СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 7](#_Toc45868022)

[Характеристика организации 10](#_Toc45868023)

[Актуальность работы, выполненной на предприятии 11](#_Toc45868024)

[Цель и задачи практики 13](#_Toc45868025)

Описание структурного подразделения организации…………………………14

[Описание работы, выполненной студентом за период прохождения практики 15](#_Toc45868026)

[ЗАКЛЮЧЕНИЕ 21](#_Toc45868027)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 22](#_Toc45868028)

[ПРИЛОЖЕНИЕ А 23](#_Toc45868029)

**ВВЕДЕНИЕ**

В настоящее время важную роль занимает контроль знаний учащихся школ, сотрудников предприятий, студентов вузов. Под контролем понимается система позволяющая измерить уровень знаний, умений и навыков и компетенций.

Существует большое количество форм контроля - экзамен, зачет, аттестация, контрольная, тестирование. На данный момент наиболее эффективной формой контроля знаний является тестирование. Тестирование – это комплекс, специальным образом подготовленных и подобранных заданий, позволяющий определить уровень знаний, умений и навыков испытуемого. Тестирование имеет ряд преимуществ, одно из них заключается в том, что оно позволяют опросить всех испытуемых в одинаковых условиях, применяя при этом шкалу оценок заранее разработанную специалистом, который проводит тестирование, что так же исключает предвзятое отношение к испытуемому со стороны человека проводящего тестирование. Таким образом намного повышается объективность, и обоснованность результата допустим с экзаменом.

Тестирование получило свое распространение практически во всех сферах человеческой жизни, а именно там, где нужно в краткие сроки, а главное точно определить знания по всему курсу обучения, где большое количество испытуемых, а так же необходимо отобрать лучших участников: тестирование в ГИБДД, аттестация сотрудников предприятий, при устройстве на работу, аттестация и контроль знаний школьников, студентов.

В настоящее время, сложно представить человека без опыта работы за компьютером, в эпоху глобальной компьютеризации, развития технологии современных компьютеров и передачи данных все более актуальными становятся разнообразные компьютерные автоматизированные системы тестирования, которые имеют ряд преимуществ перед традиционными методами проверки знаний.

Важным преимуществом компьютерных систем тестирования, является удобство проведения теста, стало намного удобнее проводить тестирования во всех сферах, где ранее применялись традиционные тесты. Например, дистанционное образование, которое становится все востребований.

Стоит также отметить остальные преимущества компьютерного тестирование. Оно отличается большой оперативностью, производительностью процесса тестирования и объективностью результатов контроля знаний – человек проводящий тестирование может провести опрос большего числа испытуемых за меньшее время по сравнению с обычным тестированием, и позволяет дать более качественный анализ подготовки испытуемых по большому набору различных вопросов. Важным фактором является адаптация содержания и сложности тестовых вопросов уровню знаний обучаемых, возможность самоконтроля.

Важным понятием является автоматизация тестирования. Автоматизация тестирования — это набор практик, вопросов, подходов, методов, которые позволяют исключить проверку результатов человеком. В данном виде тестирование результаты проверят программа [1].

Рассмотрим основные особенности и требования к процессу автоматизированного тестирования.

Общие требования к тестам:

* валидность;
* определенность;
* однозначность;
* надежность.

Требование валидности теста – это показатель эффективности теста, а также на сколько тест соответствует содержанию теоретического и практического материала в нем.

Требование определенности теста означает, что при его прочтении испытуемый хорошо понимает, какие задания и в каком объеме ему необходимо выполнить, для получения правильного решения поставленной задачи.

Требование однозначности теста означает, что тест должен иметь эталон, составленный различными экспертами.

Требование надежности теста заключается в обеспечении устойчивого результата многократного тестирования одного и того же испытуемого [2].

**Характеристика предприятия**

ДГТУ, отдел системного администрирования. Отдел решает задачи с администрированием общесетевых системных сервисов, обеспечение доступа к ним и техническую поддержку пользователей. Сотрудники отдела обеспечивают непрерывную эксплуатацию серверного и сетевого оборудования, бесперебойную работу корпоративной сети ДГТУ.

Основными функциями отдела являются

1. Администрирование техническое обслуживание серверных программно-аппаратных средств корпоративной сети ДГТУ;
2. Обеспечение бесперебойного функционирования серверного и коммутационного оборудования университета более 20 серверов и 40 управляемых коммутаторов;
3. Обеспечение работы единой информационной среды головного ВУЗа удаленных 5 удаленных подразделений и филиалов;
4. Развитие, обеспечение функционирования и системное сопровождение общесетевых сервисов корпоративной компьютерной сети университета ДГТУ, таких как подсистема доступа в Интернет, подсистема электронной почты, общее файловое хранилище, система консультант+ и другие.
5. Поддержка пользователей по вопросам доступности общесетевых сервисов университета таких как доступ в Интернет, электронная почта, файловые сервера и т.д;
6. Обеспечение целостности данных информационных систем университета реализация процедур обслуживания систем хранения данных и их резервного копирования порядка 5 Тб данных;
7. Обеспечение реализации политик лицензирования программных продуктов университета, контроль лицензий, используемых в учебном процессе и сотрудниками ВУЗа;
8. Обеспечение работы системы антивирусной и антиспам защиты в корпоративной сети ДГТУ на более чем 1200 рабочих местах.
9. Определение и реализация политик безопасного доступа подразделений и отдельных пользователей к внешним и внутренним информационным ресурсам корпоративной сети;
10. Администрирование учетных записей пользователей информационно-вычислительной системы университета; .
11. Администрирование программно-технических средств серверов обеспечения учебного процесса, обеспечение централизованного управления и бесперебойной работы 12 компьютерными классами [3].

**Актуальность работы, выполненной на предприятии**

Важно понимать необходимость тестирования сотрудников. Она заключается в следующем:

1. Для нахождение нового сотрудника на нужную должность. Данное тестирование проводят сотрудники отдела кадров после собеседования.
2. Для проверки уровня навыков, умений и знаний уже работающего сотрудника – это даст понимание соответствует ли он данной должности.
3. Для оценки психологичного состояния сотрудника.
4. Для оценки творческих способностей сотрудника [4].

Рассмотрим наиболее эффективные методики тестирования:

1. Тесты на профпригодность:

* Аттестация. Смысл, данный методики прост : сотруднику задаются вопросы, относящиеся к его профессиональной деятельности
* Испытание. Относиться больше к экзамену, когда необходимо испытать умение быстро и качественно исполнять свои должностные обязанности.
* Тесты для повышения квалификации, применяется при повышении сотрудника по карьерной лестнице. Сложность теста зависит от должности сотрудника в данный момент.
* Тесты-опросники, помогают лучше понять желание сотрудника и когда может наступить “выгорание” работника [4].

2 Тесты опросники для изучения психологического состояния:

* Диагностика самоэффективности Маддукса и Шеера.
* Диагностика агрессии Басса-Дарки.
* Тест Гилфорда на измерение социального интеллекта.
* Определение мотивации к успеху Т. Элерса.

3 Проективные тесты – это тесты без вопросов, в данных тестах испытуемому необходимо использовать фантазию и нарисовать что его попросят:

* “Дом, дерево, человек”, человеку необходимо использовать фантазию и нарисовать один рисунок с использованием данных объектов.
* “Нарисуй свою работу”, испытуемому предлагается нарисовать свою работу и постараться передать кем он себя чувствует в коллективе.
* Тест Роршаха.
* Цветной тест Рюшера [4].

Для успешного развития предприятия тестирование персонала является неотъемлемой частью. Тестирование позволяет определить на начальных этапах развития проблем у сотрудника, а также во всем коллективе. При тестировании персонала опытный руководитель использует не одну методику, а комбинирует несколько, что дает большую эффективность.

Преимуществом компьютерных автоматизированных систем является то, что система позволяет получить корректную оценку уровня знаний, умений и навыков сотрудников, уменьшить время на получение результатов [5].

**Цель и задачи работы**

Основные цели внедрения автоматизированной системы тестирования:

* Оперативный контроль работы.
* Сокращение лишнего бумажного документооборота.
* Быстрое реагирование и исправление недоработок.
* Установление единых критериев оценивания деятельности (КРI).
* Упрощение системы перераспределения заданий.

Цель внедрения автоматизированной программы оценки персонала – создание простой и открытой системы тестирования, с высокой производительностью.

Цель данной практики реализовать веб-приложение по тематике НИР, а также получить навыки по разработке Web-приложений с применением платформы Java, Spring Framework и паттерна проектирования MVC, ознакомиться с принципами работы Spring и шаблонизатора Thymeleaf.

Задачи практики:

|  |
| --- |
| 1. Анализ актуальности темы. |
| 1. Анализ предметной области. |
| 1. Поиск научной литературы. |
| 1. Изучение теоретического материала. 2. Разработка программы. 3. Тестирование программы. 4. Формирование отчета. |

Требования к обеспечению надежного функционирования программы: Надежное функционирование программы должно быть обеспечено выполнением пользователем совокупности организационно-технических мероприятий, перечень которых приведен ниже:

1. организацией бесперебойного питания технических средств;
2. организацией правильного использования модулей;
3. организацией правильного ввода параметров инициализирующих структур;
4. регулярным выполнением требований ГОСТ 51188-98. Защита информации;
5. испытания программных средств на наличие компьютерных вирусов;
6. использованием лицензионного программного обеспечения.

**Описание структурного подразделения организации**

Основной целью отдела является решение задач связанных с администрированием общесетевых системных сервисов, обеспечение доступа к ним и техническую поддержку пользователей. Сотрудники отдела обеспечивают непрерывную эксплуатацию серверного и сетевого оборудования, бесперебойную работу корпоративной сети ДГТУ. Поэтому в отделе большое количество операторов технической поддержки и администраторов. Структура отдела представлена н рисунке 1.

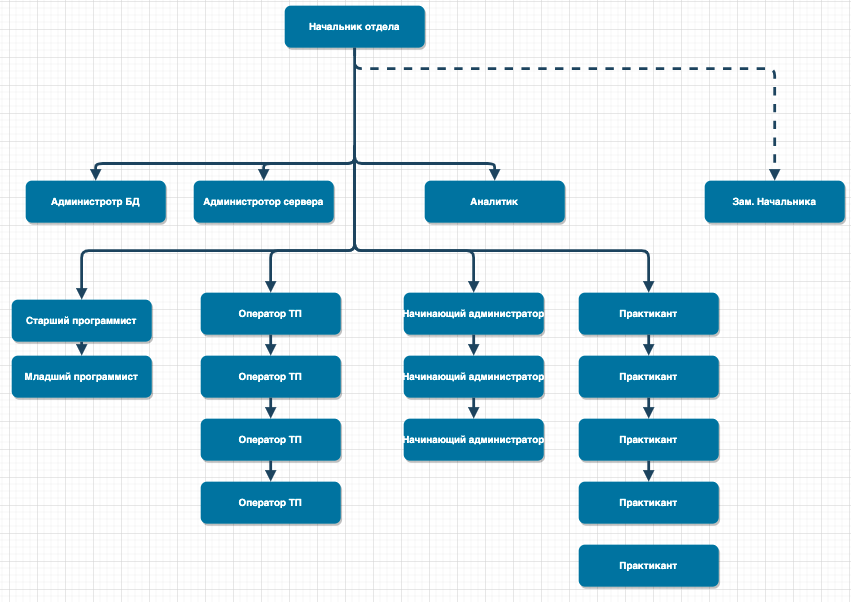


Рисунок 1 - структура отдела системного администрирования

**Описание работы, выполненной студентом за период прохождения практики**

За время прохождения практики были выполнены следующие пункты:

* Анализ актуальности темы.
* Первичный анализ предметной области, выделены основные преимущества и недостатки.
* Поиск и анализ научной литературы по данной теме.
* Изучение теоретического материала, изучение документации для написания программы.
* Сравнения и анализ аналогов, в данный отчет не включено.
* Анализ и математического обоснование компьютерного тестирования, в данный отчет не включено.
* Разработка программного средства.
* Тестирование программы.
* Формирование отчета практики.

Отдельно стоит выделить работу с перспективными информационными системами и технологиями.

Java — объектно-ориентированный язык программирования, разрабатываемый компанией Sun Microsystems с 1991 года и официально выпущенный 23 мая 1995 года. Изначально новый язык программирования назывался Oak (James Gosling) и разрабатывался для бытовой электроники, но впоследствии был переименован в Java и стал использоваться для написания апплетов, приложений и серверного программного обеспечения [6].

Программы на Java могут быть транслированы в байт-код, выполняемый на виртуальной java-машине (JVM) — программе, обрабатывающей байт-код и передающей инструкции оборудованию, как интерпретатор, но с тем отличием, что байт-код, в отличие от текста, обрабатывается значительно быстрее.

Язык Java зародился как часть проекта создания передового программного обеспечения для различных бытовых приборов. Реализация проекта была начата на языке [C++](http://progopedia.ru/language/c-plus-plus/), но вскоре возник ряд проблем, наилучшим средством борьбы с которыми было изменение самого инструмента — языка программирования. Стало очевидным, что необходим платформо-независимый язык программирования, позволяющий создавать программы, которые не приходилось бы компилировать отдельно для каждой архитектуры и можно было бы использовать на различных процессорах под различными операционными системами [6].

Spring Framework – это каркас для разработки приложений, использующих распределённую Internet-архитектуру. Как известно, существует несколько вариантов реализации этой архитектуры, в практике применен шаблон проектирования Model-View-Controller (MVC).

Данный каркас интегрирует в себе большое количество технологий и стандартов платформы Java. Наиболее важные из них:

1. Библиотеки J2EE для разработки Web-приложений.
2. Поддержка стандарта Java Persistence API.
3. Поддержка REST.
4. Интегрированные Web-контейнеры.
5. Планировщик.

Чтобы упростить доступ к этим технологиям и сократить время конфигурирования часто используют Spring Boot [6].

Для создания Web-приложения с применением шаблона проектирования MVC требуется шаблонизатор. Задачей шаблонизатора является интеграция HTML-шаблона и динамических данных (как правило, специальных элементов, которые транслируются в вызовы сервлетов). Таким образом, шаблон и сервлеты обеспечивают работу интерактивного приложения.

Spring Boot — это полезный проект, целью которого является упрощение создания приложений на основе Spring. Он позволяет наиболее простым способом создать web-приложение, требуя от разработчиков минимум усилий по его настройке и написанию кода [7].

Spring Boot обладает большим функционалом, но его наиболее значимыми особенностями являются: управление зависимостями, автоматическая конфигурация и встроенные контейнеры сервлетов

Первой особенностью является то, что Spring Boot собирает все общие зависимости и определяет их в одном месте, что позволяет разработчикам просто использовать их, вместо того, чтобы изобретать колесо каждый раз, когда они создают новое приложение

Второй превосходной возможностью Spring Boot является автоматическая конфигурация приложения.

MVC — это не шаблон проекта, это конструкционный шаблон, который описывает способ построения структуры нашего приложения, сферы ответственности и взаимодействие каждой из частей в данной структуре.

Впервые она была описана в 1979 году, конечно же, для другого окружения. Тогда не существовало концепции веб приложения. Tim Berners Lee (Тим Бернерс Ли) посеял семена World Wide Web (WWW) в начале девяностых и навсегда изменил мир. Шаблон, который мы используем сегодня, является адаптацией оригинального шаблона к веб разработке [7].

Идея, которая лежит в основе конструкционного шаблона MVC, очень проста: нужно чётко разделять ответственность за различное функционирование в наших приложениях на рисунке 2 изображено основное разделение

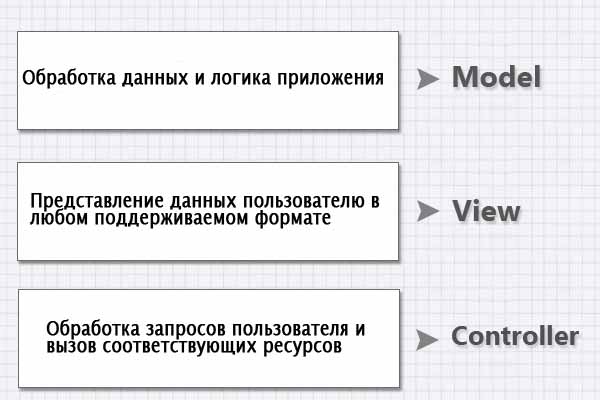


Рисунок 2 - разделение ответственности за различное функционирование

Контроллер управляет запросами пользователя (получаемые в виде запросов HTTP GET или POST, когда пользователь нажимает на элементы интерфейса для выполнения различных действий). Его основная функция — вызывать и координировать действие необходимых ресурсов и объектов, нужных для выполнения действий, задаваемых пользователем. Обычно контроллер вызывает соответствующую модель для задачи и выбирает подходящий вид.

Модель — это данные и правила, которые используются для работы с данными, которые представляют концепцию управления приложением. В любом приложении вся структура моделируется как данные, которые обрабатываются определённым образом. Что такое пользователь для приложения — сообщение или книга? Только данные, которые должны быть обработаны в соответствии с правилами (дата не может указывать в будущее, e-mail должен быть в определённом формате, имя не может быть длиннее Х символов, и так далее).

Модель даёт контроллеру представление данных, которые запросил пользователь (сообщение, страницу книги, фотоальбом, и тому подобное). Модель данных будет одинаковой, вне зависимости от того, как мы хотим представлять их пользователю. Поэтому мы выбираем любой доступный вид для отображения данных.

Модель содержит наиболее важную часть логики нашего приложения, логики, которая решает задачу, с которой мы имеем дело (форум, магазин, банк, и тому подобное). Контроллер содержит в основном организационную логику для самого приложения (очень похоже на ведение домашнего хозяйства).

Вид обеспечивает различные способы представления данных, которые получены из модели. Он может быть шаблоном, который заполняется данными. Может быть несколько различных видов, и контроллер выбирает, какой подходит наилучшим образом для текущей ситуации.

Веб приложение обычно состоит из набора контроллеров, моделей и видов. Контроллер может быть устроен как основной, который получает все запросы и вызывает другие контроллеры для выполнения действий в зависимости от ситуации [7].

Тестирование программы:

На рисунке 3 представлена структура проекта.

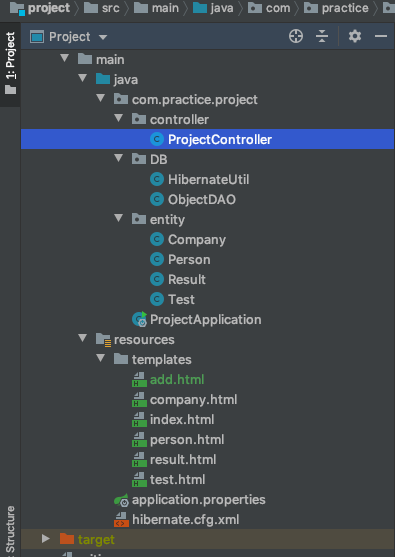
****

Рисунок 3 - структура проекта

На изображения 4-8 представлен интерфейс программы.

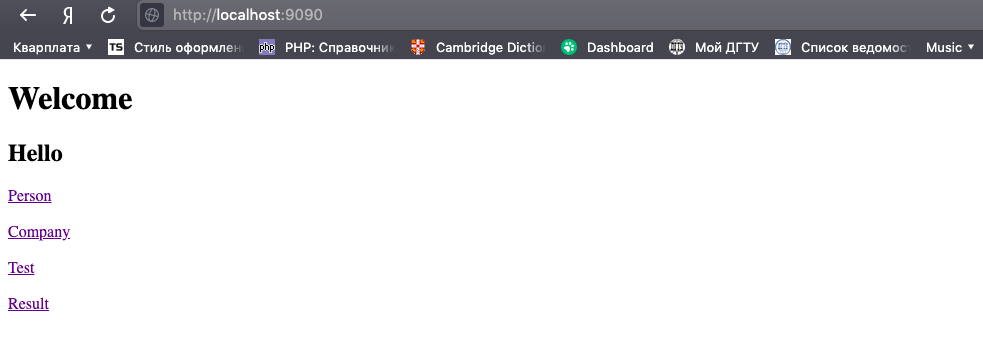
****

Рисунок 4 - начальная страница

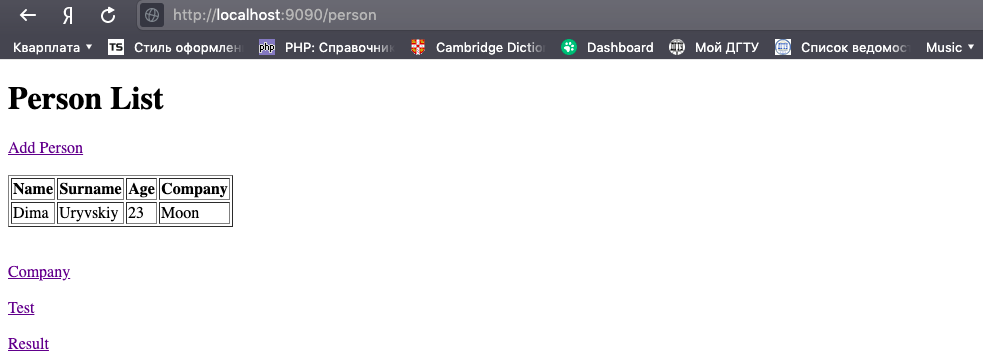
****

Рисунок 5 - страница с списком сотрудников

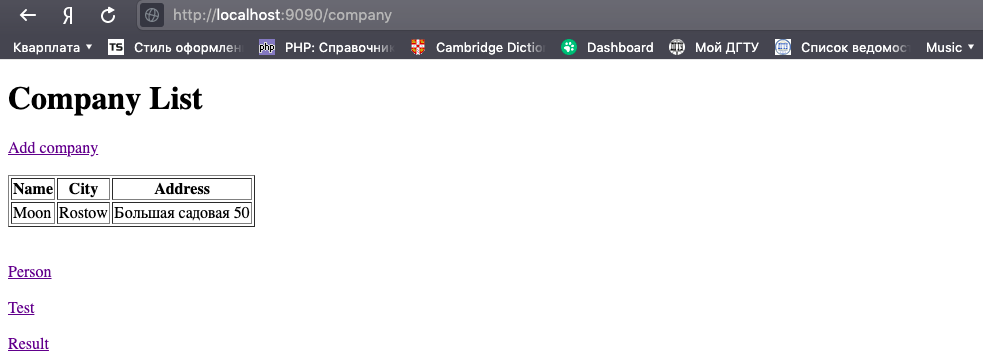
****

Рисунок 6 - страница с списком компаний

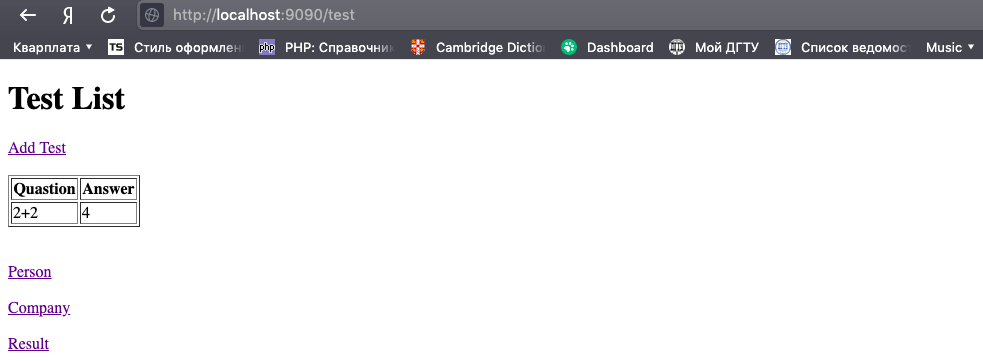
****

Рисунок 7 - страница с списком тестов

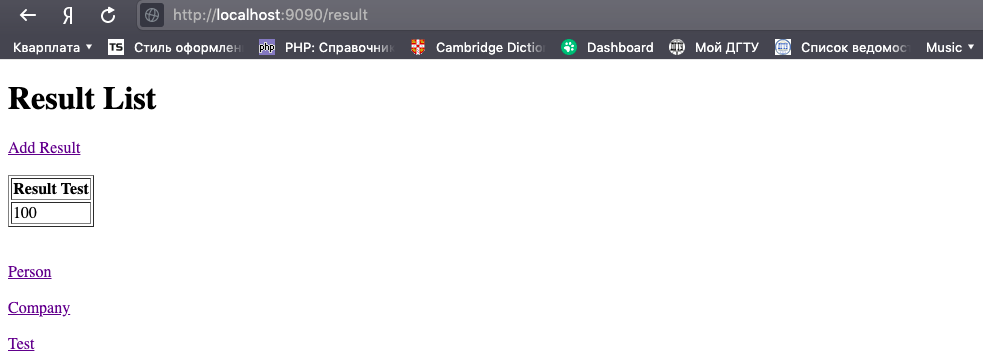


Рисунок 8 - страница с списком результатов

ЗАКЛЮЧЕНИЕ

Автоматизированное тестирование становиться важной частью жизни человека, там, где раньше использовали традиционное тестирование во многих сферах начали использовать компьютерное тестирование.

Важным преимуществом компьютерных систем тестирования, является удобство проведения теста, скорость получения результата, точность, количество участников тестирования и количество вопросов теперь не имеют границ. Однако существует некоторые недостатки:

1. вероятность выбора ответов наугад;
2. невозможно проследить логику ученика;
3. списывание [8].

Следовательно, можно сделать вывод о том, что тесты – это одна из форм контроля и оценки знаний, умений и навыков, которая может и должна использоваться в сочетании с другими формами и методами контроля и оценки. Данная тема имеет большую актуальность в современном информационном обществе, как в сфере образования, так и в любой другой сфере жизни.

**Список используемой литературы**

1. Могилев А.В. Технологии обработки текстовой информации. Технологии обработки графической и мультимедийной информации. Издатель - БХВ-Петербург 2013 г. 76с.
2. Роберт И.В. Теория и методика информатизации образования. Психолого-педагогический и технологический аспекты. Издатель- Бином. Лаборатория знаний; 2013 г. 128с.
3. <https://donstu.ru/structure/administrative/upravlenieinformatizatsii/otdel-administrirovaniya-i-sistemnogo-programmnogo-obespecheniya/> Электронный ресурс Отдел системного администрирования ДГТУ, дата обращения 09.07.2020
4. Т. А. Кабанова, В. А. Новиков. Тестирование в современном образовании Издательство: Высшая школа, 2010 г. 159 с.
5. Сиренко С.Н. Место тестирования в системе методов контроля и оценки знаний // Школьные технологии №2, 2011 г. 231 с.
6. <http://progopedia.ru/language/java/> Электронный ресурс Java – язык программирования, дата обращения 10.07.2020
7. <https://ruseller.com/lessons.php?id=666> Электронный ресурс Концепция MVC для начинающих, дата обращения 11.07.2020
8. Попов А.В. Тестирование как метод контроля качества знаний. Труды Санкт-Петербургского государственного университета культуры и искусств. Выпуск№ 1 / том 200 / 2013 год

# **ПРИЛОЖЕНИЕ А**

В приложении основные классы и файлы:

**class ProjectController:**

package com.practice.project.controller;  
  
import com.practice.project.entity.Company;  
import com.practice.project.entity.Person;  
import com.practice.project.entity.Result;  
import com.practice.project.entity.Test;  
import org.springframework.beans.factory.annotation.Value;  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.web.bind.annotation.ModelAttribute;  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RequestMethod;  
  
import java.util.ArrayList;  
import java.util.List;  
  
@Controller  
public class ProjectController {  
  
 private static List<Person> *persons* = new ArrayList<Person>();  
 private static List<Company> *companies* = new ArrayList<Company>();  
 private static List<Test> *tests* = new ArrayList<Test>();  
 private static List<Result> *results* = new ArrayList<Result>();  
  
 @Value("${message}")  
 private String message;  
  
 @Value("${error.message}")  
 private String errorMessage;  
  
 @RequestMapping(value = { "/", "/index" }, method = RequestMethod.*GET*)  
 public String index(Model model) {  
  
 model.addAttribute("message", message);  
  
 return "index";  
 }  
  
 @ModelAttribute  
 @RequestMapping(value = { "/person" }, method = RequestMethod.*GET*)  
 public String person(Model model) {  
  
 *persons*.add(new Person("Dima", "Uryvskiy", 23, "Moon"));  
  
 model.addAttribute("Persons", *persons*);  
  
 return "person";  
 }  
  
 @ModelAttribute  
 @RequestMapping(value = { "/company" }, method = RequestMethod.*GET*)  
 public String company(Model model) {  
  
 *companies*.add(new Company("Moon", "Rostow", "Большая садовая 50"));  
  
 model.addAttribute("companies", *companies*);  
  
 return "company";  
 }  
  
 @ModelAttribute  
 @RequestMapping(value = { "/test" }, method = RequestMethod.*GET*)  
 public String test(Model model) {  
  
 *tests*.add(new Test("2+2", "4"));  
  
 model.addAttribute("tests", *tests*);  
  
 return "test";  
 }  
  
 @ModelAttribute  
 @RequestMapping(value = { "/result" }, method = RequestMethod.*GET*)  
 public String personList(Model model) {  
  
 *results*.add(new Result(100));  
  
 model.addAttribute("results", *results*);  
  
 return "result";  
 }  
  
 @ModelAttribute  
 @PostMapping(value = { "/add" })  
 public String add(Model model) {  
  
 *results*.add(new Result(100));  
  
 model.addAttribute("results", *results*);  
  
 return "result";  
 }  
}

**class Person:**

package com.practice.project.entity;  
  
import org.hibernate.annotations.GenericGenerator;  
import javax.persistence.\*;  
  
@Entity  
@Table(name = "person")  
public class Person  
{  
 @Id  
 @GeneratedValue(generator = "increment")  
 @GenericGenerator(name = "increment", strategy = "increment")  
 @Column(name = "id\_person")  
 public Long id;  
  
 @Column (name = "name")  
 public String name;  
  
 @Column (name = "surname")  
 public String surname;  
  
 @Column (name = "age")  
 public Integer age;  
   
 @ManyToOne(fetch = FetchType.LAZY)  
 @JoinColumn(name = "company")  
 private Company company;  
  
  
 public Person(String name, String surname, Integer age, String company) {  
 this.name = name;  
 this.surname = surname;  
 this.age = age;  
 this.company = company;  
 }  
}

**class ProjectApplication:**

package com.practice.project;  
  
import com.practice.project.entity.Company;  
import com.practice.project.entity.Person;  
import com.practice.project.entity.Result;  
import com.practice.project.entity.Test;  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import com.practice.project.DB.ObjectDAO;  
import sun.misc.Unsafe;  
import java.lang.reflect.Field;  
  
  
@SpringBootApplication  
public class ProjectApplication {  
  
 public static void main(String[] args) {  
 // TODO: Заглушка для WARNING: An illegal reflective access operation has occurred  
 disableWarning();  
  
 SpringApplication.run(ProjectApplication.class, args);  
 ObjectDAO dao = new ObjectDAO();  
  
 Company company = new Company("Moon", "Rostow", "Большая садовая 50");  
 Person person = new Person("Dima", "Uryvskiy", 23, "Moon");  
 Test test = new Test("2+2", "4");  
 Result result = new Result(100);  
  
 dao.addValue(company);  
 dao.addValue(person);  
 dao.addValue(test);  
 dao.addValue(result);  
 }  
 }  
 public static void disableWarning() {  
 try {  
 Field theUnsafe = Unsafe.class.getDeclaredField("theUnsafe"); theUnsafe.setAccessible(true);  
 Unsafe u = (Unsafe) theUnsafe.get(null);  
 Class cls = Class.forName("jdk.internal.module.IllegalAccessLogger");  
 Field logger = cls.getDeclaredField("logger");  
 u.putObjectVolatile(cls, u.staticFieldOffset(logger), null);  
 } catch (Exception e) {  
  
 }  
 }  
}

**index:**

<!DOCTYPE HTML>  
<html xmlns:th="http://www.thymeleaf.org">  
<head>  
 <meta charset="UTF-8" />  
 <title>Welcome</title>  
</head>  
<body>  
<h1>Welcome</h1>  
<h2 th:utext="${message}">..!..</h2>  
<a href="/person">Person</a> <br/><br/>  
<a href="/company">Company</a> <br/><br/>  
<a href="/test">Test</a> <br/><br/>  
<a href="/result">Result</a> <br/><br/>  
</body>  
  
</html>

**application.properties:**

spring.main.banner-mode=off  
spring.main.log-startup-info=false  
server.port = 9090  
message=Hello  
error.message=error

**hibernate.cfg.xml:**

<!DOCTYPE hibernate-configuration PUBLIC  
 "-//Hibernate/Hibernate Configuration DTD 3.0//EN"  
 "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">  
  
<hibernate-configuration>  
 <session-factory>  
 <property name="hibernate.dialect">org.hibernate.dialect.PostgreSQLDialect</property>  
 <property name="connection.username">dima</property>  
 <property name="connection.password"></property>  
 <property name="hibernate.connection.driver\_class">org.postgresql.Driver</property>  
 <property name="hibernate.connection.url">jdbc:postgresql://localhost:5432/postgres</property>  
 <property name="connection\_pool\_size">1</property>  
 <property name="hbm2ddl.auto">create</property>  
 <property name="show\_sql">true</property>  
  
  
 <mapping class="com.practice.project.entity.Test"/>  
 <mapping class="com.practice.project.entity.Result"/>  
 <mapping class="com.practice.project.entity.Person"/>  
 <mapping class="com.practice.project.entity.Company"/>  
  
  
 </session-factory>  
</hibernate-configuration>

**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.practice</groupId>  
 <artifactId>project</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 <name>project</name>  
 <description>Demo project for Spring Boot</description>  
  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>2.0.0.RELEASE</version>  
 <relativePath/> <!-- lookup parent from repository -->  
 </parent>  
  
 <properties>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>  
 <java.version>1.8</java.version>  
 </properties>  
  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-thymeleaf</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
  
 <dependency>  
 <groupId>org.postgresql</groupId>  
 <artifactId>postgresql</artifactId>  
 <version>9.4.1211</version>  
 </dependency>  
  
 <dependency>  
 <groupId>org.hibernate</groupId>  
 <artifactId>hibernate-core</artifactId>  
 <version>3.6.10.Final</version>  
 </dependency>  
  
 <dependency>  
 <groupId>com.alexkasko.unsafe</groupId>  
 <artifactId>unsafe-tools</artifactId>  
 <version>1.2</version>  
 </dependency>  
  
 <dependency>  
 <groupId>org.hibernate</groupId>  
 <artifactId>hibernate-validator</artifactId>  
 <version>4.2.0.Final</version>  
 </dependency>  
  
 <dependency>  
 <groupId>org.thymeleaf.extras</groupId>  
 <artifactId>thymeleaf-extras-springsecurity5</artifactId>  
 <version>3.0.4.RELEASE</version>  
 </dependency>  
  
 </dependencies>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 </plugin>  
 </plugins>  
 </build>  
  
</project>