TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Informaatikainstituut
Tarkvaratehnika õppetool

Veebirakenduse loomine õppetegevuse haldamise lihtsustamiseks

Bakalaureusetöö

Üliõpilane: Jevgeni Menšenin
Üliõpilaskood: 112656IAPB
Juhendaja: Jekaterina Ivask

Tallinn
2015

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

---------- ----------

(*kuupäev*) (*allkiri*)

Аннотация

Дипломная работа посвящена созданию простого в использовании веб-приложения, направленного на решение проблем, с которыми сталкиваются студенты Таллиннского Технического Университета при организации своей учебной деятельности.

Для достижения данной цели прежде всего среди студентов был проведён и проанализирован опрос, позволивший установить требования, которым должно соответствовать приложение для полноценной работы. Также была построена архитектура приложения и создана база данных, элементы которых описаны в работе.

В ходе работы была реализована первая, работающая версия веб-приложения, решающая большинство выявленных проблем. Кроме того, было проведено юзабилити-тестирование, позволившее проверить, насколько приложение является дружелюбным к пользователю и выявить имеющиеся на данный момент проблемы пользовательского интерфейса для дальнейшего их решения.

Дипломная работа написана на русском языке, содержит 48 страниц текста, 6 глав, 22 рисунка.

Annotatsioon

Antud bakalauruse lõputöö eesmärgiks oli luua hea kasutatavusega veebirakenduse, mis aitaks lahendada probleeme millega puutuvad Tallinna Tehnika Ülikooli tudengid oma õppetegevuse organiseerimisel.

Selle eesmärgi saavutamiseks kõigepealt tudengite seas oli läbiviidud küsitlus, mille analüüs aitas asutada nõudmised millele peaks rakendus vastama täieliku töötamise jaoks. Lisaks oli loodud rakenduse arhitektuur ja andmebaas, mille elemendid on kirjeldatud lõputöös.

Töö käigus esimene töötav veebirakenduse versioon, mis lahendab enamikke tuvastatud probleeme, oli realiseeritud. Peale seda viidi läbi kasutatavuse testimise, mis lubas kontrollida kui sõbralik on see rakendus kasutaja jaoks ning tuvastada praegused probleemid, mis on seotud kasutajaliidesega, et hiljem võiks neid lahendada.

Lõputöö on kirjutatud vene keeles ning sisaldab teksti 48 leheküljel, 6 peatükki, 22 joonist.

Abstract

The goal of this bachelor`s thesis was to create easy-to-use web application aimed to solve problems which students of Tallinn University of Technology are facing while daily managing their learning activity.

To achieve this goal students were interviewed on the subject of application usefulness. After analysing their answers the main requirements needed for the proper functioning of application were established. Then the architecture of web application and database were created, elements of which are described in this thesis.

The carried out project resulted in first working version of web application which is solving most of the identified problems. Moreover, usability tests were carried out to examine how user-friendly the application is and to reveal user interface current problems to solve them in the future.

The thesis is in russian and contains 48 pages of text, 6 chapters, 22 figures.

Список сокращений и терминов

REST	***Representational State Transfer*** Метод взаимодействия компонентов распределённого приложения в сети Интернет[1]
MVC	***Model-View-Controller*** Метод иди дизайн шаблона, применяемый для успешного и эффективного соотношения пользовательского интерфейса и модели данных[2].
XML	***eXtensible Markup Language*** Язык разметки гипертекста для документов, содержащих структурированную информацию[3].
ORM	***Object-relational mapping*** Технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных» [4].
JSON	***JavaScript Object Notation*** Простой формат обмена данными, удобный для чтения и написания как человеком, так и компьютером[5].
Веб-приложение	***Web application*** Приложение, использующее веб-браузер в роли клиента[6].
Фреймворк веб-приложения	***Web application framework*** Каркас или основа, предоставляющая общую функциональность, помогающая разработчику создавать веб-приложения[7].

Инфосистема	***Информационная система*** Набор компонентов для сбора, хранения и обработки данных, а также для предоставления информации[8].
SQL	***Structured Query Language*** Стандарт и язык программирования, предназначенный для получения и изменения информации в базе данных[9].
Интерпретируемый язык программирования	***Interpreted language*** Язык программирования, в котором исходный код программы исполняется с помощью специальной программы-интерпретатора[10].
Компилируемый язык программирования	***Compiled language*** Язык программирования, исходный код которого преобразуется компилятором в машинный код[11].
HTML	***HyperText Markup Language*** Набор символов разметки, предназначенных для отображения веб-страницы в браузере[12].
Пользовательский интерфейс	***User Interface*** система правил и средств, регламентирующая и обеспечивающая взаимодействие программы с пользователем[13].
Ajax	***Asynchronous Javascript and XML*** подход к построению интерактивных пользовательских интерфейсов веб-приложений, заключающийся в «фоновом» обмене данными браузера с веб-сервером[14].
URL	***Uniform Resource Locator*** Уникальный адрес файла, доступного в Интернете[15].
Юзабилити	***Usability*** Атрибут качества, обозначающий, насколько просто пользователю использовать продукт[16].

Гедонистическое качество	***Hedonic quality*** Показатель качества системы, описывающий юзабилити и уровень полезности системы для пользователя при использовании[17].
Прагматическое качество	***Pragmatic quality*** Показатель качества системы, описывающий уровень мотивирования и стимулирования пользователя системой при использовании[17].

Список иллюстраций

Рисунок 20. Общая статистика ответов на каждый из пунктов опроса тестируемого приложения.

Дополнительный рисунок 1. Вопросы проводимого опроса

Дополнительный рисунок 2. Пункты проводимого опроса юзабилити-тестирования.

Содержание

1. Введение

Таллиннский Технический Университет (ТТУ) является крупнейшим в Эстонии техническим университетом, в котором, согласно данным с официального сайта, обучается около 14 000 студентов.

Будучи одним из этих студентов, на протяжении учёбы мне приходилось сталкиваться с определёнными проблемами при организации своей учебной деятельности. Так как на тот момент мною не было найдено хорошего решения большинства из этих проблем при помощи сторонних ресурсов, а учитывая количество обучающихся в ТТУ, можно предположить, что с данными проблемами сталкивается множество студентов, возникла идея создания веб-приложения, направленного на решение проблем организации учебной деятельности непосредственно для студентов ТТУ.

1.1 Описание проблемы

В настоящее время, при обучении в Таллиннском Техническом Университете, каждый семестр перед студентом стоит задача сбора информации о условиях сдачи задекларированных им предметов. Данная информация включает в себя список заданий (домашние работы, контрольные работы, лабораторные работы, экзамены и т.д.), необходимых для успешной сдачи предмета, «вес» каждого из заданий, сроки сдачи и дополнительные материалы - описание задания и вспомогательная информация. Кроме сбора информации, для своевременного и успешного выполнения заданий, необходимых для сдачи предмета, студент должен постоянно помнить о всех предстоящих заданиях и о условиях их выполнения, что порой, при большом количестве задекларированных предметов, бывает проблематично. Нередко происходят случаи, когда студент, не помня точной даты задания, или вовсе забыв о нём, не успевает должным образом подготовиться к сдаче.

Сложности добавляет ещё и тот факт, что, как правило, условия сдачи предметов и описание заданий разные преподаватели хранят на различных ресурсах. Это порой вызывает путаницу.

Кроме того, студенту необходимо иметь постоянно обновляющуюся статистическую информацию о своих промежуточных результатах (знать насколько хорошо были сданы все работы, знать каковы его шансы сдать предмет и на какую потенциально возможную максимальную и минимальную итоговую оценку, учитывая результаты выполненных на данных момент заданий и «вес» каждого из них). Динамический сбор и расчёт данной информации, при большом количестве предметов, нередко вызывает трудности.

К сожалению, существующие на данный момент приложения, для организации какой-либо деятельности, имеют возможность решить только малую часть из выявленных проблем, и не существует единого решения всех весомых организационных проблем для студентов Таллиннского Технического Университета, а постоянное их наличие зачастую приводит к снижению эффективности обучения.

1.2 Постановка задач

Основной целью данной работы является создание веб-приложения, способного решить представленные выше трудности в организации учебной деятельности.

Для более обстоятельного понимания проблемы и создания наиболее полезного приложения, необходимо провести опрос среди студентов Таллиннского Технического Университета и проанализировать результаты.

Кроме того, данное приложение должно быть привлекательно как для студентов, так и для преподавателей, а соответственно, оно должно быть, как можно более простым и удобным в использовании. Для проверки данных критериев, после создания прототипа приложения, необходимо провести юзабилити-тестирование.

1.3 Обзор работы

Данная работа делится на 3 основные части.

В первой части работы описывается анализ, предшествующий созданию приложения. Приводится описание проведённого среди студентов Таллиннского Технического Университета опроса и сбора полученной информации, а также на основе данного опроса – ставятся основные проблемы, требующие решения.

Во второй части работы описывается реализация веб-приложения. Приводится описание архитектуры системы, как клиентской, так и серверной части, описываются использованные в создании приложения технологии и методики, а также причины их использования с описанием реализации в данном приложении.

В третьей части работы описывается юзабилити-тестирование. Описываются использованные в ходе тестирования технологии и ресурсы, процесс проведения тестирования и полученные в ходе тестирования результаты.

Кроме того, с кодом созданного приложения можно ознакомиться по следующей ссылке: https://bitbucket.org/jevgenimensenin/diplom/src/ [обновлено 13.01.2015].

2. Анализ данных

Для определения основных проблем организации учебной деятельности, составления функциональных и нефункциональных требований к приложению, а также для упрощения процесса проектировки приложения, был проведён анализ, состоящий из двух этапов: сбора информации и анализа информации.

2.1 Сбор информации

Среди студентов Таллиннского Технического Университета был проведён опрос. Чтобы опрос отражал существующие на данный момент проблемы, целевой группой опроса были студенты второго и третьего курса ТТУ или закончившие университет прошлом году. Данный критерий выделил группу людей, способных на основе накопленного опыта учёбы, сделать объективные выводы о организации учебной деятельности. Опрос содержал 8 вопросов, целью которых было:

- Выяснить, сталкиваются ли студенты ТТУ с проблемами организации учебной деятельности, с которыми столкнулся автор.
- Выяснить, считают ли студенты данные проблемы актуальными и требующими решения.
- Выяснить, с какими проблемами приходится сталкиваться студентам чаще всего.
- Выяснить, пользуются ли студенты инфосистемой ÕIS для решения данных проблем и если нет, то почему.
- Выяснить, хотят ли студенты иметь приложение, решающее имеющиеся организационные проблемы.

Опрос был проведён с использованием сервиса Google Forms, с помощью которого можно создавать опросы и анализировать полученные ответы. Кроме того, данный сервис предоставляет графическую статистику полученных ответов.

Было опрошено 52 студента, чьи ответы позволили выявить чёткие тенденции, которых, по мнению автора, достаточно для определения и конкретизации имеющихся проблем.

2.2 Анализ информации

Вторым этапом являлся анализ полученной в ходе опроса информации. Далее будут представлены задаваемые вопросы и статистика ответов на них, которая будет проанализирована.

Рисунок 1. Второй вопрос проведённого опроса.

Данный вопрос был направлен на определение масштабов проблем организации учебной деятельности среди студентов ТТУ. Как видно из данного графика, подавляющее большинство (87%) опрошенных когда-либо сталкивались с данными проблемами. Данная статистика позволяет предположить, что большинство студентов имели трудности в организации учебной деятельности.

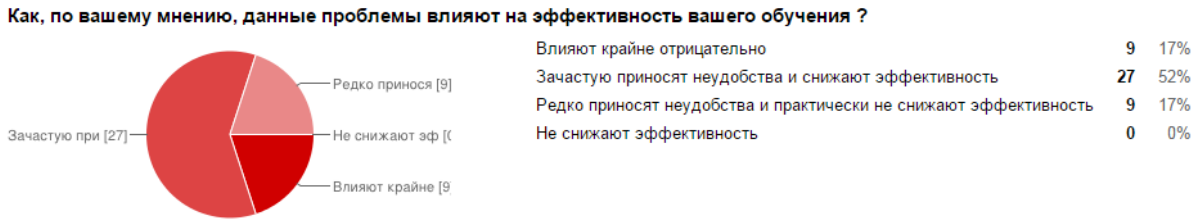


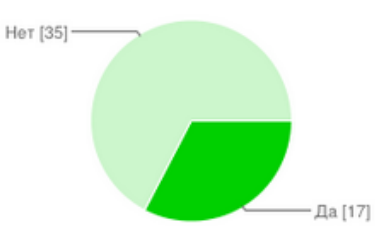
Рисунок 2. Третий вопрос проведённого опроса.

Целью следующего вопроса было выяснить, как сильно, по мнению студентов, данная проблема отражается на их учебной деятельности. Согласно полученным результатам, ни один студен не ответил, что данная проблема совершенно не снижает эффективности обучения. Крайне отрицательное влияние отметили 17% опрошенных, а 52% сочли данную проблему часто приносящей неудобства и снижающей эффективность, что в сумме представляет 69% студентов. Данные результаты позволяют утверждать, что проблема является актуальной и снижает эффективность обучения.

Рисунок 3. Четвёртый вопрос проведённого опроса.

Данный вопрос был задан с целью конкретизации организационных проблем и определения наиболее часто встречающихся в учебной практике. Кроме того, данный вопрос помог определить функциональные требования к приложению. Как мы видим, наиболее часто встречающиеся проблемы - это путаница с условием выполнения заданий (73%), трудности сбора информации о условиях сдачи предмета (65%) и сложности в доступе к необходимым для учёбы ресурсам (62%). Данные проблемы, по результатам опроса, являются наиболее критическими и требуют незамедлительного решения. Первая версия изготавливаемого приложения будет направлена на решение именно этих проблем.

Использовали ли вы когда-либо инфосистему ÕIS для организации учебной деятельности



Да	17	33%
Нет	35	67%

Если да, то для чего ?

Получение информации о предметах, преподавателях.
Для просмотра расписания
рассписание
для расписания и для декларации
Расписание
составление расписания, регистрация на экзамены
Составление расписания, информация о времени консультаций
осмотр задолженностей и необходимых предметов

Если нет, то почему ?

Работа в ÕIS занимает много времени, так как долго открываются загружаются страницы.
Медленно
Система не очень удобна, трудно найти необходимые разделы. Так же нет информации, что возможно делать в этой системе
Не знал о подобной возможности.
А как ее исользовать?
Не стоит времени.
Не знал о существовании данной функциональности, к тому-же ОЙС работает крайне медлено
неудобна

Рисунок 4. Пятый, шестой, седьмой вопрос проведённого опроса.

После вопроса, определяющего функциональные требования, были заданы 3 вопроса о инфосистеме, имеющей возможность решать некоторые организационные проблемы. В данном случае – это ÕIS. Данные вопросы помогли выяснить, пользуются ли студенты данным приложением и как именно. Если же не пользуются, то спрашивалось - почему. Как видно из графика, всего 33% студентов используют ÕIS в данном ключе и в основном – только для просмотра расписания. Большинство же студентов (67%) – не используют данную функциональность инфосистемы ÕIS. Основные причины – данное приложение работает медленно, не очень удобно, или же студены попросту не знают о существовании такой функциональности. Имея данные результаты, можно предположить, что ÕIS, являясь большой инфосистемой, не может решать индивидуальные организационные проблемы студентов, так как это не является его основной целью и его интерфейс не является дружелюбным к пользователю. Это позволяет сделать выводы, что создаваемое приложение должно быть узконаправленным - быть направленно исключительно на решение выявленных проблем

и быть простым и понятным в использовании. Для проверки данных критериев и поддержке их выполнения, необходимо провести юзабилити-тесты.

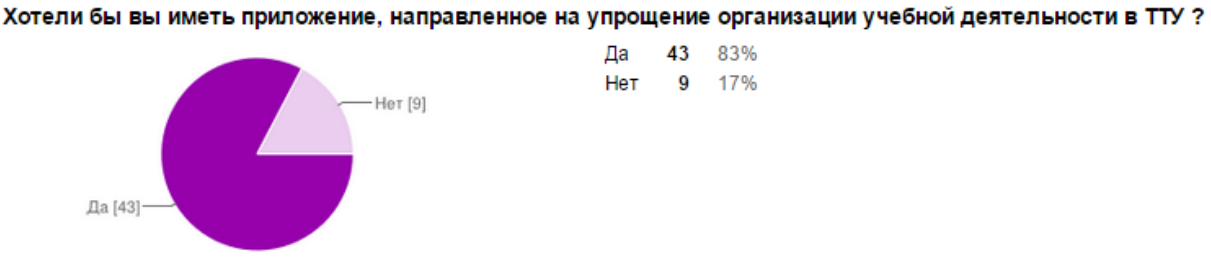


Рисунок 5. Восьмой вопрос проведённого опроса.

Последний вопрос был направлен на выяснение потенциальной популярности создаваемого веб-приложения. Полученные ответы показывают, что 83% опрошенных хотели бы иметь данное приложение. Это означает, что студенты будут пользоваться данным приложением и его создание будет оправдано.

2.3 Требования к приложению

Проведение данного опроса и его анализа позволило определить функциональные и нефункциональные требования к приложению.

2.3.1 Функциональные требования

Приложение должно решать:

1) Путаницу с условиями выполнения заданий

- У каждого студента должен быть личный аккаунт, в котором он сможет создавать или выбирать список задекларированных им предметов.
- Каждый предмет из данного списка должен являться уникальной парой предмет-преподаватель.
- Предмет должен содержать список заданий, которых требует преподаватель для его сдачи. Данный список может составить как преподаватель, а после студенты, учащиеся у данного преподавателя, лишь выбрать предмет из списка, так и сами

студенты могут создавать данные списки в пределах своего аккаунта. Задания должны содержать тип (контрольная работа, лабораторная работа и т.д.), название задания, описание задания, неделю сдачи, «вес» задания и комментарий.

- В любой момент времени студент должен иметь возможность видеть предстоящие на текущей неделе задания, а также иметь возможность видеть задания по задекларированным им предметам, которые проходили или будут проходить на любой неделе семестра.

- В любой момент времени у студента должна быть возможность просматривать информацию о заданиях к взятым им предметам, а также иметь возможность вносить результаты по сданным им заданиям и видеть статистические данные, основанные на результатах, которые учитывают «вес» предметов.

- У студента должна быть возможность сделать экспорт дат заданий в .CSV файл, для последующего импорта в календарь (Google Calendar, MS Outlook).

2) Трудности сбора информации о условиях сдачи предмета

- Приложение должно быть разделено на две роли: студент и преподаватель.

- Преподаватель должен иметь возможность оформлять преподаваемый им предмет – создавать список заданий, необходимых для сдачи данного предмета.

- Студент должен иметь возможность просматривать предметы, оформленные преподавателями и «подписываться» на данные предметы.

3) Сложности в доступе к необходимым для учёбы ресурсам

- Приложение должно позволять, при создании условий сдачи предмета (как студенту, так и преподавателю), добавлять и хранить необходимые для прохождения предметы файлы (более подробное описание заданий, вспомогательные материалы)

- Студент и преподаватель должны в любой момент времени иметь доступ к данным файлам.

2.3.2 Нефункциональные требования

- Приложение должно быть узконаправленным и решать чётко поставленные задачи, не предоставляя большого количества лишней функциональности.
- Приложение должно быть дружелюбным к пользователю (интуитивным и простым в использовании).
- Приложение должно иметь возможность для расширения функциональности в будущем.
- Приложение должно быть мультиязычным (работать на эстонском, русском и английском языках).
- Приложение должно корректно работать в следующих браузерах:
    - Google Chrome
    - Mozilla Firefox
- Приложение не должно выдавать критических ошибок, нарушающих работу пользователя. Любая полученная ошибка должна быть обработана и показана понятным простому пользователю сообщением.
- Каждый пользователь должен иметь ограниченный доступ к данным системы. Количество информации, которая доступна пользователю должна быть определена имеющимися у пользователя ролями.
- Приложение должно быть надёжным.

3. Описание реализации

На основе проведённого анализа и поставленных функциональных и нефункциональных требований, для создания приложения был построен архитектурный каркас, включающий в себя структуру взаимодействия клиента и сервера, методы взаимодействия компонентов в сети, а также шаблоны проектирования для разделения модулей приложения и обеспечения их независимости. Кроме того, были выбран оптимальные языки разработки и их расширения, необходимые для эффективного создания веб-приложения, отвечающего требованиям. В данной главе будет представлено описание данного архитектурного каркаса, а также причины выбора использующихся в проекте технологий и их преимущества.

3.1 Общая архитектура

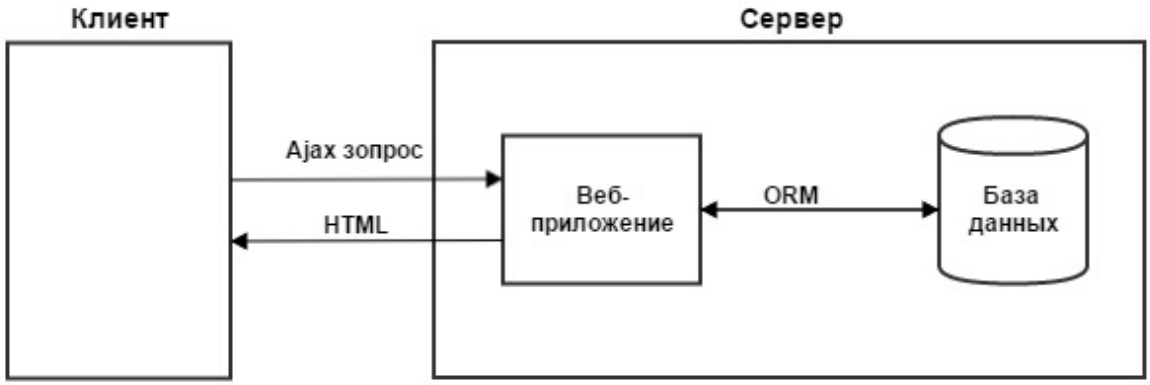


Рисунок 6. Общая архитектура приложения.

Данное веб-приложение реализует двухуровневую архитектуру клиент-сервер. В данном случае, клиентом является используемый пользователем браузер. При совершении пользователем какого-либо действия, клиент посылает серверной части Ajax запрос. Обрабатывая данный запрос, при необходимости веб-приложение общается с базой данных используя ORM метод. После этого, клиент получает ответ в виде посланного сервером HTML отображения, который отображается пользователю.

3.2 Серверная часть

Для разработки серверной части приложения был выбран язык Java.

Перед построением архитектуры проекта, было необходимо определиться с языком программирования, на котором будет написан проект. Так как приложение должно было стать веб-направленным, рассматривались исключительно языки веб-разработки. Были проанализированы преимущества и недостатки данных языков (Java, Ruby, Python, PHP). Учитывая опыт работы с ними, окончательный выбор стоял между языками Java и PHP.

Язык Java не является интерпретируемым языком программирования и на начальном этапе разработки требует дополнительных затрат времени на построения каркаса проекта, на создание и настройку связи с сервером и на компиляцию проекта. Используя язык PHP можно избежать данных затрат, однако данное решение подходит скорее для построения небольших приложения без сложной бизнес-логики, так как, учитывая особенности интерпретаторов, при росте приложения, его производительность будет снижаться. Несмотря на затраты времени на начальном этапе, а также учитывая тот факт, что создаваемое приложение будет иметь работающую на серверной части бизнес-логику и то, что оно должно иметь возможность расширения, в перспективе более подходящим будет использовать компилируемый язык Java. Данное решение позволит повысить эффективность работы данного приложения и его производительность.

Кроме того, одним из крайне важных преимуществ языка Java является поддержка большого количества расширений и библиотек, которые позволяют сократить время разработки и увеличить защищённость приложения. Данные расширения были использованы в работе.

3.2.1 Обеспечение взаимодействия пользователя с системой

Для организации взаимодействия отдельных компонентов приложения и для осуществления взаимодействия пользователя с системой был использован шаблон программирования MVC (model-view-controller).

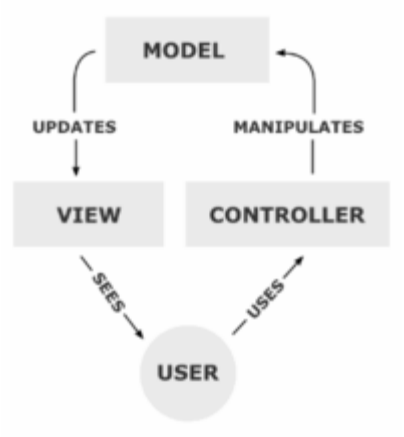


Рисунок 7. Схема работы шаблона MVC (взято из: https://commons.wikimedia.org/wiki/File:MVC-Process.png)

Данный шаблон является представлением трёх взаимодействующих компонентов – модели, вида и контроллера.

Модель является набором правил и ограничений, необходимых для использования некой информации (в нашем случае – объект). Вид является компонентом отображения данной информации (в нашем случае – HTML страница). Контроллер является связующим звеном между пользователем и системой. Принимая запрос пользователя, контроллер использует модель и вид для создания необходимого ответа.

Обычно работа MVC следует следующему сценарию:

1. Пользователь посылает серверной части запрос из пользовательского интерфейса.

Рисунок 8. Пример пользовательского интерфейса посылающего запрос серверной части.

   При нажатии на выделенную красным цветом кнопку происходит вызов Javascript метода. Данный метод, посылает запрос серверной части в виде названия метода через URL и необходимых переменных посредством Ajax запроса.

2. Контроллер принимает данный запрос, обрабатывает его и посылает команду сервису, общающемуся с базой данных.

```
@RequestMapping(value = "/getTasksByWeek", method = RequestMethod.POST, produces = "application/json")
public @ResponseBody String getTasksByWeek(@RequestParam("week") Integer week, @RequestParam("studentId")
Long studentId){
    HashMap<String, List<Task>> thisWeekTasks = studentService.getThisWeekTasks(studentId, week);
    Gson gson = new Gson();
    String thisWeekTasksJson = gson.toJson(thisWeekTasks);
    return thisWeekTasksJson;
}
```

Рисунок 9. Пример метода контроллера.

Метод getTasksByWeek является методом класса контроллера StudentController. Контроллер посылает полученные параметры.

3. Используя метод ORM, общение с базой данных происходит через объекты (модели). Соответственно, на данном этапе, послав базе данных запрос, полученный ответ представляет собой определённый набор объектов.

4. Компонент сервиса обрабатывает полученный от базы ответ, производя с ним необходимые действия и передаёт обработанный ответ контроллеру. В описываемом случае, сервис обрабатывает полученные объекты StudentSubject и формирует набор объектов Task – набора заданий для запрашиваемой недели.

5. Полученный от сервиса ответ контроллер посылает виду, который строит пользовательский интерфейс на основании полученных от контроллера данных.

После завершения приведённого выше сценария, пользователь видит в пользовательском интерфейсе запрашиваемую информацию.

Кроме того, в данном приложении шаблон MVC является хорошим решением, так как используя разделение приложения на отдельные компоненты модель-вид-контроллер, он позволяет добиться наименьшей зависимости компонентов приложения друг от друга. Данное преимущество обеспечивает возможность достаточно простого добавления и изменения элементов как серверной, так и клиентской части, что обеспечивает приложение качеством расширяемости.

В создаваемом приложении создание и использование MVC модели осуществлялось за счёт реализации Java фреймворка – Spring MVC. Данный фреймворк предоставляет набор удобных методов, облегчающих создание и конфигурацию данной методики.

Для конфигурации всего проекта и автоматизации его сборки, использовался фреймворк Apache Maven. Ввиду того, что в создании данного приложения использовались

фреймворки и библиотеки, которые, как правило, имеют сложные зависимости, большим плюсом использования Maven была его особенность – управление зависимостями, позволяющая разрешать конфликты версий фреймворков и библиотек и в случае необходимости легко переходить на новые версии.

Управление передаваемой информации между пользователем и системой обеспечивалось за счёт использования архитектуры REST. Было решено использовать REST архитектуру, ввиду её надёжности, производительности и простоты.

3.2.2 Процесс аутентификации

На данный момент, в соответствии с требованиями системы, необходимо обеспечить присваивание пользователям ролей, и по типу их роли – разделять доступную им информацию. Соответственно, для полноценной работы создаваемого приложения, необходима надёжная и безопасная система аутентификации. Так как для построения каркаса приложения был использован фреймворк Spring MVC, для обеспечения приложения системой аутентификации была использована библиотека Spring Secutiry. Данная библиотека предоставляет набор механизмов для построения данных систем и обеспечения их надёжности.

В приложении создание данной системы аутентификации состояло из следующих этапов:

1. Конфигурация Spring Security.

   Конфигурация осуществляется за счёт описания необходимых элементов в XML файле.

```
<http pattern="/login" security="none"/>
<http pattern="/frontend/**" security='none' />
<http use-expressions="true">
    <intercept-url pattern="/**" access="isAuthenticated()"/>
    <form-login login-page="/login" authentication-failure-url="/login"
    default-target-url="/successLogin" always-use-default-target="true"/>
    <logout logout-url="/logout" logout-success-url="/index"/>
</http>
```

Рисунок 10. Пример конфигурационного кода Spring Security.

2. Создание системы ролей.

На данный момент, функциональность приложения поделена на 2 роли – студент и преподаватель. Для этого, помимо таблицы пользователей, необходимо иметь в базе данных таблицу ролей и связывающую таблицу ролей с пользователями. Благодаря данной системе, каждый пользователь будет связан с необходимыми ролями. Структуру базы данных, включающую данные таблицы можно посмотреть на Рисунке 11.

Помимо этого, ввиду использования ORM-методики, необходимо создать виртуальное представление данной структуры на серверном уровне приложения.

3. Обеспечение безопасности

Наиболее важным элементом безопасности является надёжное хранение паролей пользователей. Для обеспечения данного качества, пароли необходимо хешировать. В данной работе был выбран алгоритм хеширования паролей SHA-256, являющийся на данный момент достаточно надёжный методом хеширования, позволяющим избежать коллизий.

Фреймворк Spring Security позволяет осуществить имплементацию метода SHA-256 лишь описав его в конфигурационном XML файле.

3.3 База данных

Для хранения информации, необходимой для приложения и для манипулированием ею, была использована база данных. Так как на момент написания приложения имелся опыт работы лишь в SQL базами данных, было решено использовать PostgreSQL.

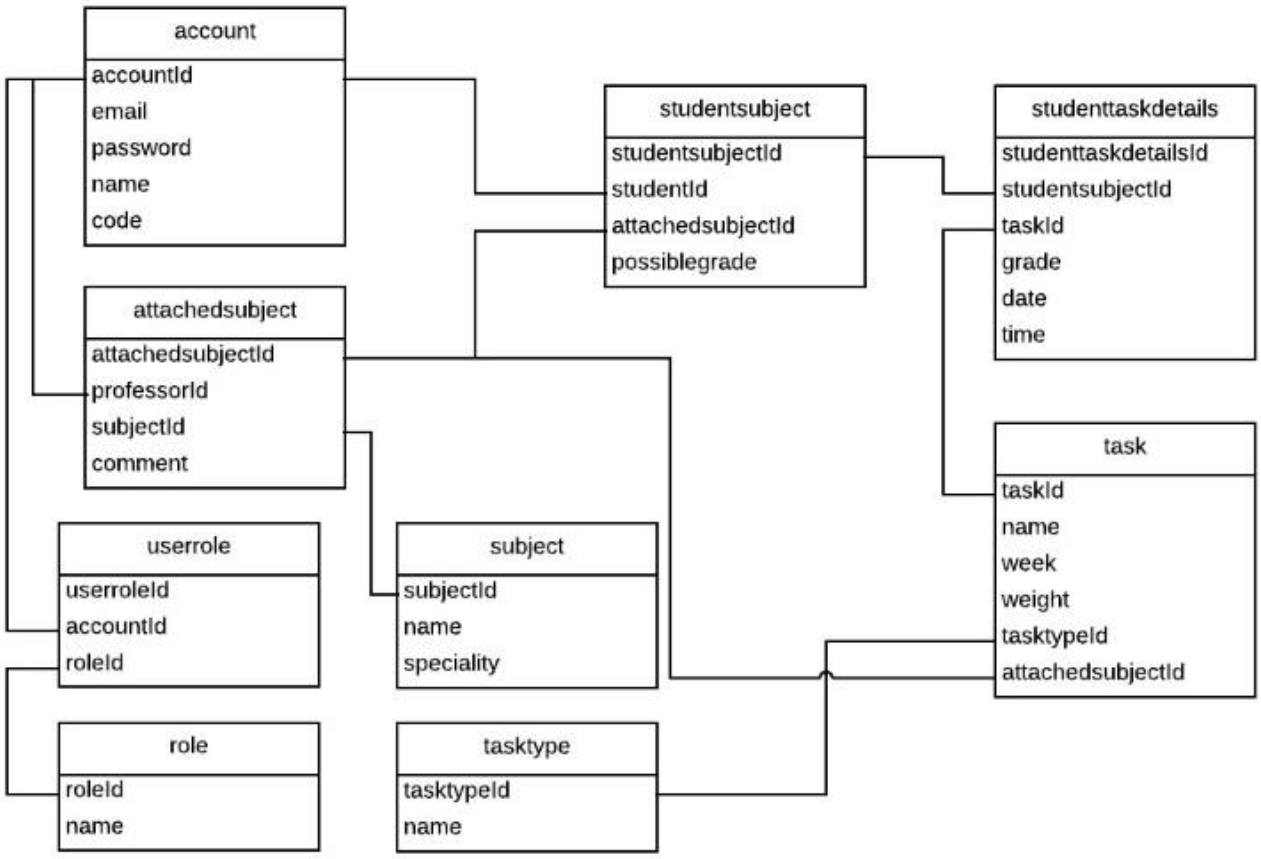


Рисунок 11. Диаграмма базы данных приложения.

Для создания базы и представления данных использовался метод объектно-реляционного представления (ORM). Данный метод связывает базу данных с концепцией объектно-ориентированного программирования. Имеющиеся таблицы имеют виртуальное представление на серверном уровне, что позволяет оперировать ими в рамках объектно-ориентированной концепции. В шаблоне модель-вид-контроллер, использованном в создании приложения, данное виртуальное представление является моделью.

Для реализации ORM модели использовалась фреймворк языка Java – Hibernate. Данный фреймворк позволяет связывать Java классы (модели) с имеющимися таблицами базы данных, значительно упрощая данный процесс и сокращая время разработки.

```java
@Entity
@Table(name="attachedsubject")
public class AttachedSubject {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long attachedSubjectId;

    @ManyToOne
    @JoinColumn(name = "professorid")
    @JsonIgnore
    private Account professor;

    @ManyToOne
    @JoinColumn(name = "subjectid")
    private Subject subject;

    @OneToMany(fetch = FetchType.EAGER, mappedBy = "attachedSubjectId")
    private List<Task> tasks;

    private String comment;

    public String getComment() {
        return comment;
    }

    public void setComment(String comment) {
        this.comment = comment;
    }
```

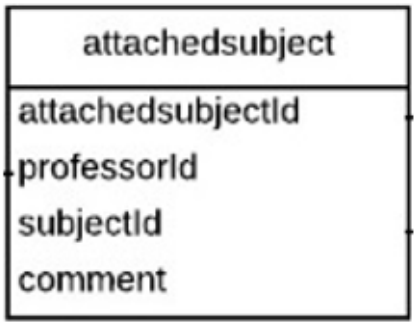


Рисунок 12. Пример модели, созданной при помощи Hibernate.

В Hibernate для организации связи между таблицами базы данных и моделями серверной части, используется проецирование Java-классов (mapping). В данном случае, при помощи аннотаций. Представленная выше модель является объектно-ориентированным представлением таблицы attachedsubject. Аннотация @Entity показывает, что данный класс является проекцией таблицы из базы данных, @Table указывает название таблицы. @Id и @GeneratedValue указывают, что обозначенный параметр объекта является автогенерируемым уникальным идентификатором таблицы. Аннотация @ManyToOne означает связь много к одному, соответственно у одного создателя может быть множество созданных им предметов. Остальные переменные объекта, при отсутствии аннотаций, также являются проекцией столбцов таблицы при условии, что имена столбцов и переменных будут совпадать. Для создания полноценной объектно-ориентированной модели базы данных, необходимо создать похожие Java классы на основании всех таблиц базы данных.

```
public class AttachedSubjectDaoImpl implements AttachedSubjectDao{

    @PersistenceContext
    private EntityManager em;

    public void saveTasksToSubject(Task task){
        em.persist(task);
    }
```

Рисунок 13. Пример использования ORM для сохранения данных – сервис.

```
@RequestMapping(value = "/saveTasksToSubject", method = RequestMethod.POST)
public @ResponseBody String saveTasksToSubject(@RequestParam("tasks") String tasks,
        @RequestParam("attachedSubjectId") Long attachedSubjectId,
        @RequestParam("comment") String comment) throws Exception{
    Gson gson = new Gson();
    List<Task> tasksToDB = gson.fromJson(tasks, new TypeToken<List<Task>>(){}.getType());
    for(Task task: tasksToDB){
        task.setAttachedSubjectId(attachedSubjectId);
        attachedSubjectService.saveTasksToSubject(task);
    }
```

Рисунок 14. Пример использования ORM для сохранения данных – контроллер.

Приведённый выше пример показывает простоту использования ORM для сохранения данных. На рисунке 13 производится имплементация интерфейса AttachedSubjectDao с использованием встроенной функции сохранения данных. При наличие данной имплементации, как мы видим на рисунке 14, контроллеру необходимо только воспользоваться данным методом, подав в виде параметра имеющий проекцию (модель) объект, который необходимо сохранить – attachedSubjectService.saveTasksToSubject(task). При вызове данного метода, в базе данных, в таблице Task будет добавлен ряд данных, соответствующий посланному, в качестве параметра, объекту.

Кроме всего прочего, плюсом использования ORM и библиотеки Hibernate является то, что за счёт использования в транзакциях между базой данных и сервером объектов, данный метод защищает приложение от прямого внедрения SQL кода в запросы.

3.4 Пользовательский интерфейс

Для обеспечения эффективного взаимодействия пользователя с системой, было необходимо создать удобный и простой в использовании пользовательский интерфейс. В виду того, что большинство потенциальных пользователей приложения в

повседневной жизни пользуются методом письма слева-направо-сверху-вниз, распределение их внимания работает таким-же образом. Следуя данной логике, распределение будет происходить по следующей схеме:

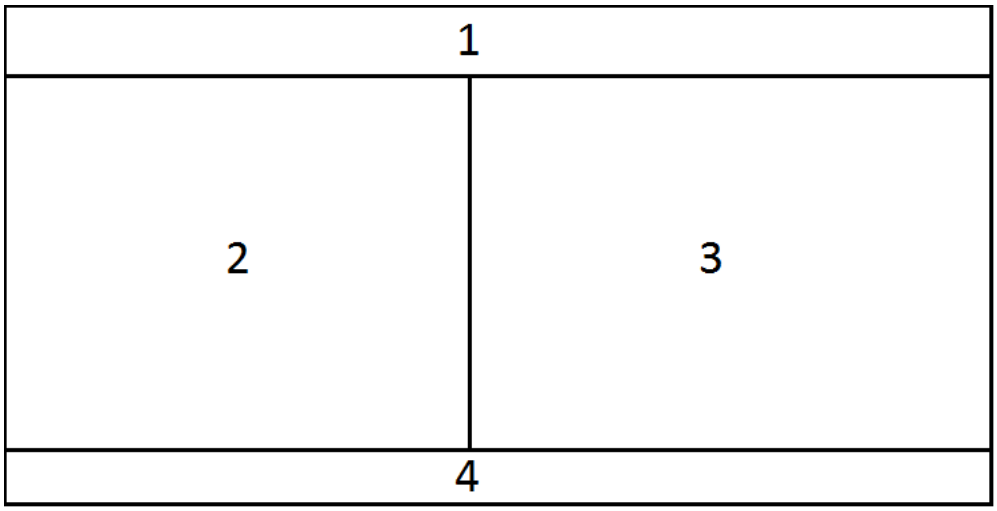


Рисунок 15. Схема распределения внимания пользователя при использовании приложения.

На представленной выше схеме, номера выделенных секторов обозначают, в какой последовательности пользователь будет обращать внимания на эти части пользовательского интерфейса приложения. Соответственно, чем меньше цифра, тем более важный элемент должен находиться в данном секторе.

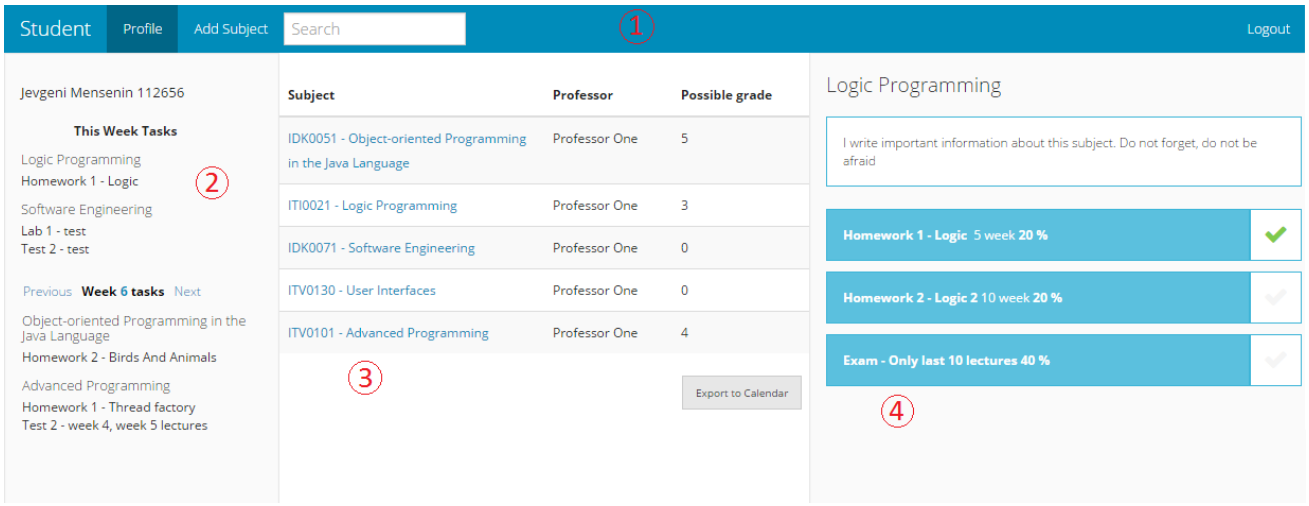


Рисунок 16. Пользовательский интерфейс профиля студента.

Представленный выше рисунок является примером распределения элементов в окне пользовательского интерфейса приложения, а именно – в профиле студента. В данном

профиле обозначенные цифрами элементы распределены по принципу схемы на рисунке 15 - по уровню их важности:

1. Так как данный сектор является наиболее важным, в него были помещены наиболее части используемые элементы, а именно – навигационная панель и быстрый поиск. Данная панель вне зависимости от хода работы системы и переключения страниц, за исключением страницы авторизации, всегда остаётся на одном месте. Данное решение позволяет пользователю эффективно пользоваться данной панелью.

2. Данный элемент содержит информацию о пользователе, а также блок, содержащий важную для пользователя информацию в конкретный момент времени, которая необходима при входе в систему. Данный блок состоит из двух частей. В первой части пользователю предоставляется динамически обновляющийся список заданий выбранных пользователем предметов, которые будут проходить на текущей неделе, а вторая часть блока содержит список заданий, проходящих на выбранной пользователем неделе семестра (неделя может быть выбрана от 1 до 16). Списки содержат важную для каждого студента информацию, в виду чего блок 2 имеет данное расположение.

3. Основным объектом данного сектора является таблица, содержащая выбранные студентом предметы, имена преподавателей, преподающих данный предмет зашедшему в систему студенту, а также максимальные оценки, который студент может получить за предмет. Оценки рассчитываются учитывая баллы полученные за сданные задания, их «вес» и «вес» не сданных на данный момент предметов. Кроме того, данный блок является своего рода управляющим блоком для 4 сектора, так как отображающаяся в 4 секторе информация напрямую зависит от выбранного в 3 секторе предмета.

4. Так как информация данного сектора напрямую зависит от действий, произведённых в секторе 3, то данный блок следует сразу за ним. В данном блоке содержится подробная информация о предмете, либо представленное на рисунке 17 окно действий при попытке экспорта заданий в .CSV файл для календаря.

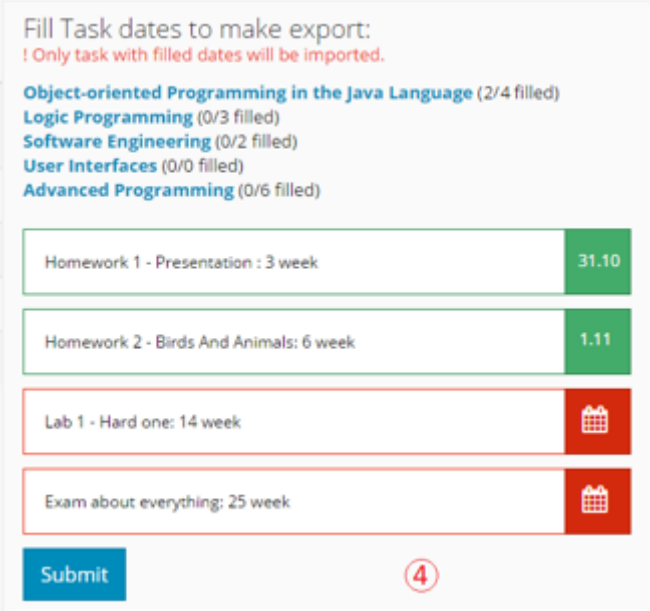


Рисунок 17. Окно действий при экспорте заданий в .CSV файл.

Так как при описании условий выполнения предмета преподаватель может поставить только номер недели, на которой будет проходить задание (у разных групп задания могут проходить в разные дни), каждый студент может индивидуально выставить в своём профиле дату и время для каждого задания. При экспорте заданий в .CSV файл, будут экспортированы только те задания, у которых была проставлена дата. В приведённом на рисунке 17 окне, студент видит список предметов, рядом с которыми имеется количество заданий с заполненными датами и общее количество заданий предмета. При нажатии на предмет, студент получает список заданий. Для более удобного ориентирования между заданиями с заполненными и незаполненными датами, они имеют цветовые различия.

В виду того, что обновление страницы, как правило, вызывает рассеивание внимания у пользователя, в профиле студента смена информации в секторах (переключение между неделями во 2 секторе, появления списка заданий, действия с экспортом данных в 4 секторе и т.д.) происходит асинхронно - без обновления всей страницы. Данное решение направленно на улучшение юзабилити продукта, позволив пользователю более сосредоточенно работать с приложением.

4. Юзабилити Тестирование

Для того, чтобы проверить, соответствует ли создаваемое приложение поставленным требованиям простоты и интуитивности пользовательского интерфейса, были проведены юзабилити тесты. Кроме этого, данное тестирование помогло выявить существующие на данный момент проблемы и определить, с чем они связаны. Так как юзабилити тестирование было проведено на начальном этапе создания приложения, выявленные проблемы не представляют большой опасности и могут быть решены без чрезмерных затрат времени.

Проведённое тестирование состояло из трёх этапов. Первым этапом был набор целевой группы тестировщиков и предоставления им возможности ознакомления с приложением. После ознакомления с приложением, был проведён второй этап, тестировщики должны были пройти опрос, содержащий вопросы о пользовательском интерфейсе. Третьим этапом являлся анализ полученной на основе результатов опроса статистики.

4.1 Первый этап тестирования

Для проведения тестирования была набрана группа людей, знакомых с системой организации учебной деятельности в ТТУ. На момент проведения тестирования не было возможности установить приложение на удалённом сервере и было необходимо проходить первые два этапа с каждым участником вместе, запуская приложение на локальном сервере. Учитывая данный фактор, провести полноценное тестирование удалось с семью людьми. Все семеро человек являлись либо студентами 3-го курса ТТУ, либо закончившими ТТУ в прошлом году.

При проведении юзабилити тестирования, в отличие от проведённого предшествующего созданию приложения опроса, важную информацию можно получить не только выделить общие тенденции, но и обратив внимание на каждый пройденный тест в отдельности. Поэтому количество человек, участвующих в тестировании данного веб-приложения является вполне достаточным для объективного анализа и выделения имеющихся проблем.

Каждому из участников тестирования была дана возможность самостоятельно ознакомиться с имеющимся приложением, а после выполнить несколько поставленных заданий, охватывающих большую часть функциональности приложения (выбрать предметы, соответствующие текущей декларации, изменить какие-либо поля, найти необходимую информацию и так далее). После выполнения данных заданий было необходимо заполнить опрос, включающий вопросы о пользовательском интерфейсе.

4.2 Второй этап тестирования

Для проведения опроса и анализа статистических данных, был использован веб-сервис AttrakDiff. Данный сервис позволяет определить, как пользователи оценивают удобство и дизайн тестируемого продукта. Для проведения тестирования был использован один из предлагаемых сервисом сценариев использования – Single Evaluation. Данный сценарий позволяет получить оценки удобства и дизайна приложения в конкретный момент времени. Предлагая тестировщикам пройти опрос один раз, данный метод проецирует оценки тестируемого приложения на график гедонистического качества и прагматического качества. Проецирование оценок позволяет определить общую привлекательность продукта.

Показатель прагматического качества описывает юзабилити продукта, а также указывает, насколько успешно пользователи справляются с поставленными задачами, используя тестируемый продукт.

Показатель гедонистического качества описывает, как хорошо продукт способен поддерживать нужды пользователя в новых, интересных функциях и насколько хорошо продукт справляется с задачей получения пользователем удовольствия при использовании данного продукта. В целом, данный показатель определяет, насколько продукт является интересным, приятным, мотивирующим и приносящим положительный опыт при использовании для пользователя.

Показатели гедонистического и прагматического качества независимы друг от друга, однако используются вместе и в равной степени для определения общего уровня привлекательности продукта.

Предлагаемый методом Single evaluation опрос состоит из 28 пунктов, не имеющих прямой связи с конкретным приложением, однако позволяющих пользователям описать

продукт. Данные пункты являются парой прилагательное-антоним прилагательного, а пользователю предлагается высказать личное мнение, выбрав из данной пары, по имеющейся шкале, наиболее подходящее для описания тестируемого приложения слово. Шкала является своего рода отрезком от слова к его антониму. Ниже представлена часть задаваемых вопросов. Со всеми вопросами можно ознакомиться в дополнении.

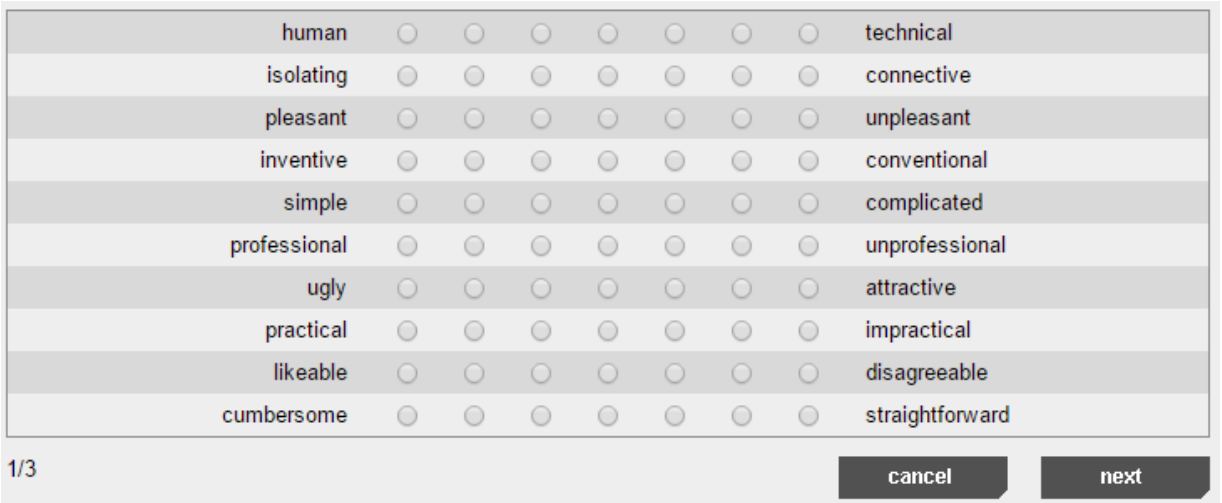


Рисунок 18. Часть вопросов опроса, проводимого при тестировании.

4.3 Третий этап тестирования

Веб-сервис AttrakDiff, кроме создания опроса, позволяет сгенерировать статистические данные, основанные на оценках продукта пользователями.

Первой частью статистики является построенный график гедонистического и прагматического качества. На графике отмечается точка P, показывающая уровень качества продукта, учитывая средние значения оценок пользователей.

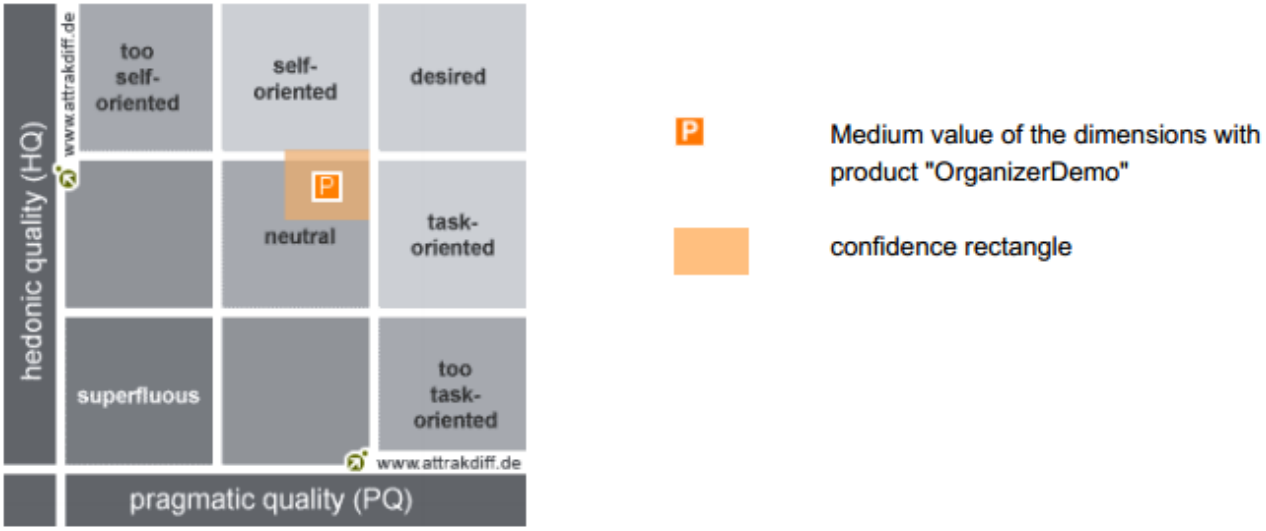


Рисунок 19. График гедонистического и прагматического качества проведённого опроса.

На представленном выше графике осью X является уровень прагматического качества, а осью Y – гедонистического. Две данные оси независимы друг от друга, однако обе должны быть на высоком уровне, чтобы продукт можно было считать качественным. Соответственно, продукт должен стремиться к отмеченному на графике квадрату «desired». Также на графике отмечается так называемый прямоугольник доверительности – confidence rectangle, который показывает, насколько ответы пользователей надёжны. Данная надёжности определяется степенью схожести ответов пользователей. Соответственно, чем чаще мнения пользователей расходились, тем больше становился данный прямоугольник. Точка уровня качества продукта P может колебаться в плоскости прямоугольника доверительности, поэтому можно сказать, что размер прямоугольника прямо пропорционален объективности опроса.

Как мы видим на рисунке 19, график свидетельствует о том, что общий уровень качества тестируемого продукта, учитывая прямоугольник доверительности, выше среднего, однако полученный результат всё же недостаточно высок и не может быть квалифицирован, как желаемый. Это свидетельствует о том, что в целом, пользовательский интерфейс тестируемого приложения является хорошим, однако в нём присутствуют проблемы.

Полученный уровень прагматического качества показывает, что на данном этапе пользователь в некоторой степени взаимодействует с продуктом, однако, для получения желаемого результата, данный уровень необходимо повысить. Уровень

гедонистического качества свидетельствует о том, что данный продукт стимулирует пользователя и в целом приносит положительный опыт использования, однако, также требует улучшения.

Каждый из задаваемых вопросов относился к одной из 4-х категорий.

- **PQ** – прагматическое качество
- **HQ-I** – гедонистическое качество(личность)
- **HQ-S** – гедонистическое качество (стимуляция)
- **ATT** – общая привлекательность продукта

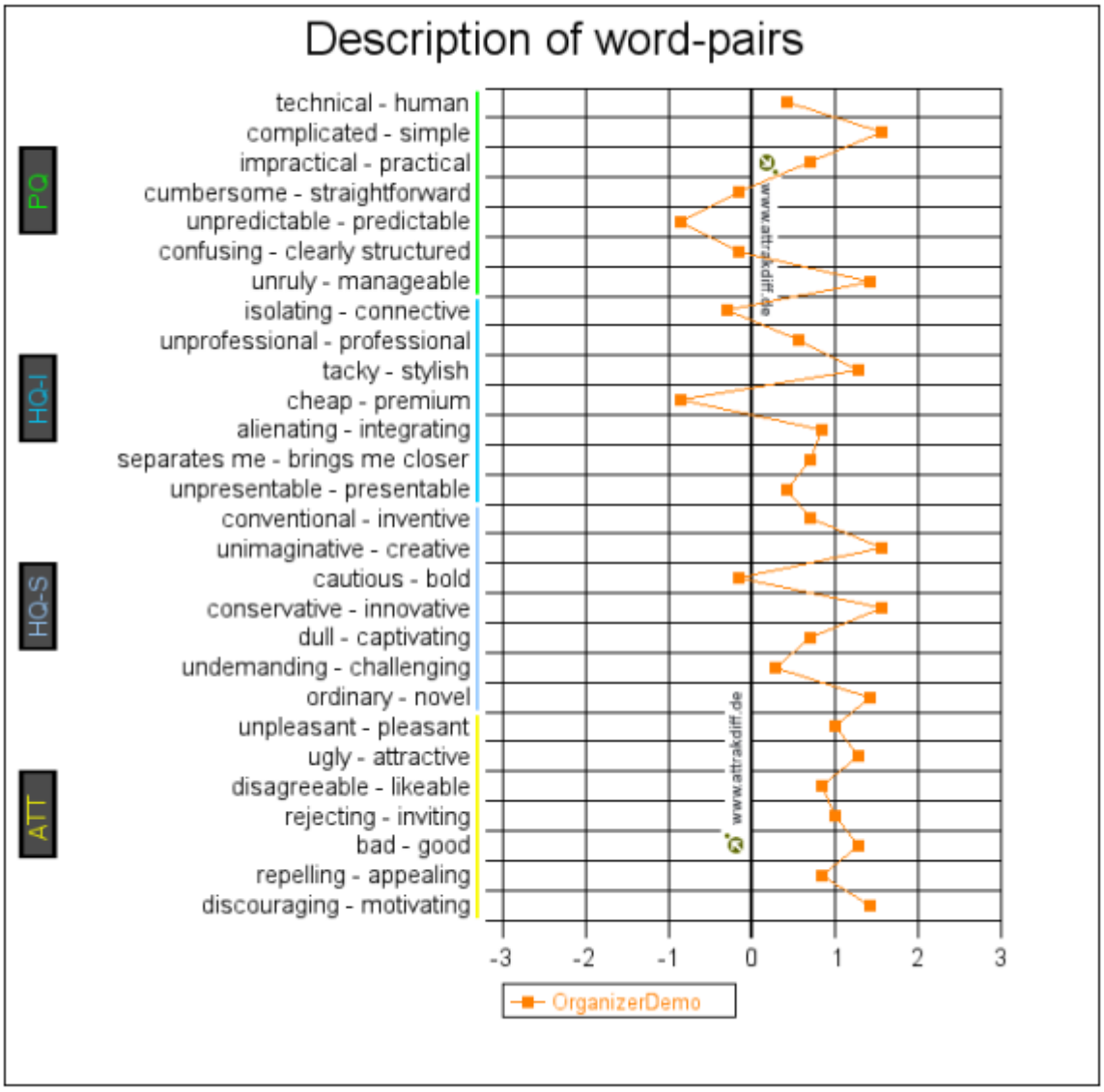


Рисунок 20. Общая статистика ответов на каждый из пунктов опроса тестируемого приложения.

С помощью представленного на рисунке 20 графика можно детально проанализировать ответы пользователей и выделить имеющиеся сильные и слабые стороны пользовательского интерфейса. Находящие на правой стороне графика точки означают, что из пары прилагательное-антоним, пользователями было выбрано слово, которое можно считать плюсом приложения. Как можно заметить, большинство пунктов находятся на правой стороне графика, что свидетельствует о наличии у приложения положительных сторон, однако по шкале от 0 до 3, ни один из пунктов не пересекает 2. Это означает, что существует достаточно большое поле для деятельности в направлении улучшения интерфейса.

Расположенные на левой части графика точки описывают проблемы приложения. Как мы видим, проблемы имеются как в категории прагматического качества, так и в категории гедонистического качества. Наиболее критичными являются пункты unpredictable-predictable и cheap-premium.

Наиболее показательным является имеющий отрицательное значение пункт unpredictable-predictable. Он означает, что пользователи находят поведение приложения и его структуру непредсказуемым. Как можно заметить, отрицательное значение также имеют пункты confusing-clearly structured и cumbersome-straightforward, а это значит, что структура приложения в целом вызывает трудности понимания. Одной из возможных причин является отсутствие на данном этапе хорошо продуманной системы навигации между ресурсами приложения. Данная проблема дезориентирует пользователя при работе с приложением, в следствии чего, без опыта работы с системой возникает путаница и непредсказуемость действий системы. Также в альфа-версии отсутствуют сообщения о успешно выполненных действиях или ошибках, что также привносит трудности в использовании и понимании системы. Решением данных проблем является создание понятной пользователю навигационной системы между ресурсами и добавление сообщений о действиях совершённых пользователем и ошибках в них. Также при работе с приложением пользователи выделяли, что использование графических изображений для описания действий не всегда понятно, поэтому в будущем необходимо дополнить данные действия текстовым описанием.

5. Планы на будущее

Имеющаяся на данный момент версия приложения несомненно требует доработки. В ближайшей перспективе планируется реализовать отсутствующую на данный момент функциональность необходимую для выполнения поставленных требований и исправить выявленные в ходе тестирования ошибки.

Наиболее важным на данный момент является создание решения сложности в доступе к необходимым для учёбы ресурсам. Данная проблема будет решена путём добавления возможности сохранения файлов при создании предмета и просмотр данных файлов при выборе предмета. После реализации данной части и исправления некоторых недочётов, планируется устроить более масштабное тестирование приложения, разместив его на общедоступном сервере, с большим количеством пользователей, с целью получения более подробной информации о недочётах функциональности и юзабилити.

Следующим шагом планируется исправление ошибок, полученных в ходе тестирования и вместе с тем – добавление необходимой функциональности – обеспечение возможности студентам самим добавлять новые предметы, обеспечение мультиязычности, а также добавление полезной для студента функциональности в его профиле.

После исправления всех полученных ошибок и добавления необходимой функциональности, планируется запуск бета-версии на общедоступном сервере с системой обратной связи, позволяющей получать информацию о нарушении работы приложения и своевременно их исправлять.

6. Заключение

Основной целью дипломной работы было создание рабочей версии веб-приложения, направленного на решение проблем организации учебной деятельности для студентов Таллиннского Технического Университета.

Среди студентов был проведён опрос, позволивший выявить данные проблемы и описать необходимые для создания приложения требования. Опираясь на полученные ответы, была создана первая рабочая версия веб-приложения, решающая большинство выявленных проблем и позволяющая преподавателям создавать предметы с условиями их сдачи, а студентам составлять список сдаваемых предметов, следить за своими результатами, видеть важную информацию о сроках и условиях сдачи заданий взятых предметов, экспортировать времена заданий в календарь. Кроме того, созданный пользовательский интерфейс был протестирован пользователями, что позволило выявить имеющиеся проблемы.

Проделанная работа показала, что на данный момент проблемы с организацией учебной деятельности являются актуальными среди студентов и их можно решить при помощи веб-приложения. Также в ходе работы была создана версия приложения, которая будет использоваться в качестве фундамента для создания полноценного приложения в будущем.

Используя знания, полученные во время обучения в университете, в ходе работы были достигнуты все поставленные цели.

Kokkuvõte

Lõputöö põhiliseks eesmärgiks oli veebi rakenduse esimese versiooni loomine, mis aitaks lahendada probleemid seotud õppetegevuse organiseerimisega, millega kokkupuutuvad Tallinna Tehnika Ülikooli tudengid.

Tudengite seas oli läbiviidud küsitlus, et neid probleeme tuvastada ja kirjeldada rakenduse loomiseks vajalikud nõudmised. Saadud vastuste põhjal oli loodud esimene töötav veebirakenduse versioon, mis lahendab enamikke tuvastatud probleeme. Õppejõud saaks selle rakenduse abil luua õppeained nende sooritamise tingimustega. Tudengitel võimaldab see rakendus looma sooritatavate ainete nimekirja, jälgima oma tulemusi, nägema infot võetud ainete tähtaja ja tingimuste kohta ning eksportima need ajad kalendrisse.

Tehtud töö näitas, et praegu probleemid mis on seotud õppetöö organiseerimisega on aktuaalsed tudengite seas ning neid on võimalik lahendada loodud veebirakenduse abil. Töö käigus loodud veebirakenduse versioon on plaanis kasutada tulevikus täieliku versiooni loomiseks.

Ülikooli õppimise ajal saadud teadmised olid kasutatud töö käigus et saavutada kõike pandud eesmärke.

Summary

The main goal of this bachelor`s thesis was to create functioning web application that would solve organisational problems which students of Tallinn University of Technology are facing while managing their studies.

To reveal these problems and understand the needed requirements for the future web application the students were given a questionnaire. Based on the received answers, first version of web application was created which solved most of the identified problems. This alpha-version lets teachers create subjects with requirements to pass it and students to compose the list of subjects to pass, to follow their results, to see important information about deadlines and pass-requirements of subjects and also to export important dates to the calendar. What is more, created user interface was tested by users which helped to reveal problems.

Carried out work showed that in the present there are problems with management of learning activity which are topical for students and could be solved with a help of created web application. The current version of the application is fundamental base for further development of the fully functioning application in the future.

All of the goals set in this thesis were achieved with a help of applying knowledge acquired in the university.

Список использованной литературы

1. **Roy Thomas Fielding.** Architectural Styles and the Design of Network-based Software Architectures. [В Интернете]. [Использовано: 27.10.2014]. http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

2. The Tech Dictionary and IT Encyclopedia. [В Интернете]. [Использовано: 04.01.2015] http://whatis.techtarget.com/definition/model-view-controller-MVC

3. **Norman Walsh.** A Technical Introduction to XML. [В Интернете]. [Использовано: 04.01.2015] http://www.xml.com/pub/a/98/10/guide0.html?page=2

4. Object-relation mapping. [В интернете]. [Использовано: 04.01.2015]. https://ru.wikipedia.org/wiki/ORM

5. Официальная домашняя страница JSON. [В Интернете]. [Использовано 04.01.2015]. http://json.org/json-ru.html

6. **Daniel Nations.** What is a Web Application. [В Интернете]. [Использовано 04.01.2015]. http://webtrends.about.com/od/webapplications/a/web_application.htm

7. DocForge - Software Development Resources. [В Интернете]. [Использовано 04.01.2015]. http://docforge.com/wiki/Web_application_framework

8. **Vladimir Zwass.** Information system [В Интернете]. [Использовано 04.01.2015]. http://global.britannica.com/EBchecked/topic/287895/information-system

9. **Margaret Rouse.** SQL. [В Интернете]. [Использовано 04.01.2015]. Опубликовано на http://searchsqlserver.techtarget.com/definition/SQL

10. Interpreted language. [В интернете]. [Использовано: 04.01.2015]. https://en.wikipedia.org/wiki/Interpreted_language

11. Compiled language. [В интернете]. [Использовано: 04.01.2015]. https://en.wikipedia.org/wiki/Compiled_language

12. **Margaret Rouse.** HTML. [В Интернете]. [Использовано 04.01.2015]. Опубликовано на http://searchsoa.techtarget.com/definition/HTML

13. Пользовательский Интерфейс. [В интернете]. [Использовано: 04.01.2015]. http://www.rae.ru/monographs/141-4635

14. AJAX. [В интернете]. [Использовано: 04.01.2015]. https://ru.wikipedia.org/wiki/AJAX

15. **Margaret Rouse.** URL [В Интернете]. [Использовано 04.01.2015] **.** Опубликовано на http://searchsoa.techtarget.com/definition/URL

16. **Jakob Nielsen.** Usability 101: Introduction to Usability. URL [В Интернете]. [Использовано 04.01.2015]. http://www.nngroup.com/articles/usability-101-introduction-to-usability/

17. Pragmatic and hedonic quality of software systems. [В Интернете]. [Использовано 04.01.2015]. Опубликовано на http://www.flamelab.de/article/pragmatic-and-hedonic-quality-of-software-systems/

18. **Raul Liivrand.** Süsteemi- ja rakendusarhitektuur [В Интернете]. [Использовано 12.10.2014]. http://maurus.ttu.ee/ained/IDU0200_2014/doc/31/SissejuhatusRakArh_I.pdf

19. **Raul Liivrand.** Sissejuhatus rakenduse arhitektuuri [В Интернете]. [Использовано 12.10.2014]. http://maurus.ttu.ee/ained/IDU0200_2014/doc/31/SissejuhatusRakArh_I.pdf

20. **Jakub Linowski.** «A Good User Interface». [В Интернете]. [Использовано 20.12.2014]. http://www.goodui.org/

21. Usability First – Usability Glossary. Hedonic quality. [В Интернете]. [Использовано 04.01.2015]. http://www.usabilityfirst.com/glossary/hedonic-quality/

22. **Ben Alex , Luke Taylor , Rob Winch.** Spring Security Reference. [В Интернете]. [Использовано 17.10.2014]. http://docs.spring.io/spring-security/site/docs/4.0.0.RC1/reference/htmlsingle/

23. Spring Framework Reference Documentation. [В Интернете]. [Использовано 06.10.2014]. http://docs.spring.io/spring/docs/4.2.0.BUILD-SNAPSHOT/spring-framework-reference/htmlsingle/

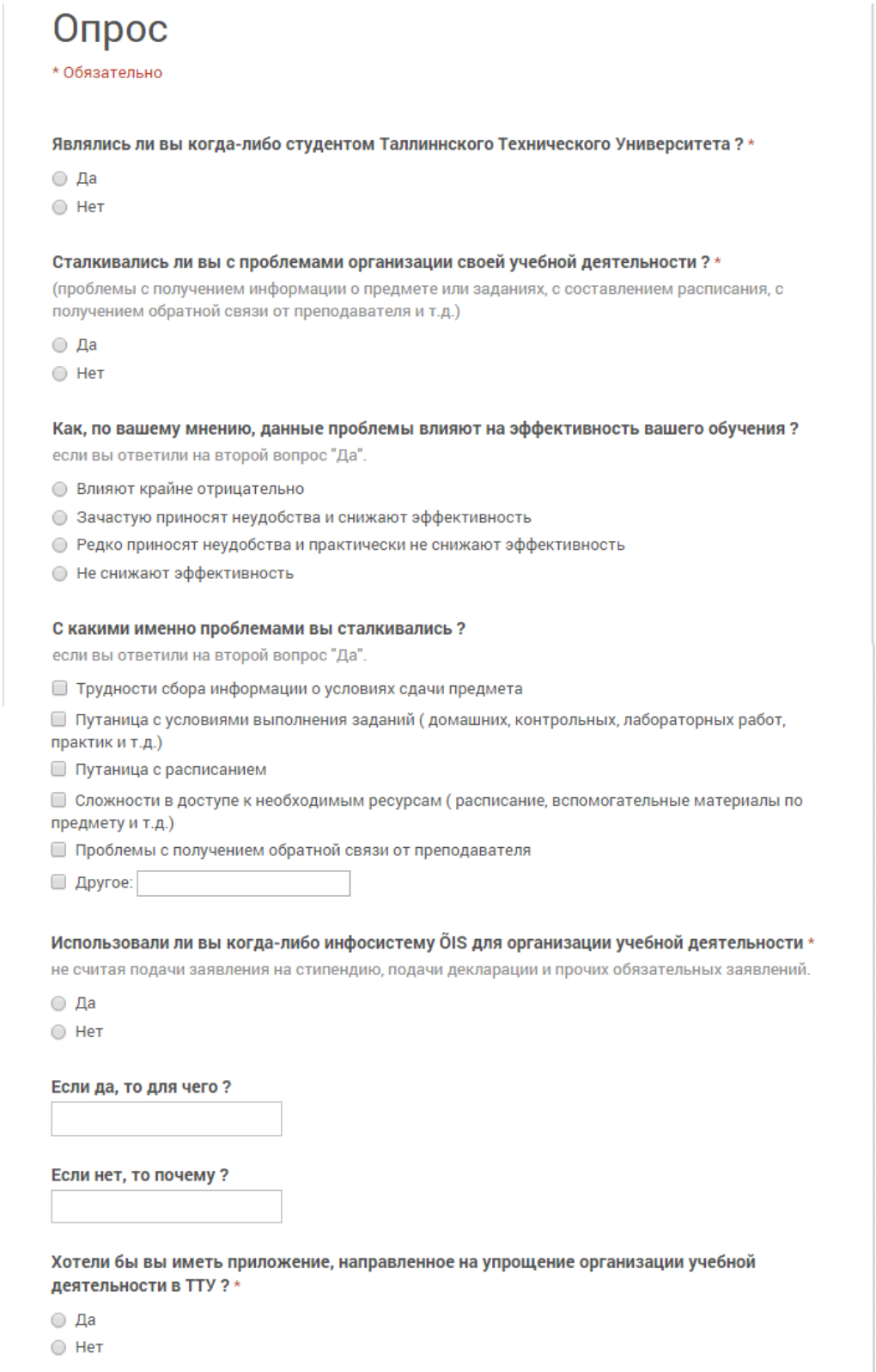
24. Tallinna Tehnikaülikool. [В Интернете]. [Использовано 01.11.2014]. http://www.ttu.ee/

25. Hibernate Reference Documentation . [В Интернете]. [Использовано 12.10.2014]. http://docs.jboss.org/hibernate/orm/4.3/manual/en-US/html/

26. **Marc Hassenzahl.** Hedonic, emotional and experiential perspectives on product quality. [В Интернете]. [Использовано 20.12.2014]. http://www.irma-international.org/viewtitle/13133/

27. **Marc Hassenzahl, Noam Tractinsky.** User experience – a research agenda. [В Интернете]. [Использовано 20.12.2014]. https://ccrma.stanford.edu/~sleitman/UserExperienceAResearchAgenda.pdf

Дополнение



Опрос

* Обязательно

Являлись ли вы когда-либо студентом Таллиннского Технического Университета ? *

- Да
- Нет

Сталкивались ли вы с проблемами организации своей учебной деятельности ? *

(проблемы с получением информации о предмете или заданиях, с составлением расписания, с получением обратной связи от преподавателя и т.д.)

- Да
- Нет

Как, по вашему мнению, данные проблемы влияют на эффективность вашего обучения ?

если вы ответили на второй вопрос "Да".

- Влияют крайне отрицательно
- Зачастую приносят неудобства и снижают эффективность
- Редко приносят неудобства и практически не снижают эффективность
- Не снижают эффективность

С какими именно проблемами вы сталкивались ?

если вы ответили на второй вопрос "Да".

- Трудности сбора информации о условиях сдачи предмета
- Путаница с условиями выполнения заданий (домашних, контрольных, лабораторных работ, практик и т.д.)
- Путаница с расписанием
- Сложности в доступе к необходимым ресурсам (расписание, вспомогательные материалы по предмету и т.д.)
- Проблемы с получением обратной связи от преподавателя
- Другое:

Использовали ли вы когда-либо инфосистему ÕIS для организации учебной деятельности *

не считая подачи заявления на стипендию, подачи декларации и прочих обязательных заявлений.

- Да
- Нет

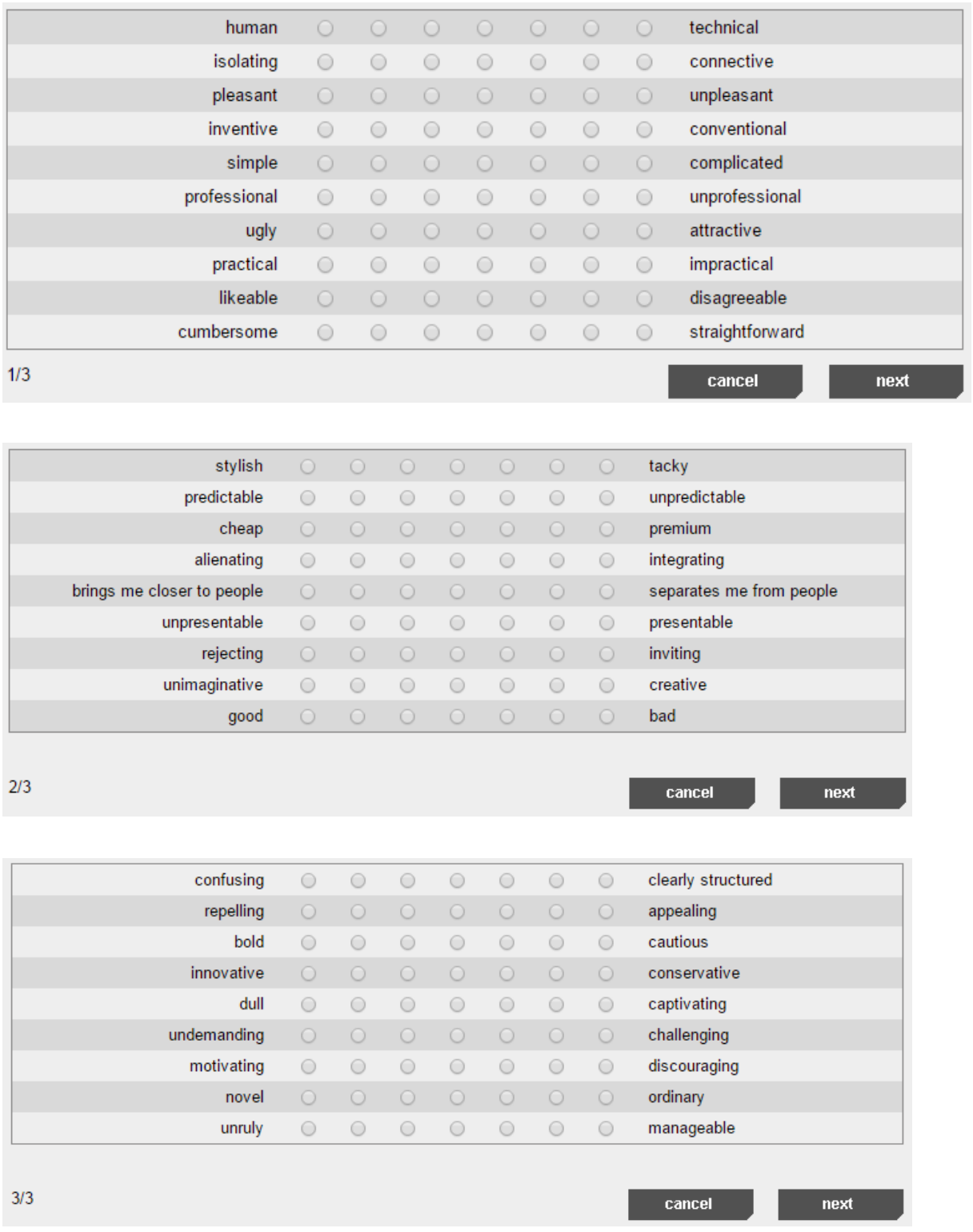
Если да, то для чего ?

Если нет, то почему ?

Хотели бы вы иметь приложение, направленное на упрощение организации учебной деятельности в ТТУ ? *

- Да
- Нет

Дополнительный рисунок 1. Вопросы проводимого опроса



Дополнительный рисунок 2. Пункты проводимого опроса юзабилити-тестирования.