МИНОБРНАУКИ РОССИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«Санкт-Петербургский политехнический университет Петра Великого»

Институт компьютерных наук и технологий

Кафедра «Распределенные вычисления и компьютерные сети»

К У Р С О В А Я Р А Б О Т А

Система учета прохода в здание

по дисциплине

«Программное обеспечение распределенных вычислительных систем»

Выполнил
студент гр. 63507/1 Е.В. Григуть

Руководитель
доцент, к.т.н. И.В. Стручков

Санкт-Петербург

2016

Оглавление

Анализ задания

Требуется создать корпоративное распределенное web-приложение с помощью технологии Enterprise Java Beans, а конкретно: систему учета прохода в здание: На сервере имеется файл со списком разрешенных номеров смарт-карт. Также реализовать операции: проверка допуска в здание, запрос количества людей в здании и присутствие конкретного человека. Сериализуемый объект – смарт-карта (имя, номер).

Варианты использования

При ознакомлении с предложенным заданием были выделены следующие возможные варианты использования.

Проверка доступа в здание

1. Некое физическое устройство считывает данные приложенной карты
2. Система пропуска проверяет наличие карты с данным номером в базе турникета
3. Система пропуска узнает, находится ли данный человек в здании
4. A) Человек в здании, приложил карту - значит он уже уходит
   B) Человек не в здании, приложил карту - значит он только пришел
   C) Человека нет в базе
5. A) Турникет выпускает
   B) Турникет впускает
   C) Турникет не дает доступ в задние

Подсчет количества людей в здании

1. Система пропуска получает запрос о количестве человек в здании
2. Система пропуска возвращает количество людей в здании

Присутствие конкретного человека в здании

1. Система пропуска получает запрос в виде фамилии и отчества некоторого человека
2. Система пропуска проверяет наличие полученных данных в базе
3. A) Человек есть в базе, то узнает, находится ли человек в здании
   B) Человека нет в базе, то сообщает об этом
4. A) Человек находится в здании, то система пропуска сообщает о положительном ответе
   B) Человека нет в здании, система пропуска сообщает об отрицательном ответе

Модель предметной области

Рис. 1: Диаграмма классов предметной области

Реализация задания с помощью технологии «EJB»

Объектно-ориентированное проектирование

Диаграмма классов

Класс TurnStile – сеансовый компонент EJB, реализующий локальный и удалённый интерфейс. Имеет методы checkPerson(Id), который позволяет проверить, есть ли доступ человеку с данной картой в здание, countPeople() – позволяет узнать количество человек в здании в данный момент, findPerson(firstName, lastName) – позволяет узнать, есть ли конкретный человек в здании в данный момент.

EntityManager – диспетчер сущностей, аннотация @PersistenceContext(unitName = "TurnStile-ejbPU") определяет поставщика и базу данных, инициализируется как EntityManager em. Используется его методы createNamedQuery(name) и persist(card).

Класс Card – сущностный компонент из базы данных с уникальным полем Id и полями для дополнительной информации – имя (firstname), фамилия (lastName), присутствие в здании(isHere).

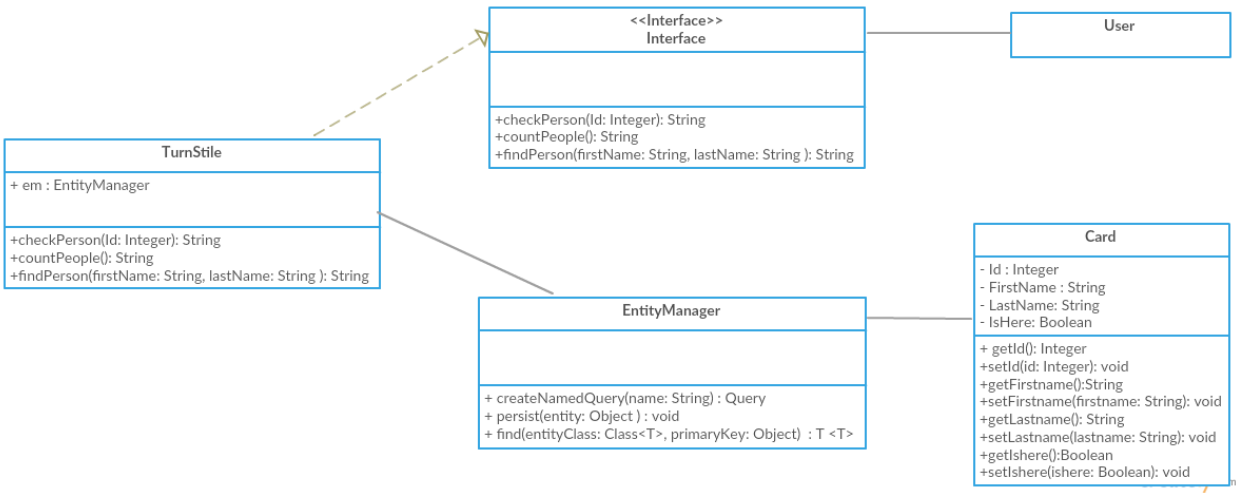

Рис. 2: Диаграмма классов системы пропуска в здание

Диаграмма последовательностей - Проверка доступа в здание

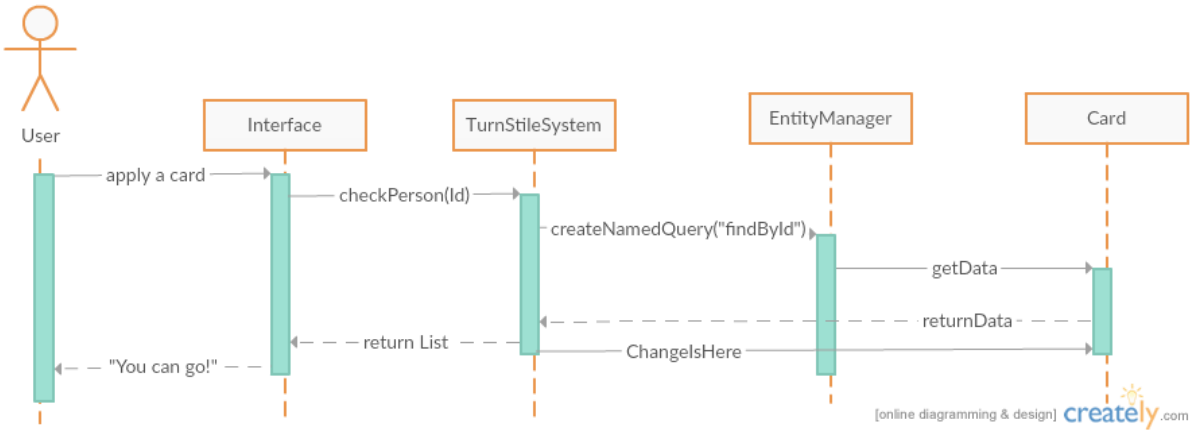

Рис. 3: Диаграмма классов системы пропуска в здание – проверка доступа в здание

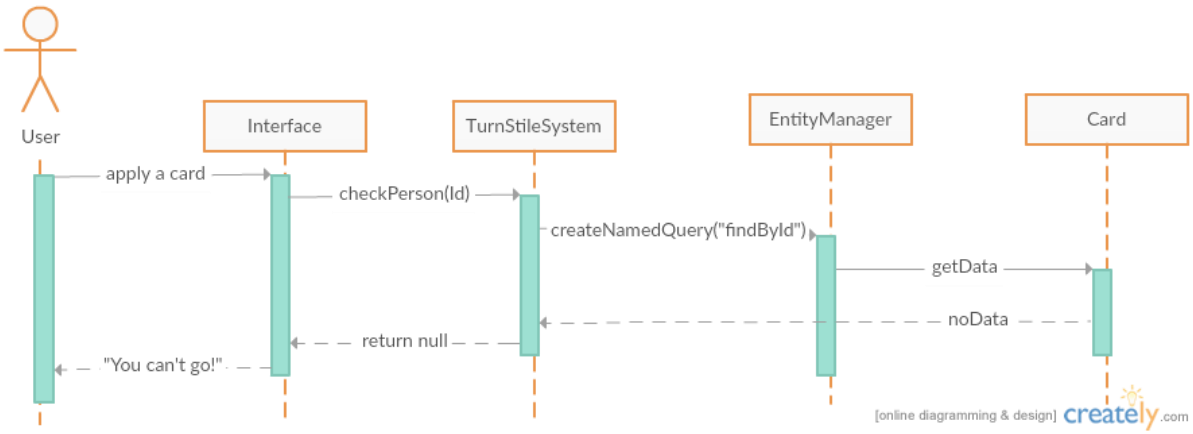

Рис. 4: Диаграмма классов системы пропуска в здание – проверка доступа в здание, альтернативный вариант

Диаграмма последовательностей - Присутствие конкретного человека в здании

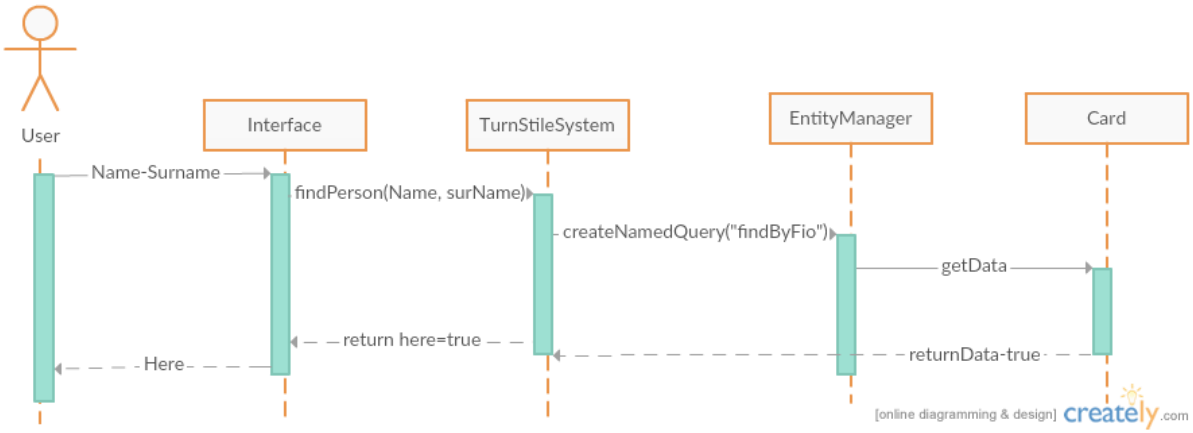

Рис. 5: Диаграмма классов системы пропуска в здание – присутствие конкретного человека в здании

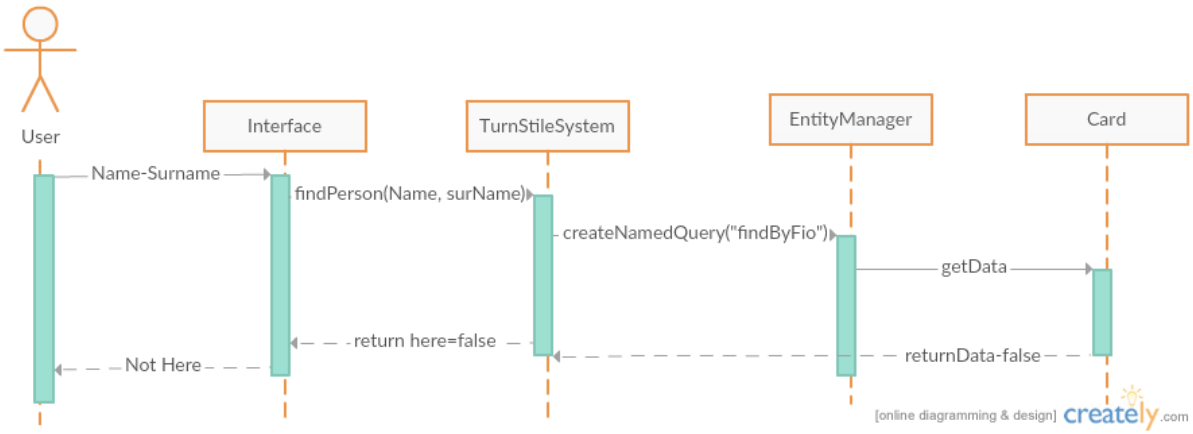

Рис. 6: Диаграмма классов системы пропуска в здание – присутствие конкретного человека в здании, альтернативный вариант 1

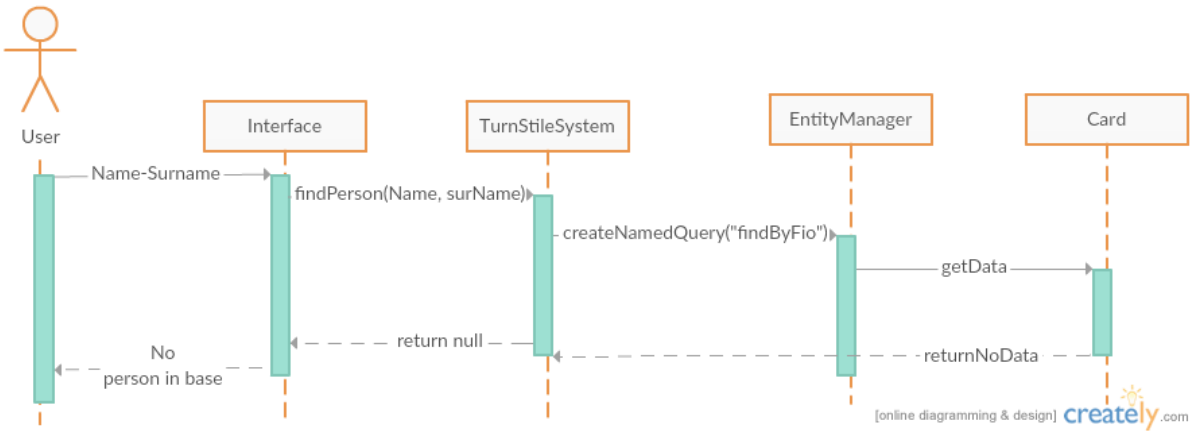

Рис. 7: Диаграмма классов системы пропуска в здание – присутствие конкретного человека в здании, альтернативный вариант 2

Диаграмма последовательностей - Подсчет количества людей в здании

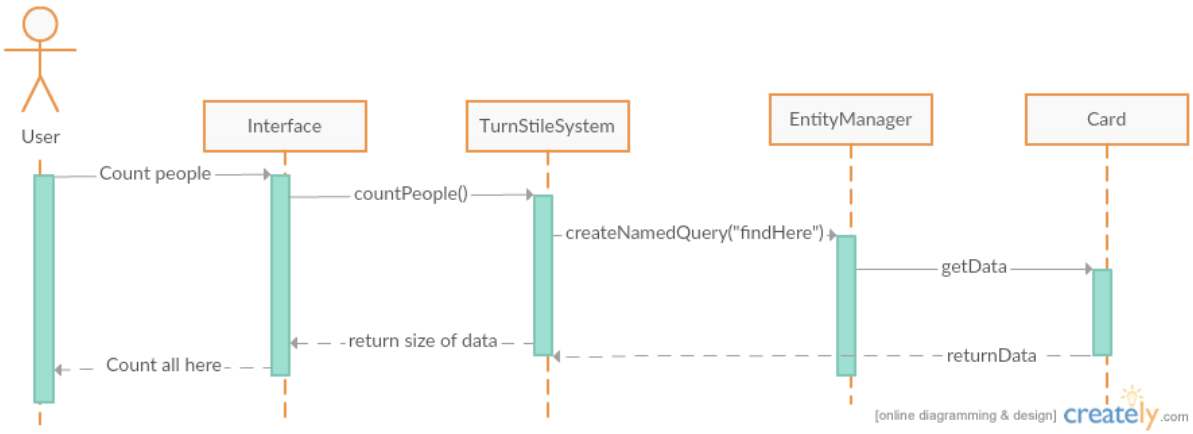

Рис. 8: Диаграмма классов системы пропуска в здание – Подсчет количества людей в здании

Описание программы

Была применена технология Enterprise Java Beans, технология хранения данных persistence Java EE 7 для размещения данных приложения в СУБД JavaDB с помощью сервера GlassFish.

В соответствии с моделью предметной области была выделена сущность – Turnstile, для которой автоматически был сгенерирован класс, соответствующий таблице с полями Id, firstName, lastName, isHere, созданной ранее в СУБД .

Сеансовый компонент TurnStileSessionBean помечен аннотацией @Stateless (без сохранения состояния), в нем содержится менеджер сущностей, позволяющий работать с классами сущностей с помощью сгенерированных именных запросов (@NamedQueries). В классе TurnStileSessionBean реализованы требуемые методы.

Представлением системы пропуска в здание являются веб-формы, созданные по технологии Java Server Pages .

- index.jsp – стартовая страница.
- error.jsp – страница вывода возникших ошибок.
- check.jsp – содержит форму ввода данных для проверки доступа в здание и передает управление checkResult.jsp.
- checkResult.jsp – использует метод checkPerson(id) из класса TurnStileSessionBean для допуска или не допуска человека в здание.
- count.jsp - возвращает количество человек в здании.
- find.jsp – содержит форму ввода данных для проверки наличия в здание конкретного человека и передает управление findResult.jsp.
- findResult.jsp – использует метод findPerson(firstname, lastname) из класса TurnStileSessionBean для определения местонахождения человека.

Текст программы

Исходный код TurnStile-ejb

TurnStileSessionBean.java

```
package turnstile;

import java.util.List;
import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import javax.persistence.Query;

@Stateless
public class TurnStileSessionBean implements TurnStileSessionBeanRemote,
TurnStileSessionBeanLocal {

    @PersistenceContext(unitName = "TurnStile-ejbPU")
    private EntityManager em;
```

```
//Метод подсчета количества человек в здании
    @Override
    public String countPeople() {
        //создание именного запроса с параметром IsHere = true
        Query query =
        em.createNamedQuery("Card.findAllHere").setParameter("param", true);
        //получение данных после выполнения запроса
        List card = query.getResultList();
        Дсчет количества полученных данных, которое и является ответом
        String str = String.valueOf(card.size());
        return str;
    }

//Метод проверки доступа для конкретного человека
    @Override
    public String checkPerson(Integer Id) {
        //создание именного запроса с параметром Id
        Query query = em.createNamedQuery("Card.findById");
        query.setParameter("id", Id);
        //получение данных после выполнения запроса
        List card = query.getResultList();
        //если результат не null, то меняем значение поля присутствия в здании
        if (card.size() > 0) {
            boolean isHere;
            String str;
            isHere = ((Card) card.get(0)).getIshere();
            str = ((Card) card.get(0)).getFirstname() + " " + ((Card)
card.get(0)).getLastname();
            if (isHere == false) {
                str = "HELLO, " + str;
            } else {
                str = "GOODBYE, " + str;
            }
            ((Card) card.get(0)).setIshere(!isHere);

            return str + "!";
        }
        //иначе проход в здание невозможен
        else {
            return "You can't go! Sorry...";
        }
    }

//метод для определения присутствия конкретного человека в здании
    @Override
    public String findPerson(String firstName, String lastName) {
        //создание именного запроса с параметрами firstName и lastName
        Query query = em.createNamedQuery("Card.findByFullname");
        query.setParameter("firstname", firstName);
        query.setParameter("lastname", lastName);
        //получение данных после выполнения запроса
        List card = query.getResultList();
        //если результат не null, то по значению поля присутствия
        //определяем   находится ли человек в здании
        if (card.size() > 0) {
            boolean here = ((Card) card.get(0)).getIshere();
            if (here == false) {
                return "NOT HERE";
            } else {
                return "HERE";
            }
        }
        //иначе сообщаем, что такого человека в базе нет
        else{
```

```
            return "No such person in base";
        }
    }

    public void persist(Object object) {
        //em.persist(object);
    }
}
```

TurnStileSessionBeanLocal.java

```
package turnstile;

import javax.ejb.Local;

@Local
public interface TurnStileSessionBeanLocal {
}
```

TurnStileSessionBeanRemote.java

```
package turnstile;

import javax.ejb.Remote;

@Remote
public interface TurnStileSessionBeanRemote {
    public String checkPerson(Integer Id);
    public String countPeople();
    public String findPerson(String firstName, String lastName);
}
```

Card.java

```
package turnstile;

import java.io.Serializable;
import javax.persistence.Basic;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.Table;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;
import javax.xml.bind.annotation.XmlRootElement;

@Entity
@Table(name = "CARD")
@XmlRootElement
@NamedQueries({
    @NamedQuery(name = "Card.findAll", query = "SELECT c FROM Card c")
    , @NamedQuery(name = "Card.findAllHere", query = "SELECT c FROM Card c
WHERE c.ishere = :param")
    , @NamedQuery(name = "Card.findById", query = "SELECT c FROM Card c WHERE
c.id = :id")
    , @NamedQuery(name = "Card.findByFirstname", query = "SELECT c FROM Card
c WHERE c.firstname = :firstname")
    , @NamedQuery(name = "Card.findByLastname", query = "SELECT c FROM Card c
WHERE c.lastname = :lastname")
    , @NamedQuery(name = "Card.findByFullname", query = "SELECT c FROM Card c
WHERE c.firstname = :firstname AND c.lastname = :lastname")
    , @NamedQuery(name = "Card.findByIshere", query = "SELECT c FROM Card c
WHERE c.ishere = :ishere")})
```

```
public class Card implements Serializable {
    private static final long serialVersionUID = 1L;
    @Id
    @Basic(optional = false)
    @NotNull
    @Column(name = "ID")
    @GeneratedValue
    private Integer id;
    @Size(max = 30)
    @Column(name = "FIRSTNAME")
    private String firstname;
    @Size(max = 30)
    @Column(name = "LASTNAME")
    private String lastname;
    @Basic(optional = false)
    @Column(name = "ISHERE")
    private Boolean ishere;

    public Card() {
    }

    public Card(Integer id) {
        this.id = id;
    }

    public Card(Integer id, Boolean ishere) {
        this.id = id;
        this.ishere = ishere;
    }

    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public String getFirstname() {
        return firstname;
    }

    public void setFirstname(String firstname) {
        this.firstname = firstname;
    }

    public String getLastname() {
        return lastname;
    }

    public void setLastname(String lastname) {
        this.lastname = lastname;
    }

    public Boolean getIshere() {
        return ishere;
    }

    public void setIshere(Boolean ishere) {
        this.ishere = ishere;
    }
```

```
    @Override
    public int hashCode() {
        int hash = 0;
        hash += (id != null ? id.hashCode() : 0);
        return hash;
    }

    @Override
    public boolean equals(Object object) {
        if (!(object instanceof Card)) {
            return false;
        }
        Card other = (Card) object;
        if ((this.id == null && other.id != null) || (this.id != null &&
!this.id.equals(other.id))) {
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
        return "turnstile.Card[ id=" + id + " ]";
    }

}
```

Исходный код TurnStile-war

index.jsp

```
<%@page contentType="text/html" pageEncoding="windows-1251"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=windows-
1251">
        <title>TurnStile Application</title>
    </head>
    <body>
        <h1>Welcome to TurnStile</h1>
       <p><a href="check.jsp">Check: GO or STOP</a></p>
       <p><a href="count.jsp">Count all here people</a></p>
       <p><a href="find.jsp">Find where is person</a></p>
    </body>
</html>
```

error.jsp

```
<%@page contentType="text/html" pageEncoding="windows-1251"%>
<!DOCTYPE HTML>
<%@ page isErrorPage="true" %>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=windows-
1251">
        <title>TurnStile Application</title>
    </head>
    <body>
        <h1>There was an error</h1>
        <p style="color: red">${pageContext.errorData.throwable.message}</p>
        <p><a href="index.jsp">Return</a></p>
    </body>
</html>
```

check.jsp

```
<%@page contentType="text/html" pageEncoding="windows-1251"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=windows-
1251">
        <title>TurnStile Application</title>
    </head>
    <body>
        <h1>Check is here</h1>
        <form action="checkResult.jsp" method="POST">
            <p>Number: <input type="text" name="id" value="" /></p>
            <input type="submit" value="Search" />
        </form>
    </body>
</html>
```

checkResult.jsp

```
<%@page contentType="text/html" pageEncoding="windows-1251"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=windows-
1251">
        <title>TurnStile Application</title>
    </head>
    <body>

        <%@page import="javax.naming.*, turnstile.*" %>
        <%@ page errorPage="error.jsp"%>
        <%!
            TurnStileSessionBeanRemote ejbRef;
        %>
        <%
            InitialContext ic = new InitialContext();
            ejbRef
                    = (TurnStileSessionBeanRemote)
ic.lookup("turnstile.TurnStileSessionBeanRemote");
        %>
        <h1>Decision: </h1>
        <p>
<%=ejbRef.checkPerson(Integer.valueOf(request.getParameter("id")).intValue())
%></p>
        <p><a href="index.jsp">Return</a></p>
    </body>
</html>
```

find.jsp

```
<%@page contentType="text/html" pageEncoding="windows-1251"%>
<!DOCTYPE HTML>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=windows-
1251">
        <title>TurnStile Application</title>
    </head>
    <body>
        <h1>Find where is person</h1>
        <form action="findResult.jsp" method="POST">
            <p>First Name: <input type="text" name="firstName" value=""
/></p>
            <p>Last Name: <input type="text" name="lastName" value="" /></p>
            <input type="submit" value="Search" />
        </form>
```

```
    </body>
</html>
```

findResult.jsp

```
<%@page contentType="text/html" pageEncoding="windows-1251"%>
<!DOCTYPE HTML>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=windows-
1251">
        <title>TurnStile Application</title>
    </head>
    <body>
        <%@page import="javax.naming.*, turnstile.*" %>
        <%@ page errorPage="error.jsp"%>
        <%!
            TurnStileSessionBeanRemote ejbRef;
        %>
        <%
            InitialContext ic = new InitialContext();
            ejbRef = (TurnStileSessionBeanRemote)
ic.lookup("turnstile.TurnStileSessionBeanRemote");
        %>
        <h1>Search Results:</h1>
        <p> <%=request.getParameter("firstName")%>
<%=request.getParameter("lastName")%> is
            <%=ejbRef.findPerson(request.getParameter("firstName"),
request.getParameter("lastName"))%> !</p>
        <p><a href="index.jsp">Return</a></p>
    </body>
</html>
```

count.jsp

```
<%@page contentType="text/html" pageEncoding="windows-1251"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=windows-
1251">
        <title>TurnStile Application</title>
    </head>
    <body>

        <%@page import="javax.naming.*, turnstile.*" %>
        <%@ page errorPage="error.jsp"%>
        <%!
            TurnStileSessionBeanRemote ejbRef;
        %>
        <%
            InitialContext ic = new InitialContext();
            ejbRef
                    = (TurnStileSessionBeanRemote)
ic.lookup("turnstile.TurnStileSessionBeanRemote");
        %>
        <h1>Now:</h1>
        <p> <%=ejbRef.countPeople()%> people in office. </p>
        <p><a href="index.jsp">Return</a></p>
    </body>
</html>
```

Методика и результаты тестирования

Вариант тестирования	Ожидаемый результат	Фактический результат
Проверка доступа в здание • Нажимаем «Check: GO or STOP». • Передаем турникету данные своей карты (номер), которые есть в базе • Нажимаем «Submit»	Турникет должен пропустить.	Турникет пропускает.
Проверка доступа в здание • Нажимаем «Check: GO or STOP». • Передаем турникету данные своей карты (номер), которые нет в базе • Нажимаем «Submit»	Турникет не должен пропустить.	Турникет не пропускает.
Подсчет количества человек в здании • Нажимаем «Count all here people».	Система должна вернуть количество человек в здании.	Система вернула количество человек в здании.
Проверка нахождения конкретного человека в здании • Нажимаем «Find where is person». • Вводим имя и фамилию человека, присутствующего в базе • Нажимаем «Submit».	Система должна сообщить нахождение человека (HERE/ NOT NERE)	Система сообщает нахождение человека (HERE/ NOT NERE)
Проверка нахождения конкретного человека в здании • Нажимаем «Find where is person». • Вводим имя и фамилию человека, отсутствующего в базе • Нажимаем «Submit».	Система должна сообщить об отсутствие этого человека в базе	Система сообщает об отсутствие этого человека в базе

Инструкция системному администратору

Рекомендуется следующий набор программных средств:

1. JDK 8u111 with NetBeans 8.2 (СУБД JavaDB входит в состав пакета) (http://www.oracle.com/technetwork/java/javase/downloads/index.html)
2. Сервер приложений GlassFish 4.1 (После установки п.1 есть возможность установить через подключаемые модули)

Для работы с базой нужно указать установочный каталог JavaDB и папку, в которой хранится база данных (\Turnstile), запустить сервер СУБД на вкладке Сервисы , сустановить связь с базой данных (логин: app, пароль: app; URL-адрес базы данных jdbc:derby://localhost:1527/turnstile [app на APP]), создать схему БД и таблицу ЕгктЫешду в сответствии с диаграммой классов.

Проверить через консоль сервера наличие пулов соединений: turnstilePool (PortNumber: 1527, DatabaseName: turnstile, User: app, Password: app, ServerName: localhost). Ресурсы JDBC: jdbc/turnstile (Имя пула: turnstilePool). В случае отсутствия, создать.

Затем, открыть проект TurnStile с помощью NetBeans, правой кнопкой мыши нажав по проекту, выбрать «Собрать», после успешной сборки – «Выполнить» для запуска приложения в браузере.

Инструкция пользователю

Приложения может быть запущено на выполнение в браузере: http://localhost:8080/Turnstile-war/

Начальная страница предлагает выбрать одно из трех возможных действий:

- проверка допуска в здание,
- запрос количества людей в здании
- присутствие конкретного человека

Поля для заполнения интуитивно понятны:

- FirstName – Имя конкретного человека
- LastName – Фамилия конкретного человека
- Number – Номер, идентифицирующий карту

Для подтверждения своих действий следует нажать кнопку «Submit», для возвращения на начальную страницу «Return».

Выводы

Задание выполнено согласно поставленным требованиям. Реализовано приложение для системы доступа в здание, которое позволяет выполнять следующие действия:

- проверка допуска в здание,
- запрос количества людей в здании
- присутствие конкретного человека

Данный проект может быть дополнен путем расширения функциональных возможностей, доступных для пользователя, усложнения базы данных и улучшения пользовательского интерфейса.

В ходе выполнения работы были получены знания в следующих областях:

- Разработка web-приложений с помощью языка Java
- Изучение компонента системы EJB, который понадобился для создания, развертывания и функционирования распределенного приложения Java масштаба предприятия
- Исследование технологии хранения данных Java EE 7 для размещения данных в СУБД