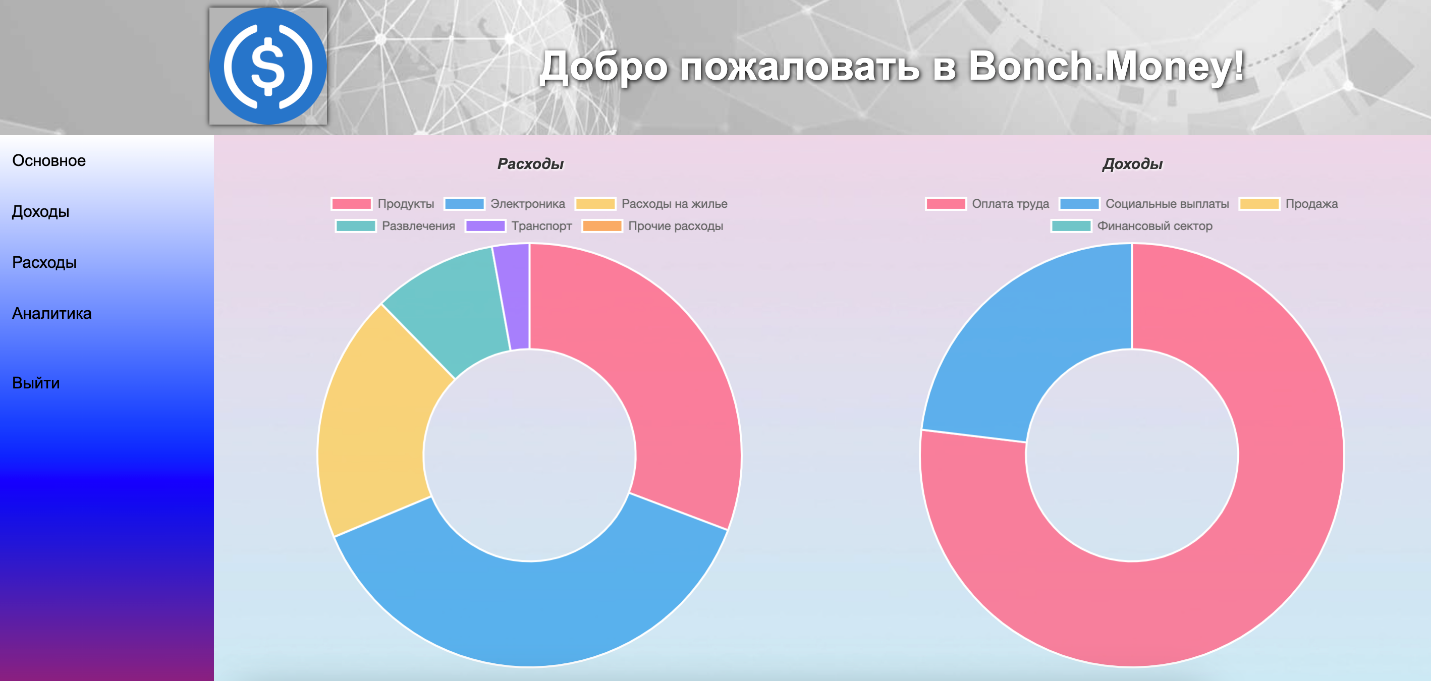


Рисунок 8 – Пример сгенерированных диаграмм доходов и расходов по категориям.

# **ЗАКЛЮЧЕНИЕ**

В ходе курсовой работы:

* Определен программный инструментарий, применяемый для разработки веб-приложения.
* Разработано приложение, позволяющее управлять учетом личных затрат пользователя. Приложение реализует функционал по учету доходов и расходов пользователя, а также позволяет генерировать диаграммы доходов и расходов по категориям.
* Проведено ручное тестирование программы.

# **СПИСОК ЛИТЕРАТУРЫ**

1. Белая Т. И. Лекции по дисциплине «Разработка Java-приложений управления телекоммуникациями»
2. Bealdung [Электронный ресурс] URL: <https://www.baeldung.com/>
3. Jakarta Serlvet Api Javadoc [Электронный ресурс] URL: <https://javadoc.io/doc/jakarta.servlet/jakarta.servlet-api/6.0.0/jakarta.servlet/module-summary.html>
4. Maven Documentation [Электронный ресурс] URL: <https://maven.apache.org/>

# **ПРИЛОЖЕНИЕ**

В приложениях приведен исходный код всей программы.

* Код всех java-файлов приведен в приложении А;
* JSP и CSS файлы представлены в приложении Б;
* Файлы конфигурации, например pom.xml, configuration.properties, script.sql и другие представлены в приложении В.

Весь исходный код также доступен по ссылке на репозиторий с проектом:

## **ПРИЛОЖЕНИЕ A. Файлы .java**

|  |
| --- |
| **Category.java** |
| package com.example.ilnitskiy\_9.model;  public class Category {  private int id;  private String name;   public Category(int id, String name) {  this.id = id;  this.name = name;  }   public int getId() {  return id;  }   public void setId(int id) {  this.id = id;  }   public String getName() {  return name;  }   public void setName(String name) {  this.name = name;  }   @Override  public String toString() {  return name;  } } |

|  |
| --- |
| **Expense.java** |
| package com.example.ilnitskiy\_9.model;  import java.sql.Timestamp; import java.text.SimpleDateFormat;  public class Expense {  private int id;  private int userId;  private int amount;  private String source;  private int categoryId;  private String category;  private Timestamp timestamp;  private String dateFormatted;   public Expense(int id, int user\_id, int amount, String source, int categoryId, Timestamp timestamp) {  this.id = id;  this.userId = user\_id;  this.amount = amount;  this.source = source;  this.categoryId = categoryId;  this.timestamp = timestamp;  }   public int getId() {  return id;  }   public void setId(int id) {  this.id = id;  }   public int getUserId() {  return userId;  }   public void setUserId(int user\_id) {  this.userId = user\_id;  }   public int getAmount() {  return amount;  }   public void setAmount(int amount) {  this.amount = amount;  }   public String getSource() {  return source;  }   public void setSource(String source) {  this.source = source;  }   public int getCategoryId() {  return categoryId;  }   public void setCategoryId(int categoryId) {  this.categoryId = categoryId;  }   public String getCategory() {  return category;  }   public void setCategory(String category) {  this.category = category;  }  public Timestamp getTimestamp() {  return timestamp;  }   public void setTimestamp(Timestamp timestamp) {  this.timestamp = timestamp;  }   public String getDateFormatted() {  return new SimpleDateFormat("dd.MM.yyyy HH:mm").format(timestamp);  }   @Override  public String toString() {  return "Expense{" +  "id=" + id +  ", userId=" + userId +  ", amount=" + amount +  ", source='" + source + '\'' +  ", categoryId=" + categoryId +  ", category='" + category + '\'' +  ", timestamp=" + timestamp +  ", dateFormatted='" + dateFormatted + '\'' +  '}';  } } |

|  |
| --- |
| **Income.java** |
| package com.example.ilnitskiy\_9.model;  import java.sql.Timestamp; import java.text.SimpleDateFormat;  public class Income {  private int id;  private int userId;  private int amount;  private String source;  private int categoryId;  private String category;  private Timestamp timestamp;  private String dateFormatted;   public Income(int id, int user\_id, int amount, String source, int categoryId, Timestamp timestamp) {  this.id = id;  this.userId = user\_id;  this.amount = amount;  this.source = source;  this.categoryId = categoryId;  this.timestamp = timestamp;  }   public int getId() {  return id;  }   public void setId(int id) {  this.id = id;  }   public int getUserId() {  return userId;  }   public void setUserId(int user\_id) {  this.userId = user\_id;  }   public int getAmount() {  return amount;  }   public void setAmount(int amount) {  this.amount = amount;  }   public String getSource() {  return source;  }   public void setSource(String source) {  this.source = source;  }   public int getCategoryId() {  return categoryId;  }   public void setCategoryId(int categoryId) {  this.categoryId = categoryId;  }   public String getCategory() {  return category;  }   public void setCategory(String category) {  this.category = category;  }  public Timestamp getTimestamp() {  return timestamp;  }   public void setTimestamp(Timestamp timestamp) {  this.timestamp = timestamp;  }   public String getDateFormatted() {  return new SimpleDateFormat("dd.MM.yyyy HH:mm").format(timestamp);  }   @Override  public String toString() {  return "Expense{" +  "id=" + id +  ", userId=" + userId +  ", amount=" + amount +  ", source='" + source + '\'' +  ", categoryId=" + categoryId +  ", category='" + category + '\'' +  ", timestamp=" + timestamp +  ", dateFormatted='" + dateFormatted + '\'' +  '}';  } } |

|  |
| --- |
| **User.java** |
| package com.example.ilnitskiy\_9.model;  public class User {  private int id;  private String name;  private String login;  private String password;   public User(int id, String name, String login,String password) {  this.id = id;  this.name = name;  this.login = login;  this.password = password;  }   public int getId() {  return id;  }   public void setId(int id) {  this.id = id;  }   public String getName() {  return name;  }   public void setName(String name) {  this.name = name;  }   public String getLogin() {  return login;  }   public void setLogin(String login) {  this.login = login;  }   public String getPassword() {  return password;  }   public void setPassword(String password) {  this.password = password;  } } |

|  |
| --- |
| **DataBaseService.java** |
| package com.example.ilnitskiy\_9.service.db;  import java.sql.\*;  public class DataBaseService {   public Connection getConnect(){  try {  Class.*forName*("org.postgresql.Driver");  } catch (ClassNotFoundException e) {  e.printStackTrace();  }  final String url = "jdbc:postgresql://217.107.219.154:49307/bonch\_2105188?characterEncoding=UTF-8";  final String user = "bonch\_2105188";  final String password = "AcemoheRIIs=";  Connection conn = null;  try {  conn = DriverManager.*getConnection*(url, user, password);  } catch (SQLException e) {  e.printStackTrace();  }  return conn;  }   public ResultSet select(String sql){  Statement statement = null;  try {  statement = getConnect().createStatement();  ResultSet rs = statement.executeQuery(sql);  return rs;  } catch (SQLException throwables) {  System.*out*.println(throwables.getMessage());  return null;  }  } } |

|  |
| --- |
| **ExpenseCategoryRepository.java** |
| package com.example.ilnitskiy\_9.service.db;  import com.example.ilnitskiy\_9.model.Category;  import java.sql.Connection; import java.sql.PreparedStatement; import java.sql.ResultSet; import java.sql.SQLException; import java.util.ArrayList; import java.util.List;  public class ExpenseCategoryRepository {  private final DataBaseService dataBaseService;  static final String *SELECT\_ALL* = "SELECT \* FROM expense\_category ORDER BY id";  static final String *INSERT* = "INSERT INTO expense\_category (name) values (?)";  static final String *DELETE* = "DELETE FROM expense\_category where id=?";  static final String *UPDATE* = "UPDATE expense\_category SET name=? where id=?";  public ExpenseCategoryRepository() {  this.dataBaseService = new DataBaseService();  }  public List<Category> getAllCategories() {  List<Category> categories = new ArrayList<>();  try {  ResultSet resultSet = dataBaseService.select(*SELECT\_ALL*);  while (resultSet.next()) {  Category category = new Category(  resultSet.getInt("id"),  resultSet.getString("name")  );  categories.add(category);  }  } catch (SQLException e) {  System.*out*.println(e.getMessage());  }  return categories;  }  public boolean addCategory(Category category) {  Connection conn = dataBaseService.getConnect();  try {  PreparedStatement statement = conn.prepareStatement(*INSERT*);  statement.setString(1,category.getName());  return statement.executeUpdate() > 0;  } catch (SQLException e) {  System.*out*.println(e.getMessage());  }  return false;  }  public boolean editCategory(Category category) {  Connection conn = dataBaseService.getConnect();  try {  PreparedStatement statement = conn.prepareStatement(*UPDATE*);  statement.setString(1, category.getName());  statement.setInt(2, category.getId());  return statement.executeUpdate() > 0;  } catch (SQLException e) {  System.*out*.println(e.getMessage());  }  return false;  }  public boolean deleteCategory(Integer id) {  Connection conn = dataBaseService.getConnect();  try {  PreparedStatement statement = conn.prepareStatement(*DELETE*);  statement.setInt(1, id);  return statement.executeUpdate() > 0;  } catch (SQLException e) {  System.out.println(e.getMessage());  }  return false;  }  } |

|  |
| --- |
| **ExpenseRepository.java** |
| package com.example.ilnitskiy\_9.service.db;  import com.example.ilnitskiy\_9.model.Category; import com.example.ilnitskiy\_9.model.Expense;  import java.sql.Connection; import java.sql.PreparedStatement; import java.sql.ResultSet; import java.sql.SQLException; import java.util.\*;  public class ExpenseRepository {  private final DataBaseService dataBaseService;  static final String *SELECT\_BY\_USERID* = "SELECT \* from expense JOIN expense\_category" +  " on expense\_category.id=expense.category where expense.user\_id=? ORDER BY timestamp desc";  static final String *INSERT* = "INSERT INTO expense (user\_id,amount,source,category,timestamp) values (?,?,?,?,NOW())";  static final String *DELETE* = "DELETE FROM expense where id=?";  static final String *UPDATE* = "UPDATE expense SET amount=?,source=?,category=? where id=?";  static final String *SELECT\_SUM\_EXPENSES\_OF\_USER\_GROUP\_BY\_CATEGORY* =  "SELECT expense\_category.id, expense\_category.name, COALESCE(SUM(amount), 0) AS sum" +  " FROM expense\_category" +  " LEFT JOIN expense ON expense.category = expense\_category.id AND expense.user\_id=?" +  " GROUP BY expense\_category.id, expense\_category.name" +  " ORDER BY expense\_category.id";   public ExpenseRepository() {  this.dataBaseService = new DataBaseService();  }  public List<Expense> getExpensesByUserId(Integer userId) {  List<Expense> expenses = new ArrayList<>();  Connection conn = dataBaseService.getConnect();  try {  PreparedStatement statement = conn.prepareStatement(*SELECT\_BY\_USERID*);  statement.setInt(1, userId);  ResultSet resultSet = statement.executeQuery();  while (resultSet.next()) {  Expense expense = new Expense(  resultSet.getInt("id"),  resultSet.getInt("user\_id"),  resultSet.getInt("amount"),  resultSet.getString("source"),  resultSet.getInt("category"),  resultSet.getTimestamp("timestamp")  );  expense.setCategory(resultSet.getString("name"));  expenses.add(expense);  }  } catch (SQLException e) {  e.printStackTrace();  }  return expenses;  }  public Map<Category,Integer> getUserExpensesByCategories(Integer userId) {  List<Expense> expenses = new ArrayList<>();  Connection conn = dataBaseService.getConnect();  Map<Category,Integer> map = new LinkedHashMap<>();  try {  PreparedStatement statement = conn.prepareStatement(*SELECT\_SUM\_EXPENSES\_OF\_USER\_GROUP\_BY\_CATEGORY*);  statement.setInt(1, userId);  ResultSet resultSet = statement.executeQuery();  while (resultSet.next()) {  Category category = new Category(  resultSet.getInt("id"),  resultSet.getString("name")  );  Integer sumExpenses = resultSet.getInt("sum");  map.put(category,sumExpenses);  }  } catch (SQLException e) {  e.printStackTrace();  }  return map;  }  public boolean addExpense(Expense expense) {  Connection conn = dataBaseService.getConnect();  try {  PreparedStatement statement = conn.prepareStatement(*INSERT*);  statement.setInt(1, expense.getUserId());  statement.setInt(2, expense.getAmount());  statement.setString(3, expense.getSource());  statement.setInt(4, expense.getCategoryId());  return statement.executeUpdate() > 0;  } catch (SQLException e) {  System.*out*.println(e.getMessage());  }  return false;  }  public boolean editExpense(Expense expense) {  Connection conn = dataBaseService.getConnect();  try {  PreparedStatement statement = conn.prepareStatement(*UPDATE*);  statement.setInt(1, expense.getAmount());  statement.setString(2, expense.getSource());  statement.setInt(3, expense.getCategoryId());  statement.setInt(4,expense.getId());  return statement.executeUpdate() > 0;  } catch (SQLException e) {  System.*out*.println(e.getMessage());  }  return false;  }  public boolean deleteExpense(Integer id) {  Connection conn = dataBaseService.getConnect();  try {  PreparedStatement statement = conn.prepareStatement(*DELETE*);  statement.setInt(1, id);  return statement.executeUpdate() > 0;  } catch (SQLException e) {  System.*out*.println(e.getMessage());  }  return false;  } } |

|  |
| --- |
| **IncomeCategoryRepository.java** |
| **package com.example.ilnitskiy\_9.service.db;**  import com.example.ilnitskiy\_9.model.Category;  import java.sql.ResultSet; import java.sql.SQLException; import java.util.ArrayList; import java.util.List;  public class IncomeCategoryRepository {  private final DataBaseService dataBaseService;  static final String *SELECT\_ALL* = "SELECT \* FROM income\_category ORDER BY id";  static final String *INSERT* = "INSERT INTO income\_category (name) values (?)";  static final String *DELETE* = "DELETE FROM income\_category where id=?";  static final String *UPDATE* = "UPDATE income\_category SET name=? where id=?";  public IncomeCategoryRepository() {  this.dataBaseService = new DataBaseService();  }  public List<Category> getAllCategories() {  List<Category> categories = new ArrayList<>();  try {  ResultSet resultSet = dataBaseService.select(*SELECT\_ALL*);  while (resultSet.next()) {  Category category = new Category(  resultSet.getInt("id"),  resultSet.getString("name")  );  categories.add(category);  }  } catch (SQLException e) {  System.*out*.println(e.getMessage());  }  return categories;  } } |

|  |
| --- |
| **IncomeRepository.java** |
| package com.example.ilnitskiy\_9.service.db;  import com.example.ilnitskiy\_9.model.Category; import com.example.ilnitskiy\_9.model.Income;  import java.sql.Connection; import java.sql.PreparedStatement; import java.sql.ResultSet; import java.sql.SQLException; import java.util.\*;  public class IncomeRepository {  private final DataBaseService dataBaseService;  static final String *SELECT\_BY\_USERID* = "SELECT \* from income JOIN income\_category" +  " on income\_category.id=income.category where income.user\_id=? ORDER BY timestamp desc";  static final String *INSERT* = "INSERT INTO income (user\_id,amount,source,category,timestamp) values (?,?,?,?,NOW())";  static final String *DELETE* = "DELETE FROM income where id=?";  static final String *UPDATE* = "UPDATE income SET amount=?,source=?,category=? where id=?";  static final String *SELECT\_SUM\_EXPENSES\_OF\_USER\_GROUP\_BY\_CATEGORY* =  "SELECT income\_category.id, income\_category.name, COALESCE(SUM(amount), 0) AS sum" +  " FROM income\_category" +  " LEFT JOIN income ON income.category = income\_category.id AND income.user\_id=?" +  " GROUP BY income\_category.id, income\_category.name" +  " ORDER BY income\_category.id";   public IncomeRepository() {  this.dataBaseService = new DataBaseService();  }  public List<Income> getIncomesByUserId(Integer userId) {  List<Income> incomes = new ArrayList<>();  Connection conn = dataBaseService.getConnect();  try {  PreparedStatement statement = conn.prepareStatement(*SELECT\_BY\_USERID*);  statement.setInt(1, userId);  ResultSet resultSet = statement.executeQuery();  while (resultSet.next()) {  Income income = new Income(  resultSet.getInt("id"),  resultSet.getInt("user\_id"),  resultSet.getInt("amount"),  resultSet.getString("source"),  resultSet.getInt("category"),  resultSet.getTimestamp("timestamp")  );  income.setCategory(resultSet.getString("name"));  incomes.add(income);  }  } catch (SQLException e) {  e.printStackTrace();  }  return incomes;  }  public Map<Category,Integer> getUserIncomesByCategories(Integer userId) {  List<Income> incomes = new ArrayList<>();  Connection conn = dataBaseService.getConnect();  Map<Category,Integer> map = new LinkedHashMap<>();  try {  PreparedStatement statement = conn.prepareStatement(SELECT\_SUM\_EXPENSES\_OF\_USER\_GROUP\_BY\_CATEGORY);  statement.setInt(1, userId);  ResultSet resultSet = statement.executeQuery();  while (resultSet.next()) {  Category category = new Category(  resultSet.getInt("id"),  resultSet.getString("name")  );  Integer sumIncomes = resultSet.getInt("sum");  map.put(category,sumIncomes);  }  } catch (SQLException e) {  e.printStackTrace();  }  return map;  }  public boolean addIncome(Income income) {  Connection conn = dataBaseService.getConnect();  try {  PreparedStatement statement = conn.prepareStatement(INSERT);  statement.setInt(1, income.getUserId());  statement.setInt(2, income.getAmount());  statement.setString(3, income.getSource());  statement.setInt(4, income.getCategoryId());  return statement.executeUpdate() > 0;  } catch (SQLException e) {  System.out.println(e.getMessage());  }  return false;  }  public boolean editIncome(Income income) {  Connection conn = dataBaseService.getConnect();  try {  PreparedStatement statement = conn.prepareStatement(UPDATE);  statement.setInt(1, income.getAmount());  statement.setString(2, income.getSource());  statement.setInt(3, income.getCategoryId());  statement.setInt(4,income.getId());  return statement.executeUpdate() > 0;  } catch (SQLException e) {  System.out.println(e.getMessage());  }  return false;  }  public boolean deleteIncome(Integer id) {  Connection conn = dataBaseService.getConnect();  try {  PreparedStatement statement = conn.prepareStatement(DELETE);  statement.setInt(1, id);  return statement.executeUpdate() > 0;  } catch (SQLException e) {  System.out.println(e.getMessage());  }  return false;  } } |

|  |
| --- |
| **UserRepository.java** |
| package com.example.ilnitskiy\_9.service.db;  import com.example.ilnitskiy\_9.model.User;  import java.sql.Connection; import java.sql.PreparedStatement; import java.sql.ResultSet; import java.sql.SQLException;  public class UserRepository {  private final DataBaseService dataBaseService;  static final String *SELECT\_BY\_ID* = "SELECT \* from users where id=";  private final String SELECT\_AUTH = "SELECT id FROM users WHERE login = ? AND password = ?";  private final String SELECT\_USER\_EXISTING = "SELECT COUNT(\*) from users where login=?";  private final String CREATE\_USER = "INSERT INTO users (name, login, password) VALUES (?,?,?) RETURNING id";   public UserRepository() {  this.dataBaseService = new DataBaseService();  }   public int getUserIdByUsernameAndPassword(String username,String password) {  Connection conn = dataBaseService.getConnect();  int userId=-1;  try {  PreparedStatement statement = conn.prepareStatement(SELECT\_AUTH);  statement.setString(1, username);  statement.setString(2, password);  ResultSet resultSet = statement.executeQuery();  if (resultSet.next()) {  userId = resultSet.getInt("id");  }  } catch (SQLException e) {  System.*out*.println(e.getMessage());  }  return userId;  }  public int createUser(String name,String login, String password) {  Connection conn = dataBaseService.getConnect();  int userId=-1;  try {  PreparedStatement statement = conn.prepareStatement(CREATE\_USER);  statement.setString(1, name);  statement.setString(2, login);  statement.setString(3, password);  ResultSet resultSet = statement.executeQuery();  if (resultSet.next()) {  return resultSet.getInt("id");  }  } catch (SQLException e) {  System.*out*.println(e.getMessage());  }  return userId;  }   public boolean isUserExistsByLogin(String login) {  Connection conn = dataBaseService.getConnect();  try {  PreparedStatement statement = conn.prepareStatement(SELECT\_USER\_EXISTING);  statement.setString(1, login);  ResultSet resultSet = statement.executeQuery();  if (resultSet.next()) {  return resultSet.getInt("count") != 0;  }  } catch (SQLException e) {  System.*out*.println(e.getMessage());  }  return true;  }   public User getUserById(int id) {  try {  ResultSet resultSet = dataBaseService.select(*SELECT\_BY\_ID*+id);  if (resultSet.next()) {  return new User(  resultSet.getInt("id"),  resultSet.getString("name"),  resultSet.getString("login"),  resultSet.getString("password")  );  }  else {  throw new IllegalArgumentException("No user with id "+id);  }  } catch (SQLException e) {  System.*out*.println(e.getMessage());  return null;  }  } } |

|  |
| --- |
| **LoginService.java** |
| package com.example.ilnitskiy\_9.service;  import com.example.ilnitskiy\_9.service.db.UserRepository;  import java.math.BigInteger; import java.nio.charset.StandardCharsets; import java.security.MessageDigest; import java.security.NoSuchAlgorithmException;  public class LoginService {  private final UserRepository userRepository;   public LoginService() {  this.userRepository = new UserRepository();  }   public int auth(String login, String password) {  String hashedPassword = hashPassword(password);  return userRepository.getUserIdByUsernameAndPassword(login,hashedPassword);  }  public int register(String name,String login,String password) {  if (userRepository.isUserExistsByLogin(login)) {  return -1;  }  String hashedPassword = hashPassword(password);  return userRepository.createUser(name,login,hashedPassword);  }  public String hashPassword(String password) {  MessageDigest digest = null;  try {  digest = MessageDigest.*getInstance*("SHA-256");  } catch (NoSuchAlgorithmException e) {  System.*out*.println(e.getMessage());  }  byte[] hashBytes = digest.digest(password.getBytes(StandardCharsets.*UTF\_8*));   BigInteger hashInt = new BigInteger(1, hashBytes);  return hashInt.toString(16);  } } |

|  |
| --- |
| **AnalyticsServlet.java** |
| package com.example.ilnitskiy\_9.servlets;  import com.example.ilnitskiy\_9.model.Category; import com.example.ilnitskiy\_9.service.db.ExpenseRepository; import com.example.ilnitskiy\_9.service.db.IncomeRepository;  import javax.servlet.ServletException; import javax.servlet.annotation.WebServlet; import javax.servlet.http.HttpServlet; import javax.servlet.http.HttpServletRequest; import javax.servlet.http.HttpServletResponse; import java.io.IOException; import java.util.Map;  @WebServlet("/analytics") public class AnalyticsServlet extends HttpServlet {  private final ExpenseRepository expenseRepository;  private final IncomeRepository incomeRepository;   public AnalyticsServlet() {  this.expenseRepository = new ExpenseRepository();  this.incomeRepository = new IncomeRepository();  }   @Override  protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  Integer userId = (Integer) req.getSession().getAttribute("user\_id");  Map<Category,Integer> expenseMap = expenseRepository.getUserExpensesByCategories(userId);  Map<Category,Integer> incomeMap = incomeRepository.getUserIncomesByCategories(userId);  req.setAttribute("expense\_categories",expenseMap);  req.setAttribute("income\_categories",incomeMap);  req.getRequestDispatcher("/analytics.jsp").forward(req,resp);  } } |

|  |
| --- |
| **AuthenticationFilter.java** |
| package com.example.ilnitskiy\_9.servlets;  import javax.servlet.\*; import javax.servlet.annotation.WebFilter; import javax.servlet.http.HttpServletRequest; import javax.servlet.http.HttpServletResponse; import javax.servlet.http.HttpSession; import java.io.IOException;  @WebFilter("/\*") public class AuthenticationFilter implements Filter {   @Override  public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)  throws IOException, ServletException {  HttpServletRequest httpRequest = (HttpServletRequest) request;  HttpServletResponse httpResponse = (HttpServletResponse) response;  HttpSession session = httpRequest.getSession(false);  boolean isLoggedIn = (session != null && session.getAttribute("user\_id") != null);  String requestURI = httpRequest.getRequestURI();  if (!isLoggedIn && !requestURI.endsWith("/login") && !requestURI.endsWith(".css") &&  !requestURI.endsWith("/register")) {  httpResponse.sendRedirect("/login");  return;  }  chain.doFilter(request, response);  }   } |

|  |
| --- |
| **ExpenseCategoryServlet.java** |
| package com.example.ilnitskiy\_9.servlets;  import com.example.ilnitskiy\_9.model.Category; import com.example.ilnitskiy\_9.service.db.ExpenseCategoryRepository; import com.google.gson.Gson; import com.google.gson.JsonObject;  import javax.servlet.ServletException; import javax.servlet.annotation.WebServlet; import javax.servlet.http.HttpServlet; import javax.servlet.http.HttpServletRequest; import javax.servlet.http.HttpServletResponse; import java.io.IOException; import java.util.List; import java.util.stream.Collectors;  @WebServlet("/expense-category") public class ExpenseCategoryServlet extends HttpServlet {  private final ExpenseCategoryRepository expenseCategoryRepository;   public ExpenseCategoryServlet() {  this.expenseCategoryRepository = new ExpenseCategoryRepository();  }   protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  List<Category> categories = expenseCategoryRepository.getAllCategories();  req.setAttribute("categories",categories);  req.getRequestDispatcher("/categories.jsp").forward(req,resp);  }   protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws IOException {  String requestBody = req.getReader().lines().collect(Collectors.*joining*(System.*lineSeparator*()));  Category category = new Gson().fromJson(requestBody, Category.class);  if (expenseCategoryRepository.addCategory(category)) {  resp.getWriter().println("Success");  }  else {  resp.setStatus(HttpServletResponse.*SC\_BAD\_REQUEST*);  resp.getWriter().write("Error while creating category");  }  }   protected void doDelete(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  int categoryId = Integer.*parseInt*(req.getParameter("id"));  if (!expenseCategoryRepository.deleteCategory(categoryId)) {  resp.setStatus(HttpServletResponse.*SC\_NOT\_FOUND*);  }  }   protected void doPut(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  String requestBody = req.getReader().lines().collect(Collectors.*joining*(System.*lineSeparator*()));  Category category = new Gson().fromJson(requestBody, Category.class);  if (expenseCategoryRepository.editCategory(category)) {  resp.getWriter().println("Success");  }  else {  resp.setStatus(HttpServletResponse.*SC\_BAD\_REQUEST*);  JsonObject error = new JsonObject();  error.addProperty("message","No expense with id "+category.getId());  resp.setContentType("application/json");  resp.getWriter().write(error.toString());  }  } } |

|  |
| --- |
| **ExpenseServlet.java** |
| package com.example.ilnitskiy\_9.servlets;  import com.example.ilnitskiy\_9.model.Expense; import com.example.ilnitskiy\_9.service.db.ExpenseCategoryRepository; import com.example.ilnitskiy\_9.service.db.ExpenseRepository; import com.google.gson.Gson; import com.google.gson.JsonObject;  import javax.servlet.ServletException; import javax.servlet.annotation.WebServlet; import javax.servlet.http.HttpServlet; import javax.servlet.http.HttpServletRequest; import javax.servlet.http.HttpServletResponse; import java.io.IOException; import java.util.List; import java.util.stream.Collectors;  @WebServlet("/expense") public class ExpenseServlet extends HttpServlet {  private final ExpenseRepository expenseRepository;  private final ExpenseCategoryRepository expenseCategoryRepository;   public ExpenseServlet() {  this.expenseRepository = new ExpenseRepository();  this.expenseCategoryRepository = new ExpenseCategoryRepository();  }   public void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  Integer userId = (Integer) req.getSession().getAttribute("user\_id");  List<Expense> expenses = expenseRepository.getExpensesByUserId(userId);  req.setAttribute("expenses",expenses);  req.setAttribute("expense\_categories",expenseCategoryRepository.getAllCategories());  req.getRequestDispatcher("/expense-history.jsp").forward(req,resp);  }   protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws IOException {  String requestBody = req.getReader().lines().collect(Collectors.*joining*(System.*lineSeparator*()));  Expense expense = new Gson().fromJson(requestBody, Expense.class);  if (expenseRepository.addExpense(expense)) {  resp.getWriter().println("Success");  }  else {  resp.setStatus(HttpServletResponse.*SC\_BAD\_REQUEST*);  JsonObject error = new JsonObject();  error.addProperty("message","No user with id "+expense.getUserId());  resp.setContentType("application/json");  resp.getWriter().write(error.toString());  }  }   protected void doDelete(HttpServletRequest req, HttpServletResponse resp) {  int expenseId = Integer.*parseInt*(req.getParameter("id"));  if (!expenseRepository.deleteExpense(expenseId)) {  resp.setStatus(HttpServletResponse.*SC\_NOT\_FOUND*);  }  }   @Override  protected void doPut(HttpServletRequest req, HttpServletResponse resp) throws IOException {  String requestBody = req.getReader().lines().collect(Collectors.*joining*(System.*lineSeparator*()));  Expense expense = new Gson().fromJson(requestBody, Expense.class);  if (expenseRepository.editExpense(expense)) {  resp.getWriter().println("Success");  }  else {  resp.setStatus(HttpServletResponse.*SC\_BAD\_REQUEST*);  JsonObject error = new JsonObject();  error.addProperty("message","No expense with id "+expense.getId());  resp.setContentType("application/json");  resp.getWriter().write(error.toString());  }  } } |

|  |
| --- |
| **IncomeCategoryRepository.java** |
| package com.example.ilnitskiy\_9.servlets;  import com.example.ilnitskiy\_9.model.Category; import com.example.ilnitskiy\_9.service.db.IncomeCategoryRepository;  import javax.servlet.ServletException; import javax.servlet.http.HttpServlet; import javax.servlet.http.HttpServletRequest; import javax.servlet.http.HttpServletResponse; import java.io.IOException; import java.util.List;  public class IncomeCategoryServlet extends HttpServlet {  private final IncomeCategoryRepository incomeCategoryRepository;   public IncomeCategoryServlet() {  this.incomeCategoryRepository = new IncomeCategoryRepository();  }   @Override  protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  List<Category> categories = incomeCategoryRepository.getAllCategories();  req.setAttribute("categories",categories);  req.getRequestDispatcher("/categories.jsp").forward(req,resp);  } } |

|  |
| --- |
| **IncomeServlet.java** |
| package com.example.ilnitskiy\_9.servlets;  import com.example.ilnitskiy\_9.model.Income; import com.example.ilnitskiy\_9.service.db.IncomeCategoryRepository; import com.example.ilnitskiy\_9.service.db.IncomeRepository; import com.google.gson.Gson; import com.google.gson.JsonObject;  import javax.servlet.ServletException; import javax.servlet.annotation.WebServlet; import javax.servlet.http.HttpServlet; import javax.servlet.http.HttpServletRequest; import javax.servlet.http.HttpServletResponse; import java.io.IOException; import java.util.List; import java.util.stream.Collectors;  @WebServlet("/income") public class IncomeServlet extends HttpServlet {  private final IncomeRepository incomeRepository;  private final IncomeCategoryRepository incomeCategoryRepository;   public IncomeServlet() {  this.incomeRepository = new IncomeRepository();  this.incomeCategoryRepository = new IncomeCategoryRepository();  }   public void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  Integer userId = (Integer) req.getSession().getAttribute("user\_id");  List<Income> incomes = incomeRepository.getIncomesByUserId(userId);  req.setAttribute("incomes",incomes);  req.setAttribute("userId",userId);  req.setAttribute("income\_categories",incomeCategoryRepository.getAllCategories());  req.getRequestDispatcher("/income-history.jsp").forward(req,resp);  }   protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  String requestBody = req.getReader().lines().collect(Collectors.*joining*(System.*lineSeparator*()));  Income income = new Gson().fromJson(requestBody, Income.class);  if (incomeRepository.addIncome(income)) {  resp.getWriter().println("Success");  }  else {  resp.setStatus(HttpServletResponse.*SC\_BAD\_REQUEST*);  JsonObject error = new JsonObject();  error.addProperty("message","No user with id "+income.getUserId());  resp.setContentType("application/json");  resp.getWriter().write(error.toString());  }  }   protected void doDelete(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  int incomeId = Integer.*parseInt*(req.getParameter("id"));  if (!incomeRepository.deleteIncome(incomeId)) {  resp.setStatus(HttpServletResponse.*SC\_NOT\_FOUND*);  }  }   protected void doPut(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  String requestBody = req.getReader().lines().collect(Collectors.*joining*(System.*lineSeparator*()));  Income income = new Gson().fromJson(requestBody, Income.class);  if (incomeRepository.editIncome(income)) {  resp.getWriter().println("Success");  }  else {  resp.setStatus(HttpServletResponse.*SC\_BAD\_REQUEST*);  JsonObject error = new JsonObject();  error.addProperty("message","No income with id "+income.getId());  resp.setContentType("application/json");  resp.getWriter().write(error.toString());  }  } } |

|  |
| --- |
| **LoginServlet.java** |
| package com.example.ilnitskiy\_9.servlets;  import com.example.ilnitskiy\_9.service.LoginService;  import javax.servlet.ServletException; import javax.servlet.annotation.WebServlet; import javax.servlet.http.HttpServlet; import javax.servlet.http.HttpServletRequest; import javax.servlet.http.HttpServletResponse; import java.io.IOException;  @WebServlet("/login") public class LoginServlet extends HttpServlet {  private final LoginService loginService;   public LoginServlet() {  this.loginService = new LoginService();  }   @Override  protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  req.setAttribute("errorMessage", "");  req.getRequestDispatcher("/login.jsp").forward(req, resp);  resp.setContentType("text/html");  super.doGet(req, resp);  }  @Override  protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  String username = req.getParameter("username");  String password = req.getParameter("password");  int userId = loginService.auth(username, password);  if (userId != -1) {  req.getSession().setAttribute("user\_id", userId);  resp.sendRedirect( "/dashboard");  } else {  req.setAttribute("errorMessage", "error login or pass");  req.getRequestDispatcher("/login.jsp").forward(req, resp);  super.doPost(req, resp);  }  } } |

|  |
| --- |
| **RegisterServlet.java** |
| package com.example.ilnitskiy\_9.servlets;  import com.example.ilnitskiy\_9.service.LoginService;  import javax.servlet.ServletException; import javax.servlet.annotation.WebServlet; import javax.servlet.http.HttpServlet; import javax.servlet.http.HttpServletRequest; import javax.servlet.http.HttpServletResponse; import java.io.IOException;  @WebServlet("/register") public class RegisterServlet extends HttpServlet {  private final LoginService loginService;   public RegisterServlet() {  this.loginService = new LoginService();  }   @Override  protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  req.setAttribute("errorMessage", "");  req.getRequestDispatcher("/register.jsp").forward(req, resp);  resp.setContentType("text/html");  super.doGet(req, resp);  }  @Override  protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  String name = req.getParameter("name");  String login = req.getParameter("login");  String password = req.getParameter("password");  int userId = loginService.register(name,login,password);  if (userId != -1) {  req.getSession().setAttribute("user\_id", userId);  resp.sendRedirect( "/dashboard");  } else {  req.setAttribute("errorMessage", "Логин занят!");  req.getRequestDispatcher("/register.jsp").forward(req, resp);  super.doPost(req, resp);  }  } } |

|  |
| --- |
| **UserServlet.java** |
| package com.example.ilnitskiy\_9.servlets;  import com.example.ilnitskiy\_9.model.Expense; import com.example.ilnitskiy\_9.model.User; import com.example.ilnitskiy\_9.service.db.ExpenseCategoryRepository; import com.example.ilnitskiy\_9.service.db.ExpenseRepository; import com.example.ilnitskiy\_9.service.db.UserRepository;  import javax.servlet.ServletException; import javax.servlet.annotation.WebServlet; import javax.servlet.http.HttpServlet; import javax.servlet.http.HttpServletRequest; import javax.servlet.http.HttpServletResponse; import java.io.IOException; import java.util.List;  @WebServlet("/dashboard") public class UserServlet extends HttpServlet {  private final UserRepository userRepository = new UserRepository();  private final ExpenseCategoryRepository expenseCategoryRepository = new ExpenseCategoryRepository();  private final ExpenseRepository expenseRepository = new ExpenseRepository();   public void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  Integer userId = (Integer) req.getSession().getAttribute("user\_id");  User user = userRepository.getUserById(userId);  List<Expense> expenses = expenseRepository.getExpensesByUserId(userId);  req.setAttribute("expense\_categories", expenseRepository.getUserExpensesByCategories(userId));  req.setAttribute("user", user);  req.setAttribute("expenses",expenses);  req.getRequestDispatcher("/userCabinet.jsp").forward(req, resp);  } } |

## **ПРИЛОЖЕНИЕ Б. Файлы .jsp, .css и .js**

|  |
| --- |
| **analytics.jsp** |
| <%@ page import="com.google.gson.Gson" %> <%@ page import="com.example.ilnitskiy\_9.model.Category" %> <%@ page import="java.util.Map" %> <%@ page contentType="text/html;charset=UTF-8" language="java" %> <script src="https://cdn.jsdelivr.net/npm/chart.js"></script> <html> <head>  <title>Аналитика</title>  <link rel="stylesheet" href="/styles/userCabinet.css">  <link rel="stylesheet" href="/styles/navbar.css">  <link rel="stylesheet" href="/styles/main.css">  <style>  .chart-row {  display: flex;  justify-content: space-around;  }   .chart-container {  width: 40%;  text-align: center;  }   .chart-label {  font-weight: bold;  margin-top: 20px;  }  </style> </head> <body>  <%  Map<Category,Integer> expenseMap = (Map<Category, Integer>) request.getAttribute("expense\_categories");  Map<Category,Integer> incomeMap = (Map<Category, Integer>) request.getAttribute("income\_categories");  %>  <jsp:include page="main.html"/>  <div class="content chart-row">  <div class="chart-container">  <p class="chart-label">Расходы</p>  <canvas id="pieChart"></canvas>  </div>  <div class="chart-container">  <p class="chart-label">Доходы</p>  <canvas id="pieChart1"></canvas>  </div>  </div>  <script>  let categoryMap = <%= new Gson().toJson(expenseMap) %>;  let labels = Object.keys(categoryMap);  let data = Object.values(categoryMap);  let ctx = document.getElementById('pieChart').getContext('2d');  let pieChart = new Chart(ctx, {  type: 'doughnut',  data: {  labels: labels,  datasets: [{  data: data,  backgroundColor: [  'rgba(255, 99, 132, 0.8)',  'rgba(54, 162, 235, 0.8)',  'rgba(255, 206, 86, 0.8)',  'rgba(75, 192, 192, 0.8)',  'rgba(153, 102, 255, 0.8)',  'rgba(255, 159, 64, 0.8)',  'rgba(0, 128, 0, 0.8)',  'rgba(128, 0, 128, 0.8)',  ],  }],  },  });   let categoryMapIncome = <%= new Gson().toJson(incomeMap) %>;  let labelsIncome = Object.keys(categoryMapIncome);  let dataIncome = Object.values(categoryMapIncome);  let ctxIncome = document.getElementById('pieChart1').getContext('2d');  let pieChartIncome = new Chart(ctxIncome, {  type: 'doughnut',  data: {  labels: labelsIncome,  datasets: [{  data: dataIncome,  backgroundColor: [  'rgba(255, 99, 132, 0.8)',  'rgba(54, 162, 235, 0.8)',  'rgba(255, 206, 86, 0.8)',  'rgba(75, 192, 192, 0.8)',  'rgba(153, 102, 255, 0.8)',  'rgba(255, 159, 64, 0.8)',  'rgba(0, 128, 0, 0.8)',  'rgba(128, 0, 128, 0.8)',  ],  }],  },  });  </script>  </body>  </html> |

|  |
| --- |
| **expense-history.jsp** |
| <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %> <%@ page contentType="text/html;charset=UTF-8" language="java" %> <html> <head>  <title>История расходов</title>  <link rel="stylesheet" href="/styles/finance-history.css">  <link rel="icon" href="/img/logo.png" type="image/x-icon">  <link rel="shortcut icon" href="/img/logo.png" type="image/x-icon">  <link rel="stylesheet" href="/styles/main.css">  <link rel="stylesheet" href="/styles/userCabinet.css"> </head> <body>  <jsp:include page="main.html"/>  <div class="content">  <div class="recent-purchases">  <h2>Последние покупки</h2>  <table>  <thead>  <tr>  <th>Дата</th>  <th>Сумма</th>  <th>Описание</th>  <th>Категория</th>  <th>Действие</th>  </tr>  </thead>  <tbody>  <c:forEach items="${expenses}" var="expense">  <tr>  <td>${expense.dateFormatted}</td>  <td>${expense.amount}</td>  <td>${expense.source}</td>  <td>${expense.category}</td>  <td>  <button class="edit-button"  onclick="*openEditExpenseModal*(${expense.id},${expense.amount},'${expense.source}')"  >Изменить  </button>  <button class="delete-button" type="submit" onclick="*deleteExpense*(${expense.id})">Удалить  </button>  </td>  </tr>  </c:forEach>  </tbody>  </table>  </div>  </div>  <div id="createExpenseModal" class="modal">  <div class="modal-content">  <span class="close" onclick="*closeCreateExpenseModal*()">&times;</span>  <h2>Добавить запись о покупке</h2>  <form id="createExpenseForm">  <input type="hidden" id="user\_id" name="user\_id" required value="${user.id}">  <label for="amount">Сумма:</label>  <input type="number" id="amount" name="amount" required>   <label for="source">Покупка:</label>  <input type="text" id="source" name="source" maxlength="50" required>   <label for="category\_id">Категория:</label>  <select id=category\_id name="category\_id">  <option value="0">Выберите категорию</option>  <c:forEach items="${expense\_categories}" var="category">  <option value="${category.id}">${category.name}</option>  </c:forEach>  </select>  <button type="submit">Создать</button>  <div id="message"></div>  </form>  </div>  </div>  <div id="editExpenseModal" class="modal">  <div class="modal-content">  <span class="close" onclick="*closeEditExpenseModal*()">&times;</span>  <h2>Редактировать запись о покупке</h2>  <form id="editExpenseForm">  <input type="hidden" id="edit-expense-id" name="user\_id" required>  <label for="amount">Сумма:</label>  <input type="number" id="edit-amount" name="amount" required value="">   <label for="source">Покупка:</label>  <input type="text" id="edit-source" name="source" maxlength="50" required>   <label for="category\_id">Категория:</label>  <select id="edit-category-id" name="category\_id">  <option value="0">Выберите категорию</option>  <c:forEach items="${expense\_categories}" var="category">  <option value="${category.id}">${category.name}</option>  </c:forEach>  </select>  <button type="submit">Сохранить</button>  <div id="edit-message"></div>  </form>  </div>  </div>  <script src="/scripts/expense.js"></script> </body> </html> |

|  |
| --- |
| **income-history.jsp** |
| <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %> <%@ page contentType="text/html;charset=UTF-8" language="java" %> <html> <head>  <title>История расходов</title>  <link rel="stylesheet" href="/styles/finance-history.css">  <link rel="icon" href="/img/logo.png" type="image/x-icon">  <link rel="shortcut icon" href="/img/logo.png" type="image/x-icon">  <link rel="stylesheet" href="/styles/main.css">  <link rel="stylesheet" href="/styles/userCabinet.css"> </head> <body>  <jsp:include page="main.html"/>  <div class="content">  <button class="create-button" onclick="*openCreateIncomeModal*()">Добавить запись о доходе</button>  <div class="recent-purchases">  <h2>Последняя информация о доходах</h2>  <table>  <thead>  <tr>  <th>Дата</th>  <th>Сумма</th>  <th>Источник</th>  <th>Категория</th>  <th>Действие</th>  </tr>  </thead>  <tbody>  <c:forEach items="${incomes}" var="income">  <tr>  <td>${income.dateFormatted}</td>  <td>${income.amount}</td>  <td>${income.source}</td>  <td>${income.category}</td>  <td>  <button class="edit-button"  onclick="*openEditIncomeModal*(${income.id},${income.amount},'${income.source}')"  >Изменить  </button>  <button class="delete-button" type="submit" onclick="*deleteIncome*(${income.id})">Удалить  </button>  </td>  </tr>  </c:forEach>  </tbody>  </table>  </div>  </div>  <div id="createIncomeModal" class="modal">  <div class="modal-content">  <span class="close" onclick="*closeCreateIncomeModal*()">&times;</span>  <h2>Добавить запись о доходе</h2>  <form id="createIncomeForm">  <input type="hidden" id="user\_id" name="user\_id" required value="${userId}">  <label for="amount">Сумма:</label>  <input type="number" id="amount" name="amount" required>   <label for="source">Источник:</label>  <input type="text" id="source" name="source" maxlength="50" required>   <label for="category\_id">Категория:</label>  <select id=category\_id name="category\_id">  <option value="0">Выберите категорию</option>  <c:forEach items="${income\_categories}" var="category">  <option value="${category.id}">${category.name}</option>  </c:forEach>  </select>  <button type="submit">Создать</button>  <div id="message"></div>  </form>  </div>  </div>  <div id="editIncomeModal" class="modal">  <div class="modal-content">  <span class="close" onclick="*closeEditIncomeModal*()">&times;</span>  <h2>Редактировать запись о доходе</h2>  <form id="editIncomeForm">  <input type="hidden" id="edit-income-id" name="user\_id" required>  <label for="amount">Сумма:</label>  <input type="number" id="edit-amount" name="amount" required value="">   <label for="source">Источник:</label>  <input type="text" id="edit-source" name="source" maxlength="50" required>   <label for="category\_id">Категория:</label>  <select id="edit-category-id" name="category\_id">  <option value="0">Выберите категорию</option>  <c:forEach items="${income\_categories}" var="category">  <option value="${category.id}">${category.name}</option>  </c:forEach>  </select>  <button type="submit">Сохранить</button>  <div id="edit-message"></div>  </form>  </div>  </div>  <script src="/scripts/income.js"></script> </body> </html> |

|  |
| --- |
| **index.jsp** |
| <%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %> <!DOCTYPE html> <html> <head>  <title>BonchMoney</title>  <link rel="stylesheet" href="/styles/navbar.css">  <link rel="stylesheet" href="/styles/index.css">  <link rel="stylesheet" href="/styles/main.css"> </head> <body>  <jsp:include page="main.html" />  <div class="content">  <h1><%= "Добро пожаловать в приложение!" %>  </h1>  <br/>  <a href="/login" >Перейти к списку пользователей</a>  </div> </body> </html> |

|  |
| --- |
| **login.jsp** |
| <%@ page contentType="text/html;charset=UTF-8" language="java" %> <html> <head>  <title>Вход</title>  <link rel="stylesheet" href="/styles/login.css"> </head> <body>  <div class="login-container">  <h2>Авторизация</h2>  <form action="/login" method="post">  <label for="username">Логин:</label>  <input type="text" id="username" name="username" required>  <br>  <label for="password">Пароль:</label>  <input type="password" id="password" name="password" required>  <br>  <button type="submit">Войти</button>  <p>Чтобы зарегистрироваться, нажмите &nbsp;<a href="/register">cюда</a></p>  </form>  <div class="error-message">${errorMessage}</div>  </div> </body> </html> |

|  |
| --- |
| **logout.jsp** |
| <%@ page contentType="text/html;charset=UTF-8" language="java" %> <%  session.removeAttribute("userId");  session.invalidate();  response.sendRedirect("/login"); %> |

|  |
| --- |
| **main.html** |
| <!doctype html> <html lang="en"> <head>  <meta charset="UTF-8">  <meta name="viewport"  content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">  <link rel="stylesheet" type="text/css" href="/styles/main.css">  <link rel="icon" href="/img/logo.png" type="image/x-icon">  <link rel="shortcut icon" href="/img/logo.png" type="image/x-icon">  <title>Welcome</title> </head> <body>  <header>  <a href="#"><img src="/img/logo.png" id="logo" alt="LOGO"></a>  <h1>Добро пожаловать в Bonch.Money!</h1>  </header>  <div class="sidebar">  <a href="/dashboard" style="text-decoration: none;">Основное</a>  <a href="/income">Доходы</a>  <a href="/expense">  Расходы  </a>  <a href="/analytics">Аналитика</a>  <br>  <a href="logout.jsp">Выйти</a>  </div> </body> </html> |

|  |
| --- |
| **register.jsp** |
| <%@ page contentType="text/html;charset=UTF-8" language="java" %> <html> <head>  <title>Регистрация</title>  <link rel="stylesheet" href="/styles/login.css"> </head> <body>  <div class="login-container">  <h2>Форма регистрации</h2>  <form action="/register" method="post">  <label for="name">Имя:</label>  <input type="text" id="name" name="name" maxlength="50" required>  <br>  <label for="login">Логин:</label>  <input type="text" id="login" name="login" maxlength="50" required>  <br>  <label for="password">Пароль:</label>  <input type="password" id="password" name="password" maxlength="50" required>  <br>  <button type="submit">Register</button>  <p>Уже зарегистрированы? &nbsp;<a href="/login">Вход</a></p>  </form>  <div class="error-message">${errorMessage}</div>  </div> </body> </html> |

|  |
| --- |
| **userCabinet.jsp** |
| <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %> <%@ page contentType="text/html;charset=UTF-8" language="java" %> <html> <head>  <title>Личный кабинет</title>  <link rel="icon" href="/img/logo.png" type="image/x-icon">  <link rel="shortcut icon" href="/img/logo.png" type="image/x-icon">  <link rel="stylesheet" href="/styles/userCabinet.css">  <link rel="stylesheet" href="/styles/navbar.css">  <link rel="stylesheet" href="/styles/main.css"> </head> <body>  <jsp:include page="main.html"/>  <div class="content">  <h1>Добрый день, ${user.name}!</h1>  <div class="categories-container">  <c:forEach items="${expense\_categories}" var="category\_map">  <div class="category-card"  onclick="*openCreateExpenseModal*('${category\_map.key.name}',${category\_map.key.id})">  <img class="category-image" src="/img/category.png" alt="${category\_map.key.name}">  <div class="category-title">${category\_map.key.name}</div>  <div class="category-amount">${category\_map.value} ₽</div>  </div>  </c:forEach>  </div>  <div class="card-container">  <h2>Ваши последние покупки</h2>  <c:forEach items="${expenses}" var="expense">  <div class="card">  <h3>${expense.source}</h3>  <p><span>${expense.amount} ₽</span></p>  <p>Категория: <span>${expense.category}</span></p>  <p><span>${expense.dateFormatted}</span></p>  </div>  </c:forEach>  </div>   </div>  <div id="createExpenseModal" class="modal">  <div class="modal-content">  <span class="close" onclick="*closeCreateExpenseModal*()">&times;</span>  <h2 id="modalTitle">Новая покупка в категории </h2>  <form id="createExpenseForm">  <input type="hidden" id="user\_id" name="user\_id" required value="${user.id}">  <label for="amount">Сумма:</label>  <input type="number" id="amount" name="amount" required>   <label for="source">Покупка:</label>  <input type="text" id="source" name="source" maxlength="50" required>   <input id="category\_id" type="text" name="category\_id" hidden>  <button type="submit">Создать</button>  <div id="message"></div>  </form>  </div>  </div>  <div id="editExpenseModal" class="modal">  <div class="modal-content">  <span class="close" onclick="*closeEditExpenseModal*()">&times;</span>  <h2>Редактировать запись о покупке</h2>  <form id="editExpenseForm">  <input type="hidden" id="edit-user-id" name="user\_id" required value="${user.id}">  <input type="hidden" id="edit-expense-id" name="user\_id" required>  <label for="amount">Сумма:</label>  <input type="number" id="edit-amount" name="amount" required value="">   <label for="source">Покупка:</label>  <input type="text" id="edit-source" name="source" maxlength="50" required>   <label for="category\_id">Категория:</label>  <select id="edit-category-id" name="category\_id">  <option value="0">Выберите категорию</option>  <c:forEach items="${expense\_categories}" var="category">  <option value="${category.key.id}">${category.key.name}</option>  </c:forEach>  </select>  <button type="submit">Сохранить</button>  <div id="edit-message"></div>  </form>  </div>  </div>  <script src="/scripts/userCabinet.js"></script> </body> </html> |

|  |
| --- |
| **finance-history.css** |
| body {  font-family: 'Arial', sans-serif;  background-color: #f5f5f5;  color: #333;  margin: 0;  padding: 0; } table {  width: 80%;  margin: 20px auto;  border-collapse: collapse;  background-color: #fff;  box-shadow: 0 0 10px rgba(0, 0, 0, 0.1); }  table, th, td {  border: 1px solid #ddd; }  th, td {  padding: 12px;  text-align: left; }  th {  background-color: #007BFF;  color: #fff; }  tr:hover {  background-color: #f5f5f5; }  button {  padding: 8px 12px;  margin: 4px;  cursor: pointer;  background-color: #007BFF;  color: #fff;  border: none;  border-radius: 4px; } |

|  |
| --- |
| **index.css** |
| body {  font-family: 'Arial', sans-serif;  background-color: #f5f5f5;  color: #333;  margin: 0;  padding: 0; }  .container {  text-align: center;  margin-top: 20px; }  h1 {  color: #007BFF;  }  table {  width: 80%;  margin: 20px auto;  border-collapse: collapse;  background-color: #fff;  box-shadow: 0 0 10px rgba(0, 0, 0, 0.1); }  table, th, td {  border: 1px solid #ddd; }  th, td {  padding: 12px;  text-align: left; }  th {  background-color: #007BFF;  color: #fff; }  tr:hover {  background-color: #f5f5f5; }  button {  padding: 8px 12px;  margin: 4px;  cursor: pointer;  background-color: #007BFF;  color: #fff;  border: none;  border-radius: 4px; } |

|  |
| --- |
| **login.css** |
| body {  font-family: 'Arial', sans-serif;  background-color: #f5f5f5;  display: flex;  align-items: center;  justify-content: center;  height: 100vh;  margin: 0; }  .login-container {  background-color: #fff;  box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);  padding: 20px;  border-radius: 8px;  text-align: center; }  .login-container label {  display: block;  margin-bottom: 10px; }  .login-container input {  width: 100%;  padding: 10px;  margin-bottom: 15px;  box-sizing: border-box; }  .login-container button {  padding: 10px 20px;  background-color: #007BFF;  color: #fff;  border: none;  border-radius: 4px;  cursor: pointer; }  .login-container button:hover {  background-color: #0056b3; }  .error-message {  color: #dc3545;  margin-top: 10px; } |

|  |
| --- |
| **main.css** |
| body {  font-family: 'Arial', sans-serif;  background-color: #f5f5f5;  display: flex;  align-items: center;  justify-content: center;  height: 100vh;  margin: 0; }  .login-container {  background-color: #fff;  box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);  padding: 20px;  border-radius: 8px;  text-align: center; }  .login-container label {  display: block;  margin-bottom: 10px; }  .login-container input {  width: 100%;  padding: 10px;  margin-bottom: 15px;  box-sizing: border-box; }  .login-container button {  padding: 10px 20px;  background-color: #007BFF;  color: #fff;  border: none;  border-radius: 4px;  cursor: pointer; }  .login-container button:hover {  background-color: #0056b3; }  .error-message {  color: #dc3545;  margin-top: 10px; } |

|  |
| --- |
| **userCabinet.css** |
| body {  font-family: 'Arial', sans-serif;  background-color: #f5f5f5;  color: #333;  margin: 0;  padding: 0; }  .container {  text-align: center;  margin-top: 20px; }  h1 {  color: #007BFF; }  .card-container {  display: flex;  flex-wrap: wrap;  justify-content: space-around; }  .card {  width: 80%; margin: 10px;  display: flex;  flex-wrap: wrap;  justify-content: space-around;  padding: 15px;  border: 1px solid #ddd;  border-radius: 10px;  box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);  background-color: #fff;  transition: transform 0.3s; }  .card:hover {  transform: scale(1.02); }  .card h3 {  color: #007BFF; }  .card p {  margin-bottom: 10px; }  .card span {  font-weight: bold; }  .card button {  padding: 8px 12px;  cursor: pointer;  background-color: #007BFF;  color: #fff;  border: none;  border-radius: 4px; }  .card button:hover {  background-color: #0056b3; }   .edit-button {  background-color: #28a745; }  .delete-button {  background-color: #dc3545; } .create-button {  padding: 8px 12px;  margin: 10px;  cursor: pointer;  background-color: #17a2b8;  color: #fff;  border: none;  border-radius: 4px; }  .create-button:hover {  background-color: #138496; } .modal {  display: none;  position: fixed;  top: 0;  left: 0;  width: 100%;  height: 100%;  background-color: rgba(0, 0, 0, 0.5); }  .modal-content {  background-color: #fff;  margin: 10% auto;  padding: 20px;  border: 1px solid #888;  width: 80%; }  .close {  color: #aaa;  float: right;  font-size: 28px;  font-weight: bold;  cursor: pointer; }  .close:hover, .close:focus {  color: #000;  text-decoration: none;  cursor: pointer; } .modal-content form {  margin-top: 20px;  text-align: left; }  .modal-content form label {  display: block;  margin-bottom: 8px; }  .modal-content form input {  width: 100%;  padding: 8px;  margin-bottom: 16px; }  .modal-content form button {  padding: 10px 20px;  background-color: #007BFF;  color: #fff;  border: none;  border-radius: 4px;  cursor: pointer; }  .modal-content form button:hover {  background-color: #0056b3; }  .modal-content form select {  width: 100%;  padding: 8px;  margin-bottom: 16px;  box-sizing: border-box;  border: 1px solid #ccc;  border-radius: 4px;  background-color: #fff;  font-size: 14px; }  .modal-content form select option {  font-size: 14px; } select {  width: 100%;  padding: 8px;  margin-bottom: 16px;  box-sizing: border-box;  border: 1px solid #ccc;  border-radius: 4px;  background-color: #fff;  font-size: 14px; }  select option {  padding: 8px;  font-size: 14px;  border-bottom: 1px solid #ccc; }  .categories-container {  display: flex;  flex-wrap: wrap;  justify-content: space-around;  padding: 20px; }  .category-card {  flex: 0 0 13%;  box-sizing: border-box;  margin: 10px;  padding: 10px;  border: 1px solid #777777;  border-radius: 8px;  text-align: center;  transition: transform 0.2s ease-in-out; }  .category-card:hover {  cursor: pointer;  transform: scale(1.03); }  .category-image {  max-width: 65%;  border-radius: 50%;  transition: transform 0.2s ease-in-out; }  .category-card:hover .category-image {  transform: scale(1.1); }  .category-title {  margin-top: 10px;  font-weight: bold; }  .category-amount {  margin-top: 5px;  color: #007BFF;  font-weight: bold; } |

|  |
| --- |
| **expense.js** |
| function *openCreateExpenseModal*(categoryName,categoryId) {  document.getElementById('createExpenseModal').style.display = 'block'; }  function *closeCreateExpenseModal*() {  document.getElementById('createExpenseModal').style.display = 'none';  document.getElementById('createExpenseForm').reset(); }  window.onclick = function (event) {  let createModal = document.getElementById('createExpenseModal');  let editModal = document.getElementById('editExpenseModal');  if (event.target === createModal) {  createModal.style.display = 'none';  document.getElementById('createExpenseForm').reset();  }  else if (event.target === editModal) {  editModal.style.display = 'none';  document.getElementById('editExpenseForm').reset();  } };   function *submitExpenseForm*(event) {  event.preventDefault();  let form = document.getElementById('createExpenseForm');  let formData = {  userId: document.getElementById('user\_id').value,  amount: document.getElementById('amount').value,  source: document.getElementById('source').value,  categoryId: document.getElementById('category\_id').value  };  console.log(formData)   *fetch*('/expense', {  method: 'POST',  headers: {  'Content-Type': 'application/json'  },  body: JSON.stringify(formData)  })  .then(response => {  if (!response.ok) {  return response.json().then(error => {  throw new *Error*(error.error);  });  }  return response.text();  })  .then(data => {  document.getElementById('message').innerHTML = 'Запись успешно добавлена! '  *setTimeout*(() => location.reload(),1000)  })  .catch(error => {  console.error('Error:', error.message);  console.log('Server Error:', error.message);  document.getElementById('message').innerHTML = 'Произошла ошибка! ' +error.message  });  }   function *deleteExpense*(expenseId) {  console.log(expenseId)  *fetch*(`/expense?id=${expenseId}`, {  method: 'DELETE',  })  .then(response => response.text())  .then(data => {  location.reload()  })  .catch(error => {  console.error('Error:', error);  }) }  document.getElementById('createExpenseForm').addEventListener('submit', *submitExpenseForm*);  function *openEditExpenseModal*(ExpenseId,amount,source) {  document.getElementById("edit-expense-id").value = ExpenseId;  document.getElementById("edit-amount").value = amount;  document.getElementById("edit-source").value = source;  document.getElementById('editExpenseModal').style.display = 'block'; }  function *closeEditExpenseModal*() {  document.getElementById('editExpenseModal').style.display = 'none';  document.getElementById('editExpenseForm').reset(); }  function *submitEditExpenseForm*(event) {  event.preventDefault();   let form = document.getElementById('createExpenseForm');  let formData = {  id:document.getElementById("edit-expense-id").value,  amount: document.getElementById('edit-amount').value,  source: document.getElementById('edit-source').value,  categoryId: document.getElementById('edit-category-id').value  };  console.log(formData)   *fetch*('/expense', {  method: 'PUT',  headers: {  'Content-Type': 'application/json'  },  body: JSON.stringify(formData)  })  .then(response => {  if (!response.ok) {  return response.json().then(error => {  throw new *Error*(error.error);  });  }  return response.text();  })  .then(data => {  document.getElementById('edit-message').innerHTML = 'Изменения сохранены! '  *setTimeout*(() => location.reload(),1000)  })  .catch(error => {  console.error('Error:', error.message);  console.log('Server Error:', error.message);  document.getElementById('edit-message').innerHTML = 'Произошла ошибка! ' +error.message  }); } document.getElementById('editExpenseForm').addEventListener('submit', *submitEditExpenseForm*); |

|  |
| --- |
| **income.js** |
| function *openCreateIncomeModal*() {  document.getElementById('createIncomeModal').style.display = 'block'; }  function closeCreateIncomeModal() {  document.getElementById('createIncomeModal').style.display = 'none';  document.getElementById('createIncomeForm').reset(); }  window.onclick = function (event) {  let createModal = document.getElementById('createIncomeModal');  let editModal = document.getElementById('editIncomeModal');  if (event.target === createModal) {  createModal.style.display = 'none';  document.getElementById('createIncomeForm').reset();  }  else if (event.target === editModal) {  editModal.style.display = 'none';  document.getElementById('editIncomeForm').reset();  } };   function submitIncomeForm(event) {  event.preventDefault();  let form = document.getElementById('createIncomeForm');  let formData = {  userId: document.getElementById('user\_id').value,  amount: document.getElementById('amount').value,  source: document.getElementById('source').value,  categoryId: document.getElementById('category\_id').value  };  console.log(formData)   fetch('/income', {  method: 'POST',  headers: {  'Content-Type': 'application/json'  },  body: JSON.stringify(formData)  })  .then(response => {  if (!response.ok) {  return response.json().then(error => {  throw new Error(error.error);  });  }  return response.text();  })  .then(data => {  document.getElementById('message').innerHTML = 'Запись успешно добавлена! '  setTimeout(() => location.reload(),1000)  })  .catch(error => {  console.error('Error:', error.message);  console.log('Server Error:', error.message);  document.getElementById('message').innerHTML = 'Произошла ошибка! ' +error.message  });  }   function deleteIncome(incomeId) {  console.log(incomeId)   fetch(`/income?id=${incomeId}`, {  method: 'DELETE',  })  .then(response => response.text())  .then(data => {  location.reload()  })  .catch(error => {  console.error('Error:', error);  }) }  document.getElementById('createIncomeForm').addEventListener('submit', submitIncomeForm);  function openEditIncomeModal(IncomeId,amount,source) {  document.getElementById("edit-income-id").value = IncomeId;  document.getElementById("edit-amount").value = amount;  document.getElementById("edit-source").value = source;  document.getElementById('editIncomeModal').style.display = 'block'; }  function closeEditIncomeModal() {  document.getElementById('editIncomeModal').style.display = 'none';  document.getElementById('editIncomeForm').reset(); }  function submitEditIncomeForm(event) {  event.preventDefault();   let form = document.getElementById('createIncomeForm');  let formData = {  id:document.getElementById("edit-income-id").value,  amount: document.getElementById('edit-amount').value,  source: document.getElementById('edit-source').value,  categoryId: document.getElementById('edit-category-id').value  };  console.log(formData)   fetch('/income', {  method: 'PUT',  headers: {  'Content-Type': 'application/json'  },  body: JSON.stringify(formData)  })  .then(response => {  if (!response.ok) {  return response.json().then(error => {  throw new Error(error.error);  });  }  return response.text();  })  .then(data => {  document.getElementById('edit-message').innerHTML = 'Изменения сохранены! '  setTimeout(() => location.reload(),1000)  })  .catch(error => {  console.error('Error:', error.message);  console.log('Server Error:', error.message);  document.getElementById('edit-message').innerHTML = 'Произошла ошибка! ' +error.message  }); } document.getElementById('editIncomeForm').addEventListener('submit', submitEditIncomeForm); |

|  |
| --- |
| **userCabinet.js** |
| function *openCreateExpenseModal*(categoryName,categoryId) {  console.log(categoryName)  document.getElementById('modalTitle').innerText = "Новая покупка в категории " + categoryName;  document.getElementById("category\_id").value = categoryId;  document.getElementById('createExpenseModal').style.display = 'block'; }  function *closeCreateExpenseModal*() {  document.getElementById('createExpenseModal').style.display = 'none';  document.getElementById('createExpenseForm').reset(); }  window.onclick = function (event) {  let createModal = document.getElementById('createExpenseModal');  let editModal = document.getElementById('editExpenseModal');  if (event.target === createModal) {  createModal.style.display = 'none';  document.getElementById('createExpenseForm').reset();  }  else if (event.target === editModal) {  editModal.style.display = 'none';  document.getElementById('editExpenseForm').reset();  } };   function submitExpenseForm(event) {  event.preventDefault();  let form = document.getElementById('createExpenseForm');  let formData = {  userId: document.getElementById('user\_id').value,  amount: document.getElementById('amount').value,  source: document.getElementById('source').value,  categoryId: document.getElementById('category\_id').value  };  console.log(formData)   fetch('/expense', {  method: 'POST',  headers: {  'Content-Type': 'application/json'  },  body: JSON.stringify(formData)  })  .then(response => {  if (!response.ok) {  return response.json().then(error => {  throw new Error(error.error);  });  }  return response.text();  })  .then(data => {  document.getElementById('message').innerHTML = 'Запись успешно добавлена! '  setTimeout(() => location.reload(),1000)  })  .catch(error => {  console.error('Error:', error.message);  console.log('Server Error:', error.message);  document.getElementById('message').innerHTML = 'Произошла ошибка! ' +error.message  });  }   function deleteExpense(expenseId) {  console.log(expenseId)   fetch(`/expense?id=${expenseId}`, {  method: 'DELETE',  })  .then(response => response.text())  .then(data => {  location.reload()  })  .catch(error => {  console.error('Error:', error);  }) }  document.getElementById('createExpenseForm').addEventListener('submit', submitExpenseForm);  function openEditExpenseModal(expenseId,amount,source) {  document.getElementById("edit-expense-id").value = expenseId;  document.getElementById("edit-amount").value = amount;  document.getElementById("edit-source").value = source;  document.getElementById('editExpenseModal').style.display = 'block'; }  function closeEditExpenseModal() {  document.getElementById('editExpenseModal').style.display = 'none';  document.getElementById('editExpenseForm').reset(); }  function submitEditExpenseForm(event) {  event.preventDefault();   let form = document.getElementById('createExpenseForm');  let formData = {  userId: document.getElementById('user\_id').value,  id:document.getElementById("edit-expense-id").value,  amount: document.getElementById('edit-amount').value,  source: document.getElementById('edit-source').value,  categoryId: document.getElementById('edit-category-id').value  };  console.log(formData)   fetch('/expense', {  method: 'PUT',  headers: {  'Content-Type': 'application/json'  },  body: JSON.stringify(formData)  })  .then(response => {  if (!response.ok) {  return response.json().then(error => {  throw new Error(error.error);  });  }  return response.text();  })  .then(data => {  document.getElementById('edit-message').innerHTML = 'Изменения сохранены! '  setTimeout(() => location.reload(),1000)  })  .catch(error => {  console.error('Error:', error.message);  console.log('Server Error:', error.message);  document.getElementById('edit-message').innerHTML = 'Произошла ошибка! ' +error.message  }); } document.getElementById('editExpenseForm').addEventListener('submit', submitEditExpenseForm); |

## **ПРИЛОЖЕНИЕ В. Файлы конфигурации**

|  |
| --- |
| **pom.xml** |
| *<?*xml version="1.0" encoding="UTF-8"*?>* <project xmlns="http://maven.apache.org/POM/4.0.0"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">  <modelVersion>4.0.0</modelVersion>   <groupId>com.example</groupId>  <artifactId>ilnitskiy\_9</artifactId>  <version>1.0-SNAPSHOT</version>  <name>kursovaya</name>  <packaging>war</packaging>   <properties>  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  <maven.compiler.target>17</maven.compiler.target>  <maven.compiler.source>17</maven.compiler.source>  </properties>   <dependencies>  <dependency>  <groupId>javax.servlet</groupId>  <artifactId>javax.servlet-api</artifactId>  <version>4.0.1</version>  <scope>provided</scope>  </dependency>  <dependency>  <groupId>javax.servlet</groupId>  <artifactId>jsp-api</artifactId>  <version>2.0</version>  <scope>provided</scope>  </dependency>  <dependency>  <groupId>jstl</groupId>  <artifactId>jstl</artifactId>  <version>1.2</version>  </dependency>  <dependency>  <groupId>org.postgresql</groupId>  <artifactId>postgresql</artifactId>  <version>42.5.1</version>  </dependency>   <dependency>  <groupId>com.google.code.gson</groupId>  <artifactId>gson</artifactId>  <version>2.10.1</version>  </dependency>  </dependencies>   <build>  <plugins>  <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-war-plugin</artifactId>  <version>3.3.2</version>  </plugin>  </plugins>  </build> </project> |

|  |
| --- |
| **web.xml** |
| *<?*xml version="1.0" encoding="UTF-8"*?>* <web-app  xmlns="http://java.sun.com/xml/ns/javaee"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd"  version="3.0">  <filter>  <filter-name>setCharacterEncodingFilter</filter-name>  <filter-class>org.apache.catalina.filters.SetCharacterEncodingFilter</filter-class>  <init-param>  <param-name>encoding</param-name>  <param-value>UTF-8</param-value>  </init-param>  </filter>   *<!-- The mapping for the Set Character Encoding Filter -->* <filter-mapping>  <filter-name>setCharacterEncodingFilter</filter-name>  <url-pattern>/\*</url-pattern>  </filter-mapping> </web-app> |

|  |
| --- |
| **script.sql** |
| CREATE TABLE users (  id SERIAL PRIMARY KEY ,  name VARCHAR(50) NOT NULL,  login VARCHAR(50) NOT NULL,  password VARCHAR(100) NOT NULL ); CREATE TABLE income\_category(  id SERIAL PRIMARY KEY,  name VARCHAR(50) NOT NULL ); CREATE TABLE expense\_category(  id SERIAL PRIMARY KEY,  name VARCHAR(50) NOT NULL ); CREATE TABLE income (  id SERIAL PRIMARY KEY,  user\_id INTEGER REFERENCES users(id),  amount INTEGER NOT NULL,  source VARCHAR(50) NOT NULL,  category INTEGER REFERENCES income\_category(id),  timestamp TIMESTAMP NOT NULL ); CREATE TABLE expense (  id SERIAL PRIMARY KEY,  user\_id INTEGER REFERENCES users(id),  amount INTEGER NOT NULL,  source VARCHAR(50) NOT NULL,  category INTEGER REFERENCES expense\_category(id),  timestamp TIMESTAMP NOT NULL ); |