ПЕТРОЗАВОДСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
КАФЕДРА ИНФОРМАТИКИ И МАТЕМАТИЧЕСКОГО ОБЕСПЕЧЕНИЯ

Отчёт по дисциплине
«Верификация программного обеспечения»

Выполнил:
студент группы 22609 Д. С. Пятин
Лектор:
к.т.н., доцент К. А. Кулаков

Петрозаводск
2016

Содержание

Выбор и согласование тематики проекта [к содержанию]

Проект представляет собой мобильное приложение для просмотра расписания занятий ПетрГУ, студентами с помощью мобильного телефона, созданное с помощью фреймворка Ionic. Данный фреймворк является надстройкой для Apache Cordova, позволяющей использование Angular JS первой версии для построение каркаса приложения [1]. Приложении позволяет просматривать расписание занятий студенческих групп, преподавателей и аудиторий. В данном проекте для тестирования представлена только версия приложения для системы Android.

Ссылка на приложение:
https://play.google.com/store/apps/details?id=com.ionicframework.myapp259509

Описание функциональности [к содержанию]

Заявляется и будет протестирована следующая функциональность:

- Просмотр расписания занятий студенческой группы.
- Просмотр расписания занятий в выбранной аудитории.
- Просмотр расписания занятий преподавателя.
- Отображение местоположения корпуса университета где идет занятие на карте.

Ограничения проекта [к содержанию]

Вышеперечисленные функции должны быть доступны для мобильных телефонов с операционной системой Android версии 4.4.0 и старше.

Приложение должно быть размещено в магазине Google play [2].

Обоснованным критерием данного ограничения является соотношение затрат на поддержку старых версий ОС Android с потенциальным увеличением количества пользователей приложения [3].

Описание вариантов использования приложения [к содержанию]

Программа предназначена для просмотра расписания студенческих групп, преподавателей и аудиторий с мобильного устройства (преимущественно смартфона) под операционной системой Android. Соответственно можно выделить следующие основные варианты использования данного приложения:

- Пользователь с помощью мобильного приложения с операционной системой Android просматривает свое расписание занятий.
- Пользователь просматривает расписание занятий преподавателя, чтобы определить время и место его нахождений.

- Пользователь просматривает расписание выбранной аудитории, например - компьютерного класса, чтобы определить, когда он свободен.

Архитектура системы [к содержанию]

Приложение (здесь и далее синоним понятия ПО и система) является web-ориентированным (web view) мобильным приложением, созданным с помощью фреймворка Ionic. Ionic позволяет использовать подход фреймворка Angular JS первой версии для создания мобильных приложений вместе с Apache Cordova.

Выбор способа создания мобильного приложения обоснован следующими показателями:

Выделяют два основных способа созданий мобильных приложений:

- Native приложения
- Гибридные (Web View) приложения
    - Платформа (Apache Cordova, Steroids, Polymer, Ionic, PhoneGap)
    - UI Framework (Topcoat, Framework7, Ionic, Bootstrap)
- Преимущества Web View
    - Использование не зависящий от мобильных платформ технологий: JS, Angular, HTML, CSS\SASS
    - Скорость разработки (при хорошем знании framework’a)
- Недостатки Web View
    - Слабая привязанность к отдельным платформам (максимум схожий внешний вид)
    - Отображение зависит от поддержки CSS на соответствующей платформе
    - Скорость работы на слабых аппаратах

При имеющихся ресурсах и ограничений проекта преимущества Web View приложений многократно перевесили их недостатки.

Обоснованностью выбора фреймворка Ionic являются следующие показатели [4]:

web view это:

- Стремительно развивающийся фреймворк на основе Apache Cordova, Angular JS
- В компанию разработчика уже инвестировано около 4 млн $.
- Для iOS и Android ~ 90% рынка
- Много виджетов.
- Оптимизация (среди web view приложений)

Архитектуру системы принято делить на логическую и физическую.

Логическая архитектура системы является слоевой и состоит из трех слоев (здесь и далее синоним понятия подсистема):

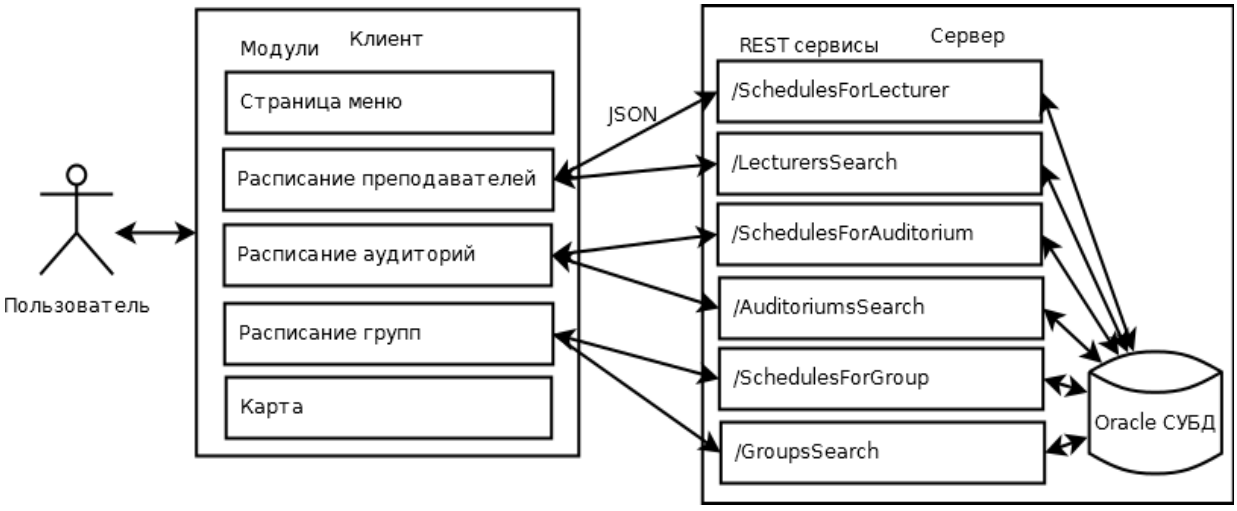
- Пользовательский интерфейс (слой представления)
- Бизнес-логика
- Слой данных

Физическая архитектура системы состоит из 2х частей:

Мобильный клиент (сочетает пользовательский интерфейс и часть бизнес-логики)

Oracle APEX Rest сервисы, сочетающие часть бизнес-логики, связанной с предобработкой данных и доступом к системе данных университета (Oracle Database). Форматом передачи данных от сервисов к клиенту является JSON.В случае информационно-ориентированных приложений использование JSON предпочтительнее XML, в силу его простоты и легкости обработки данных на стороне клиента. XML может быть незаменимым на сервере, но с JSON определенно проще работать на клиенте.

Общая высокоуровневая диаграмма системы:



Пользователь системы – отдельная сущность при проектировании. В контексте данного проекта пользователь системы – это человек устанавливающий, запускающий и использующий функции приложения на мобильном устройстве с операционной системой Android версии >= 4.4.0.

Перечень модулей и функциональных элементов [к содержанию]

1. **Модули клиентской части**

   Модуль «Расписание аудиторий»

   Файлы: auditoriums_schedule.html, auditoriumsController.js
   Высокоуровневое описание функций:

   - Ввод номера\части номера аудитории в поле, вывод аудиторий, подходящих под шаблон ввода, выбор аудитории из представленных, загрузка расписания для выбранной аудитории.
   - Кнопка перехода назад в главное меню.
   - По нажатию на аудиторию в расписании переход на страницу карты с отображением соответствующего корпуса.

   Модель данных представления:

```
    $scope.data = {
    "auditoriums" : [],
    "schedules":[],
    "search" : { "template": '', "obj":{}},
    "days": dataService.getDays(),
    "currentDay": dataService.getCurrentDay(),
    "pairs": dataService.getPairs()};
```

   dataService является вспомогательным локальным (на клиенте) сервисом для выдачи «захардкоженых» данных, дней недели, номеров пар и т. д, которые по определенным причинам еще не представлены в базе данных. (т. е. это своего рода заглушки (dummies или mocks) для данных).

   Низкоуровневое описание функций:

   - searchAuditoriums(); поиск по data.search.template. Выполняется AJAX get запрос к сервису AuditoriumsSearch и результат приходит в data.auditoriums.
   - getSchedulesForAuditorium(auditorium); загрузка расписания по auditorium.num, auditorium.buildingShortName. Выполняется

AJAX get запрос к сервису SchedulesForAuditorium и результат приходит в data.schedules.

Модуль «Расписание групп»
Файлы: groups_schedule.html, groupsController.js
Высокоуровневое описание функций:

- Ввод номера\части номера группы в поле, вывод групп, подходящих под шаблон ввода, выбор группы из представленных, загрузка расписания для выбранной группы.
- Кнопка перехода назад в главное меню.
- По нажатию на аудиторию в расписании переход на страницу карты с отображением соответствующего корпуса.

Модель данных:

```
$scope.data = {
"groups" : [],
"schedules":[],
"search" : { "template": '', "obj":{}},
"days": dataService.getDays(),
"currentDay": dataService.getCurrentDay(),
"pairs": dataService.getPairs()};
```

Низкоуровневое описание функций:

- searchGroups(); поиск по data.search.template. Выполняется AJAX get запрос к сервису GroupsSearch и результат приходит в data.groups.
- getSchedulesForGroup(group); загрузка расписания по group.code, group.specialityCode. Выполняется AJAX get запрос к сервису SchedulesForGroup и результат приходит в data.schedules.

Модуль «Расписание преподавателей»
Файлы: lecturers_schedule.html, lecturersController.js
Высокоуровневое описание функций:

- Ввод фамилии\части фамилии преподавателя в поле, вывод преподавателей, подходящих под шаблон ввода, выбор преподавателя из представленных, загрузка расписания для выбранного преподавателя.
- Кнопка перехода назад в главное меню.
- По нажатию на аудиторию в расписании переход на страницу карты с отображением соответствующего корпуса.

Модель данных:

```
$scope.data = {
"lecturers" : [],
"schedules":[],
```

```
"search" : { "template": '', "obj":{}},
"days": dataService.getDays(),
"currentDay": dataService.getCurrentDay(),
"pairs": dataService.getPairs()};
```

Низкоуровневое описание функций:

- searchLecturers(); поиск по data.search.template. Выполняется AJAX get запрос к сервису LecturersSearch и результат приходит в data.lecturers.
- getSchedulesForLecturer(lecturer); загрузка расписания по lecturer.fullnames. Выполняется AJAX get запрос к сервису SchedulesForLecturer и результат приходит в data.schedules.

Модуль «Страница меню»
Файлы: home.html, homeController.js

Функции: переход в соответствующие разделы приложения.

- Переход в раздел «Расписание групп»
- Переход в раздел «Расписание преподавателей»
- Переход в раздел «Расписание аудиторий»

Модуль «Карта»
*Файлы:*map.html, mapController.js
Высокоуровневое описание функций:

- Отображение карты Google Maps с меткой адреса корпуса, выбранной аудитории.
- Кнопка перехода назад к просмотру расписания.

Модель данных:

```
$scope.mapHeight = screen.height-44 + "px";
$scope.mapWidth = screen.width + "px";
$scope.geocodingAddress = geocodingAddress;
```

Низкоуровневое описание функций:
Initialize(); инициализирует карту google map с новым геокодером, соответствующим geocodingAddress, задает размеры карты по экрану устройства (screen – системная глобальная переменная).

Общие функции клиента
Файлы: app.js
Данные:

- geocodingAddress - глобальная переменная для передачи адреса корпуса между несколькими страницами приложения.

Низкоуровневые функции:

- loadGMaps(); загрузка\инициализация google maps.
- isOnline(); проверка подключения к сети.
- onNetworkError(code); действия при ошибке сети.
- isMobile(); проверка того что устройство мобильный телефон на Android.
- angular.module
  .run({инициализация приложения, загрузка googlemaps (loadGMaps))})
  .config({настройка роутинга});

Логика работы функции кнопки «назад» на всех страницах реализуется штатными средствами Ionic Framework, т. е. неявным javascript, срытым в недрах библиотеки. От разработчика требуется лишь добавить html разметку кнопки в нужное место. Это накладывает ограничение на использование только штатного роутинга ($stateProvider) между страницами.

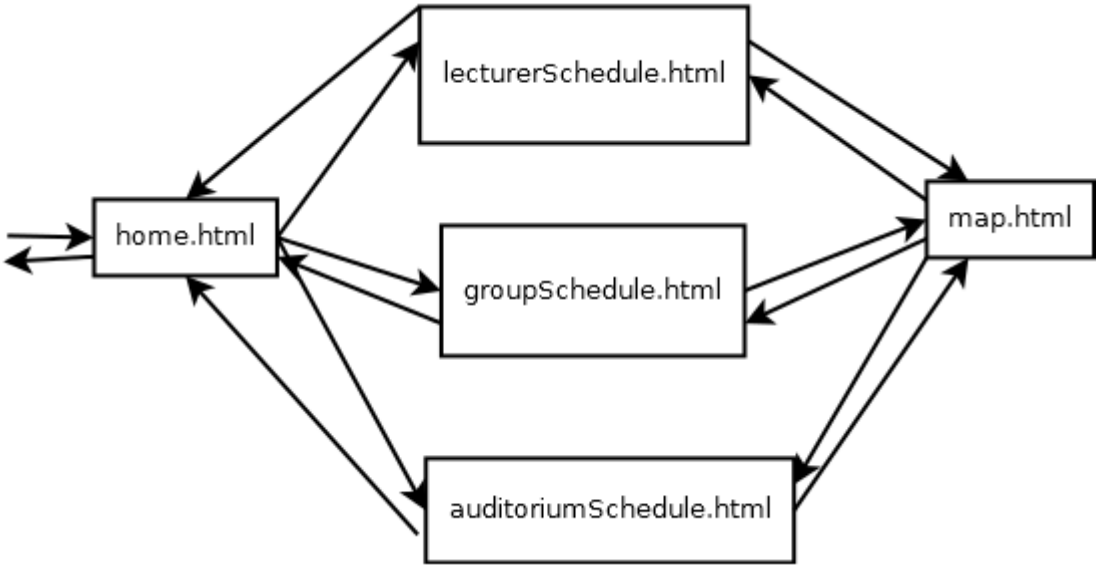
Отдельного упоминания заслуживает dataService. Это локальный сервис данных, находящийся в файле services.js является вспомогательным локальным (на клиенте) сервисом для выдачи захардкоженых данных, дней недели, номеров пар и т. д, которые по определенным причинам еще не представлены в базе данных. (т. е. это своего рода заглушки (dummies или mocks) для данных при модульном тестировании бизнес-логики клиента).

Настройки сборки клиента

- config.xml

Также имеется раздел для дополнительных библиотек, изображений (например - логотипа) и таблицы стилей.

Для будущего тестирования приложения важна диаграмма переходов между страницами приложения:



2. Модули серверной части

Модулями серверной части являются **сервисы**, которые располагаются на удаленном сервере, каждый сервис можно считать отдельным модулем так как он выполняет свою единственную функцию возврата и предобработки соответствующих данных.

Список сервисов:

- SchedulesForGroup
  Выдача из БД расписания для группы по номеру группы и идентификатору специальности.

- SchedulesForAuditorium
  Выдача из БД расписания для аудитории по номеру аудитории и идентификатору корпуса.

- SchedulesForLecturer
  Выдача из БД расписания для преподавателя по идентификатору.

- LecturersSearch
  Выдача из БД списка преподавателей по шаблону фамилии.

- GroupsSearch
  Выдача из БД списка групп по шаблону номера группы.

- AuditoriumsSearch
  Выдача из БД списка аудиторий по шаблону номера аудитории

Вышеприведенная спецификация архитектуры проекта успешно протестирована и утверждена, при дальнейшем тестировании ее можно считать верной. **Ошибкой необходимо считать несоответствие поведения приложения данной спецификации.**

План тестирования и интеграции [к содержанию]

Предлагается и утвержден следующий план тестирования системы отражающий подход к тестированию «снизу-вверх» или восходящее тестирование:

Сначала **модульно** (отдельными страницами) тестируется пользовательский интерфейс и бизнес-логика представления данных на клиенте, при этом данные поставляются из заглушки dataService. Все это выполняется в браузере затем в эмуляторе, затем и на телефоне.

Цель модульного тестирование отдельных страниц - проверка работы функциональности каждой отдельной страницы приложения.

Это тестирование делится на 2 этапа:

- На первом этапе тестируется работа каждой отдельной функции контроллера.
  Цель этапа: удостовериться в том, что поведение отдельных функций ожидаемо их спецификации.
- На втором этапе тестируется работа всей страницы и контроллера.
  Цель этапа: удостовериться в том, что отображение и логика работы страницы ожидаема ее спецификации.

Параллельно с этим **модульно тестируются** сервисы, своеобразной заглушкой клиента в этом случае является браузер, тестовые запросы вводятся в адресную строку вручную или с помощью программы чеккера.

Цель модульного тестирования сервисов – проверка соответствия их поведения их спецификации.

После завершения модульного тестирования клиента, интеграции частей клиента и завершения модульного тестирования сервисов, происходит интеграция клиента и сервисов и **интеграционное тестирование.**

Цель интеграционного тестирования – проверка работы приложения с настоящими данными, настоящими задержками и на настоящей системе.

Тестирование производится в браузере, эмуляторе и телефоне.

После завершения интеграционного тестирования начинается **системное тестирование**.

В него входят:

Аттестационное тестирование проводится разработчиками с ручной установкой приложения на телефон перед заказчиком.

На данном этапе проверяется соответствие работы приложения заявленным требованиям и проводятся специальные тесты (например - нагрузочные испытания сервера). **Нагрузочные испытания** сервера проходят с помощью самописного JS-скрипта (базовый пример):

```
function testQuery() {

$http.get('serveraddress/api/SchedulesForGroup?groupCode=22509&specialityCode
=01.04.02', {}).error(function (data, status) {
        console.log('error response');
    }).then(function (response) {
        console.log('success response');
    });
}

var connections = 20;
var timeout = 1000;

for(var i = 0; i < connections; ++i){
    setTimeout(testQuery, timeout*i);
}
```

Тестировщик корректирует значения переменных connections и timeout, и смотрит время ответа\возможные ошибки в консоли браузера.

Бета –тестирование проводится после успешного завершения альфа-тестирования и выкладывания бета-версии приложения на Google Play Market. На этом этапе приложение тестируется пользователями, которые оставляют замечание по его работе.

Цель бета-тестирования - поиск максимально возможно числа не замеченных ранее ошибок потенциальными пользователями продукта и их скорейшее исправление.

Когда разработчики убедятся, что приложение соответствует их критериям качества (например, таким критерием качества может быть процент положительных отзывов или оценка бета-версии на Play Market) то они объявляют очередную бета-версию **релизом** и начинается этап **сопровождения** продукта, после этого тестирование проводится в соответствии с моделью разработки проекта.

Таким образом план тестирования и интеграции имеет следующий вид:

1. Модульное тестирование.
   Объекты модульного тестирования клиента:
   - Модуль «Расписание групп»
     Функции: *searchGroups, getSchedulesForGroup*
   - Модуль «Расписание аудиторий»
     Функции: *searchAuditoriums, getSchedulesForAuditorium*
   - Модуль «Расписание преподавателей»
     Функции: *searchLecturers, getSchedulesForLecturer*
   - Модуль «Страница меню»
     Функции: переход в соответствующие разделы.
   - Модуль «Карта»
     Функции: *Initialize*
   - Общие функции клиента:
     - *loadGMaps*
     - *isOnline*
     - *onNetworkError*
     - *isMobile*

   Объекты модульного тестирования сервера:
   - Сервис SchedulesForGroup
   - Сервис SchedulesForAuditorium
   - Сервис SchedulesForLecturer
   - Сервис LecturersSearch
   - Сервис GroupsSearch
   - Сервис AuditoriumsSearch

2. Интеграционное тестирование клиента
   Объекты интеграционного тестирования:
   - Связь функции searchGroups модуля «Расписание групп» клиента с сервисом GroupsSearch сервера.
   - Связь функции getSchedulesForGroup модуля «Расписание групп» клиента с сервисом GroupsSearch сервера.
   - Связь функции searchAuditoriums модуля «Расписание аудиторий» клиента с сервисом AuditoriumsSearch сервера.
   - Связь функции getSchedulesForAuditorium модуля «Расписание аудиторий» клиента с сервисом AuditoriumsSearch сервера.
   - Связь функции searchLecturers модуля «Расписание преподавателей» клиента с сервисом LecturersSearch сервера.

- Связь функции getSchedulesForLecturer модуля «Расписание преподавателей» клиента с сервисом LecturersSearch сервера.

3. Системное тестирование (аттестационное и бета-тестирование)
   Объекты аттестационного и бета-тестирования:
   - Работа система в целом (Интегрированная).

Инструменты тестирования:

Для тестирования используются следующие инструменты:

- Тестирование в браузере: GoogleChrome + его отладчик.
- Тестирование в эмуляторе: GenyMotion 2.4.0. (Android > 4.4.0)
- Тестирование на реальном устройстве: Lenovo A530, Android 4.4.0 KitKat.
- Синтаксический анализ JS и HTML кода с помощью IDE WebStorm 10.0.1 от JetBrains.
- Синтаксический анализ SQL кода с помощью Oracle SQL Developer.
- Система контроля версий: Git.
- Самописный JS-скрипт для нагрузочного тестирования.

Инструкция по ручной установке приложения на мобильное устройство при тестировании:

- cd appdirectory
- cordova build --release android
- cd appdirectory\platforms\android\ant-build\
  Копировать apk
- cd C:\Program Files\Java\jdk1.7.0_67\bin
- keytool -genkey -v -keystore my-release-key.keystore -alias alias_name -keyalg RSA -keysize 2048 -validity 10000
- jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore my-release-key.keystore my_schedule-unsigned.apk alias_name
- cd C:\Users\user\AppData\Local\Android\sdk\build-tools\20.0.0
- zipalign -v 4 my_schedule-unsigned.apk my_schedule.apk

Подход к тестированию [к содержанию]

Используется распространенный подход к тестированию «снизу-вверх» или другими словами – восходящее тестирование.

При использовании данного подхода программное обеспечение собирается и тестируется снизу-вверх. Только модули самого нижнего уровня тестируются

изолированно, автономно. Затем тестируются модули, непосредственно вызывающие их, которые тестируются не автономно, а вместе с уже проверенными модулями. И так, пока не будет достигнута вершина. В последнюю очередь тестируется программное обеспечение в целом.

Критерии прохождения тестов [к содержанию]

Тест считается пройдённым, если конечный результат соответствует ожидаемому результату. Ожидаемый результат - это результат соответствующий ожидаемой работе функции (в соответствии с ее спецификацией).

Ожидаемый результат должен присутствовать в описании каждого теста в плане тестирования.

Критерии приостановления работы [к содержанию]

Тестирование приостанавливается при первой же ошибке. После этого тестировщик должен проанализировать результат и сообщить об ошибке, либо признать новый результат корректным (ожидаемым). После этого может быть принято решение об исправлении либо программы, либо результата. После достижения первой ошибки можно продолжить тестирование до следующей ошибки.

Критерии возобновления работы [к содержанию]

После получения заявки на повторное тестирование нужно начать тестирование с самого начала. Такая заявка по идее должна поступать после каждого сообщения об исправлении ошибки. (Регрессионное тестирование).

Требуемая документация [к содержанию]

Необходим отчёт о проведении тестирования, в нём следует указать дату, версию ПО, средства и вообще максимум информации, которая покажется тестировщику полезной.

Необходимое оборудование и ПО [к содержанию]

Для полноценного тестирования требуется современный компьютер (подключенный к сети Интернет) с операционной системой Windows 8.1 Pro и процессором, поддерживающим виртуализацию Intel VT-x. На компьютере должен быть установлен эмулятор системы Android (в данном случае используется GenyMotion [5], как один из наиболее быстрых). Также требуется мобильное устройство с операционной системой Android >= 4.4.0 и подключением к интернету и wifi и 2g\3g.

Описание тестов [к содержанию]

1. Модульные тесты для страниц клиента

Данные тесты выполняются с помощью данных, взятых с локального сервиса dataService, сразу после написания соответствующей страницы и контроллера.

1.1. Модульные тесты для модуля «Расписание групп»

Тест 1.1.1.

Цель: проверка работы функции searchGroups в модуле «Расписание групп».

Тип: общий

Кем и как выполняется: выполняется программистом, сразу после написания функции searchGroups. Мобильное приложение запускается в браузере машины программиста. Обращение к сервису с помощью $http.get заменяется заглушкой dataService.searchGroups();

Входные данные: нет

Результат: список групп из dataService.groups. Можно проверить выводом результата в консоль браузера с помощью console.log.

Тест 1.1.2.

Цель: проверка работы функции getSchedulesForGroups в модуле «Расписание групп».

Тип: общий

Кем и как выполняется: выполняется программистом, сразу после написания функции getSchedulesForGroups. Мобильное приложение запускается в браузере машины программиста. Обращение к сервису с помощью $http.get заменяется заглушкой dataService.getSchedulesFor();

Входные данные: нет

Результат: список занятий из dataService.schedules. Можно проверить выводом результата в консоль браузера с помощью console.log.

Тест 1.1.3

Цель: проверка разметки страницы group_schedule.html и выполнения основных функций интерфейса, предоставляемого этой страницей, а также корректного связывания модели интерфейса с моделью данных.

Тип: общий\UI

Кем и как выполняется: выполняется программистом, сразу после написания разметки в файле group_schedule.html и контроллера groupController.js. Мобильное приложение запускается в браузере машины программиста.

Входные данные: сценарий теста, ввести в поле поиска группы значение “2250” (без кавычек) из предложенного списка групп выбрать группу 22509.

Результат: на странице должно отобразиться расписание из dataService.schedules.

Тест 1.1.4

Цель: проверка разметки страницы group_schedule.html и выполнения основных функций интерфейса, предоставляемого этой страницей, а также корректного связывания модели интерфейса с моделью данных.

Тип: негативный

Кем и как выполняется: выполняется программистом, сразу после написания разметки в файле group_schedule.html и контроллера groupController.js. Мобильное приложение запускается в браузере машины программиста.

Входные данные: сценарий теста, ввести в поле поиска группы значение “ ” (несколько пробелов, без кавычек).

Результат: должен отобразиться пустой список групп для выбора (без всяких [Object] и т.п.)

1.2. Модульные тесты для модуля «Расписание аудиторий»

Тест 1.2.1.

Цель: проверка работы функции searchAuditoriums в модуле «Расписание аудиторий».

Тип: общий

Кем и как выполняется: выполняется программистом, сразу после написания функции searchAuditoriums. Мобильное приложение запускается в браузере машины программиста. Обращение к сервису с помощью $http.get заменяется заглушкой dataService.searchAuditoriums ();

Входные данные: нет

Результат: список аудиторий из dataService.auditoriums. Можно проверить выводом результата в консоль браузера с помощью console.log.

Тест 1.2.2.

Цель: проверка работы функции getSchedulesForAuditoriums в модуле «Расписание аудиторий».

Тип: общий

Кем и как выполняется: выполняется программистом, сразу после написания функции getSchedulesForAuditoriums. Мобильное приложение запускается в браузере машины программиста. Обращение к сервису с помощью $http.get заменяется заглушкой dataService.getSchedulesFor();

Входные данные: нет

Результат: список занятий из dataService.schedules. Можно проверить выводом результата в консоль браузера с помощью console.log.

Тест 1.2.3

Цель: проверка разметки страницы auditorium_schedule.html и выполнения основных функций интерфейса, предоставляемого этой страницей.

Тип: общий\UI

Кем и как выполняется: выполняется программистом, сразу после написания разметки в файле auditorium_schedule.html и контроллера auditoriumController.js. Мобильное приложение запускается в браузере машины программиста.

Входные данные: сценарий теста, ввести в поле поиска аудитории значение “14” (без кавычек) из предложенного списка аудиторий выбрать аудиторию с номером 146 в главном корпусе (ГК).

Результат: на странице должно отобразиться расписание из dataService.schedules.

Тест 1.2.4

Цель: проверка разметки страницы auditorium_schedule.html и выполнения основных функций интерфейса, предоставляемого этой страницей.

Тип: негативный

Кем и как выполняется: выполняется программистом, сразу после написания разметки в файле auditorium_schedule.html и контроллера auditoriumController.js. Мобильное приложение запускается в браузере машины программиста.

Входные данные: сценарий теста, ввести в поле поиска аудитории значение “ ” (несколько пробелов, без кавычек).

Результат: должен отобразиться пустой список аудиторий для выбора (без всяких [Object] и т.п.)

1.3. Модульные тесты для модуля «Расписание преподавателей»

Тест 1.3.1.

Цель: проверка работы функции searchLecturers в модуле «Расписание преподавателей».

Тип: общий

Кем и как выполняется: выполняется программистом, сразу после написания функции searchLecturers. Мобильное приложение запускается в браузере машины программиста. Обращение к сервису с помощью $http.get заменяется заглушкой dataService.searchLecturers ();

Входные данные: нет

Результат: список преподавателей из dataService.lecturers. Можно проверить выводом результата в консоль браузера с помощью console.log.

Тест 1.3.2.

Цель: проверка работы функции getSchedulesForLecturers в модуле «Расписание преподавателей».

Тип: общий

Кем и как выполняется: выполняется программистом, сразу после написания функции getSchedulesForLecturers. Мобильное приложение запускается в браузере машины программиста. Обращение к сервису с помощью $http.get заменяется заглушкой dataService.getSchedulesFor();

Входные данные: нет

Результат: список занятий из dataService.schedules. Можно проверить выводом результата в консоль браузера с помощью console.log.

Тест 1.3.3

Цель: проверка разметки страницы lecturer_schedule.html и выполнения основных функций интерфейса, предоставляемого этой страницей, а также корректного связывания модели интерфейса с моделью данных.

Тип: общий\UI

Кем и как выполняется: выполняется программистом, сразу после написания разметки в файле lecturer_schedule.html и контроллера lecturerController.js. Мобильное приложение запускается в браузере машины программиста.

Входные данные: сценарий теста, ввести в поле поиска преподавателя значение “Кузне” (без кавычек) из предложенного списка преподавателей выбрать преподавателя Кузнецов В. А.

Результат: на странице должно отобразиться расписание из dataService.schedules.

Тест 1.3.4

Цель: проверка разметки страницы lecturer_schedule.html и выполнения основных функций интерфейса, предоставляемого этой страницей, а также корректного связывания модели интерфейса с моделью данных.

Тип: негативный

Кем и как выполняется: выполняется программистом, сразу после написания разметки в файле lecturer_schedule.html и контроллера lecturerController.js. Мобильное приложение запускается в браузере машины программиста.

Входные данные: сценарий теста, ввести в поле поиска преподавателя значение “ ” (несколько пробелов, без кавычек).

Результат: должен отобразиться пустой список преподавателей для выбора (без всяких [Object] и т.п.)

1.4. Модульные тесты для модуля «Карта»

Тест 1.4.1.

Цель: проверка работы функции searchLecturers в модуле «Расписание преподавателей».

Тип: общий

Кем и как выполняется: выполняется программистом, сразу после написания функции searchLecturers. Мобильное приложение запускается в браузере машины программиста. Обращение к сервису с помощью $http.get заменяется заглушкой dataService.searchLecturers ();

Входные данные: нет

Результат: список преподавателей из dataService.lecturers. Можно проверить выводом результата в консоль браузера с помощью console.log.

Тест 1.4.2

Цель: проверка разметки страницы map.html и выполнения основных функций интерфейса, предоставляемого этой страницей.

Тип: общий\UI

Кем и как выполняется: выполняется программистом, сразу после написания разметки в файле map.html и контроллера mapController.js. Мобильное приложение запускается в браузере машины программиста, обязательно с подключением к сети Интернет.

Входные данные: сценарий теста, присвоить переменной geocodingAddress значение из dataService.sampleAddress(); Собрать приложение.

Результат: на странице с картой google map должна отобразиться метка на соответствующем адресе.

Тест 1.4.3

Цель: проверка разметки страницы map.html и выполнения основных функций интерфейса, предоставляемого этой страницей при отключенном соединении с сетью Интернет.

Тип: негативный

Кем и как выполняется: выполняется программистом, сразу после написания разметки в файле map.html и контроллера mapController.js. Мобильное приложение запускается в браузере машины программиста, обязательно с отсутствием подключения к сети Интернет.

Входные данные: нет

Результат: на странице с картой google map должно появиться alert сообщение об отсутствии сети.

1.5. Модульные тесты для модуля «Страница меню»

Тест 1.5.1

Цель: проверка разметки страницы home.html и выполнения основных функций интерфейса, предоставляемого этой страницей.

Тип: общий\UI

Кем и как выполняется: выполняется программистом, сразу после написания разметки в файле home.html и контроллера homeController.js. Мобильное приложение запускается в браузере машины программиста.

Входные данные: нет

Результат: на странице home.js должен выводиться список из подменю:

- Расписание групп
- Расписание преподавателей
- Расписание аудиторий

По щелчку на каждый пункт, должен происходить переход на соответствующую страницу.

1.6. Модульные тесты для общих функций в app.js

Тест 1.6.1.

Цель: проверка работы функции loadGMaps() - общей функции клиента.

Тип: общий

Кем и как выполняется: выполняется программистом, сразу после написания функции loadGMaps(). Мобильное приложение запускается в браузере машины программиста. Требуется подключение к сети Интернет.

Входные данные: нет.

Результат: в консоли браузера не должно быть ошибок, связанных с инициализацией карт google maps.

Тест 1.6.2.

Цель: проверка работы функции loadGMaps()- общей функции клиента, при отсутсвии подключения к сети Интернет.

Тип: негативный

Кем и как выполняется: выполняется программистом, сразу после написания функции loadGMaps(). Мобильное приложение запускается в браузере машины программиста. Требуется отсутствие подключения к сети Интернет.

Входные данные: нет.

Результат: в консоли браузера не должно быть ошибок, связанных с инициализацией карт google maps. Должно появиться alert предупреждение о отсутствии сети, сама карта не должна загружаться.

Тест 1.6.3.

Цель: проверка работы функции isOnline() - общей функции клиента при подключении к сети Интернет через wifi;

Тип: общий

Кем и как выполняется: выполняется программистом, сразу после написания функции isOnline(). Мобильное приложение запускается в браузере машины программиста. Требуется подключение к сети Интернет через wifi.

Входные данные: нет.

Результат: функция isOnline(); вернула статус “wifi” (без кавычек).

Тест 1.6.4.

Цель: проверка работы функции isOnline() - общей функции клиента при подключении к сети Интернет через 3g\2g модем;

Тип: общий

Кем и как выполняется: выполняется программистом, сразу после написания функции isOnline(). Мобильное приложение запускается в

браузере машины программиста. Требуется подключение к сети Интернет через 3g\2g модем.

Входные данные: нет.

Результат: функция isOnline(); вернула статус “2g”\”3g”\ (без кавычек).

Тест 1.6.5.

Цель: проверка работы функции isOnline() - общей функции клиента при подключении к сети Интернет через ethernet;

Тип: общий

Кем и как выполняется: выполняется программистом, сразу после написания функции isOnline(). Мобильное приложение запускается в браузере машины программиста. Требуется подключение к сети Интернет через Ethernet.

Входные данные: нет.

Результат: функция isOnline(); вернула статус “ethernet” (без кавычек).

Тест 1.6.6.

Цель: проверка работы функции onNetworkError(code) - общей функции клиента;

Тип: общий

Кем и как выполняется: выполняется программистом, после написания функции onNetworkError(code).

Входные данные: из app.run.ready производится вызов функции onNetworkError(code).

Результат: alert предупреждение об ошибке сети.

Тест 1.6.7.

Цель: проверка работы функции isMobile() - общей функции клиента;

Тип: общий

Кем и как выполняется: выполняется программистом в браузере рабочей машины и на мобильном устройстве.

Входные данные: из app.run.ready производится вызов функции isMobile();

Результат: true на мобильном устройстве, false в браузере рабочей машины.

2. Модульные тесты для сервисов

2.1. Модульные тесты для сервиса SchedulesForGroup

Тест 2.1.1.

Цель: проверка возврата адекватных данных сервисом SchedulesForGroup.

Тип: общий.

Кем и как выполняется: тест выполняется программистом, непосредственно после создания сервиса SchedulesForGroup в APEX. Тест выполняется в два этапа: сначала в браузере машины разработчика, затем в браузере устройства с ОС Android.

Входные данные: serveraddress/api/SchedulesForGroup?groupCode=22509&specialityCode=01.04.02

Результат: в ответе от сервера должно прийти расписание группы 22509 специальности 01.04.02. Осуществляется сравнение с учебным планом данной группы.

Тест 2.1.2.

Цель: проверка обработки сервисом SchedulesForGroup некорректного запроса (не указаны значения параметров и указаны не все параметры).

Тип: негативный.

Кем и как выполняется: тест выполняется программистом, непосредственно после создания сервиса SchedulesForGroup в APEX.

Входные данные: serveraddress/api/SchedulesForGroup?groupCode=

Результат: в ответе от сервера APEX должно прийти сообщение об отсутствии сервиса с подобными параметрами.

Тест 2.1.3.

Цель: проверка обработки сервисом SchedulesForGroup некорректного запроса (не указаны параметры).

Тип: негативный.

Кем и как выполняется: тест выполняется программистом, непосредственно после создания сервиса SchedulesForGroup в APEX.

Входные данные: serveraddress/api/SchedulesForGroup

Результат: в ответе от сервера должна прийти ошибка с кодом 404. (Not Found)

2.2. Модульные тесты для сервиса SchedulesForAuditorium

Тест 2.2.1.

Цель: проверка возврата адекватных данных сервисом SchedulesForAuditorium.

Тип: общий.

Кем и как выполняется: тест выполняется программистом, непосредственно после создания сервиса SchedulesForAuditorium в APEX. Тест выполняется в два этапа: сначала в браузере машины разработчика, затем в браузере устройства с ОС Android.

Входные данные: serveraddress/api/SchedulesForAuditorium?auditoriumNumber=146&buildingShortName=ГК

Результат: в ответе от сервера должно прийти расписание аудитории номер 146 в главном корпусе. Осуществляется проверка по базе данных.

Тест 2.2.2.

Цель: проверка обработки сервисом SchedulesForAuditorium некорректного запроса (не указаны значения параметров и указаны не все параметры).

Тип: негативный.

Кем и как выполняется: тест выполняется программистом, непосредственно после создания сервиса SchedulesForAuditorium в APEX.

Входные данные: serveraddress/api/SchedulesForAuditorium?auditoriumNumber=

Результат: в ответе от сервера APEX должно прийти сообщение об отсутствии сервиса с подобными параметрами.

Тест 2.2.3.

Цель: проверка обработки сервисом SchedulesForAuditorium некорректного запроса (не указаны параметры).

Тип: негативный.

Кем и как выполняется: тест выполняется программистом, непосредственно после создания сервиса SchedulesForAuditorium в APEX.

Входные данные: serveraddress/api/SchedulesForAuditorium

Результат: в ответе от сервера должна прийти ошибка с кодом 404. (Not Found)

2.3. Модульные тесты для сервиса SchedulesForLecturer

Тест 2.3.1.

Цель: проверка возврата адекватных данных сервисом SchedulesForLecturer.

Тип: общий.

Кем и как выполняется: тест выполняется программистом, непосредственно после создания сервиса SchedulesForLecturer в APEX. Тест выполняется в два этапа: сначала в браузере машины разработчика, затем в браузере устройства с ОС Android.

Входные данные: serveraddres/api/SchedulesForLecturer?lecturerFullName=Кузнецов В. А.

Результат: в ответе от сервера должно прийти расписание преподавателя Кузнецова В. А. Осуществляется сравнение с учебным планом.

Тест 2.3.2.

Цель: проверка обработки сервисом SchedulesForLecturer некорректного запроса (не указаны значения параметров и указаны не все параметры).

Тип: негативный.

Кем и как выполняется: тест выполняется программистом, непосредственно после создания сервиса SchedulesForLecturer в APEX.

Входные данные:
serveraddress/api/SchedulesForLecturer?lecturerFullName=

Результат: в ответе от сервера APEX должно прийти сообщение об отсутствии сервиса с подобными параметрами.

Тест 2.3.3.

Цель: проверка обработки сервисом SchedulesForLecturer некорректного запроса (не указаны параметры).

Тип: негативный.

Кем и как выполняется: тест выполняется программистом, непосредственно после создания сервиса SchedulesForLecturer в APEX.

Входные данные: serveraddress/api/SchedulesForLecturer

Результат: в ответе от сервера должна прийти ошибка с кодом 404. (Not Found)

2.4. Модульные тесты для сервиса AuditoriumsSearch

Тест 2.4.1.

Цель: проверка возврата адекватных данных сервисом AuditoriumsSearch.

Тип: общий.

Кем и как выполняется: тест выполняется программистом, непосредственно после создания сервиса AuditoriumsSearch в APEX. Тест выполняется в два этапа: сначала в браузере машины разработчика, затем в браузере устройства с ОС Android.

Входные данные:
serveraddress/api/AuditoriumsSearch?template=1&count=5

Результат: в ответе от сервера должен прийти список из пяти аудиторий, чьи номера начинаются на 1.

Тест 2.4.2.

Цель: проверка возврата адекватных данных сервисом AuditoriumsSearch при указании пустого номера.

Тип: краевой.

Кем и как выполняется: тест выполняется программистом, непосредственно после создания сервиса AuditoriumsSearch в APEX.

Входные данные:
serveraddress/api/AuditoriumsSearch?template=’’&count=5

Результат: в ответе от сервера должен прийти пустой список аудиторий.

Тест 2.4.3.

Цель: проверка обработки сервисом AuditoriumsSearch некорректного запроса (не указаны значения параметров и указаны не все параметры).

Тип: негативный.

Кем и как выполняется: тест выполняется программистом, непосредственно после создания сервиса AuditoriumsSearch в APEX.

Входные данные:
serveraddress/api/AuditoriumsSearch?template=1&count=

Результат: в ответе от сервера APEX должно прийти сообщение об отсутствии сервиса с подобными параметрами.

Тест 2.4.4.

Цель: проверка обработки сервисом AuditoriumsSearch некорректного запроса (не указаны параметры).

Тип: негативный.

Кем и как выполняется: тест выполняется программистом, непосредственно после создания сервиса AuditoriumsSearch в APEX.

Входные данные: serveraddress/api/AuditoriumsSearch

Результат: в ответе от сервера должна прийти ошибка с кодом 404. (Not Found)

2.5. Модульные тесты для сервиса GroupsSearch

Тест 2.5.1.

Цель: проверка возврата адекватных данных сервисом GroupsSearch.

Тип: общий.

Кем и как выполняется: тест выполняется программистом, непосредственно после создания сервиса GroupsSearch в APEX. Тест выполняется в два этапа: сначала в браузере машины разработчика, затем в браузере устройства с ОС Android.

Входные данные:
serveraddress/api/GroupsSearch?template=2250&count=5

Результат: в ответе от сервера должен прийти список из пяти студенческих групп, чьи номера начинаются с 2250. Осуществляется проверка с базой данных.

Тест 2.5.2.

Цель: проверка возврата адекватных данных сервисом GroupsSearch при указании пустого номера группы.

Тип: краевой.

Кем и как выполняется: тест выполняется программистом, непосредственно после создания сервиса GroupsSearch в APEX.

Входные данные: serveraddress /api/GroupsSearch?template=’’&count=5

Результат: в ответе от сервера должен прийти пустой список аудиторий.

Тест 2.5.3.

Цель: проверка обработки сервисом GroupsSearch некорректного запроса (не указаны значения параметров и указаны не все параметры).

Тип: негативный.

Кем и как выполняется: тест выполняется программистом, непосредственно после создания сервиса GroupsSearch в APEX.

Входные данные: /api/GroupsSearch?template=2250&count=

Результат: в ответе от сервера APEX должно прийти сообщение об отсутствии сервиса с подобными параметрами.

Тест 2.5.4.

Цель: проверка обработки сервисом GroupsSearch некорректного запроса (не указаны параметры).

Тип: негативный.

Кем и как выполняется: тест выполняется программистом, непосредственно после создания сервиса GroupsSearch в APEX.

Входные данные: serveraddress/api/GroupsSearch

Результат: в ответе от сервера должна прийти ошибка с кодом 404. (Not Found)

2.6. Модульные тесты для сервиса LecturersSearch

Тест 2.6.1.

Цель: проверка возврата адекватных данных сервисом LecturersSearch.

Тип: общий.

Кем и как выполняется: тест выполняется программистом, непосредственно после создания сервиса LecturersSearch в APEX. Тест выполняется в два этапа: сначала в браузере машины разработчика, затем в браузере устройства с ОС Android.

Входные данные: serveraddress/api/SchedulesForGroup?groupCode=22509&specialityCode=01.04.02

Результат: в ответе от сервера должно прийти расписание группы 22509 специальности 01.04.02. Осуществляется сравнение с учебным планом данной группы.

Тест 2.6.2.

Цель: проверка возврата адекватных данных сервисом LecturersSearch при указании пустого номера.

Тип: краевой.

Кем и как выполняется: тест выполняется программистом, непосредственно после создания сервиса LecturersSearch в APEX.

Входные данные: serveraddress/api/AuditoriumsSearch?template=’’&count=5

Результат: в ответе от сервера должен прийти пустой список аудиторий.

Тест 2.6.3.

Цель: проверка обработки сервисом LecturersSearch некорректного запроса (не указаны значения параметров и указаны не все параметры).

Тип: негативный.

Кем и как выполняется: тест выполняется программистом, непосредственно после создания сервиса LecturersSearch в APEX.

Входные данные: serveraddress/api/SchedulesForGroup?groupCode=

Результат: в ответе от сервера APEX должно прийти сообщение об отсутствии сервиса с подобными параметрами.

Тест 2.6.4.

Цель: проверка обработки сервисом LecturersSearch некорректного запроса (не указаны параметры).

Тип: негативный.

Кем и как выполняется: тест выполняется программистом, непосредственно после создания сервиса LecturersSearch в APEX.

Входные данные: serveraddress/api/ LecturersSearch

Результат: в ответе от сервера должна прийти ошибка с кодом 404. (Not Found)

3. Тесты полной интеграции

3.1. Тесты страниц

Аналогичны тестам из пункта 1, но проводятся не программистом, а тестировщиком с использованием реальных сервисов, а не заглушки dataService, соответственно ожидаемые результаты берутся из пункта 3.

Тест 3.1.1.

Аналог теста 1.1.1.

Тест 3.1.2.

Аналог теста 1.1.2.

Тест 3.2.1.

Аналог теста 1.2.1.

Тест 3.2.2.

Аналог теста 1.2.2.

Тест 3.3.1.

Аналог теста 1.3.1.

Тест 3.3.2.

Аналог теста 1.3.2.

4. Системное тестирование

4.1. Загрузка, установка и запуск приложения.

Тест 4.1.1.

Цель: проверка загрузки из магазина, установки и запуска приложения.

Тип: системный.

Кем и как выполняется: тест выполняется тестировщиком с помощью мобильного устройства.

Входные данные: с мобильного устройства зайти на Google Play маркет и установить приложение «Расписание ПетрГУ», найти ярлык на рабочем столе устройства и запустить приложение.

Результат: приложение должно установиться и запуститься без ошибок, после запуска должна отобразиться страница home.html.

4.2. Контрольное тестирование

Тест 4.2.1.

Цель: проверка работы приложения

Тип: общий

Кем и как выполняется: выполняется тестировщиком вручную или с помощью автоматизированного сценария (например на языке C# с использованием библиотеки Selenium). Выполняется в браузере рабочей машины тестировщика.

Входные данные: выполняется следующая последовательность переходов home.html -> lecturerSchedule.html -> map.html -> lecturerSchedule.html -> home.html -> groupSchedule.html -> home.html.

Результат: все работает в соответствии со спецификацией

Тест 4.2.2.

Цель: стресс проверка UI.

Тип: свободный

Кем и как выполняется: выполняется тестировщиком вручную или спомощью автоматизированного сценария (например на языке C# с использованием библиотеки Selenium). Выполняется в браузере рабочей машины тестировщика.

Входные данные: произвольные, на усмотрение тестировщика.

Результат: поведение программы, ожидаемое в соответствии со спецификацией.

Тест 4.2.3.

Цель: стресс проверка UI на мобильном устройстве.

Тип: свободный

Кем и как выполняется: выполняется тестировщиком вручную. Выполняется на мобильном устройстве.

Входные данные: произвольные, на усмотрение тестировщика.

Результат: поведение программы, ожидаемое в соответствии со спецификацией.

Тест 4.2.4.

Цель: контрольная проверка UI на эмуляторе.

Тип: свободный

Кем и как выполняется: выполняется тестировщиком вручную. Выполняется на эмуляторе JenyMotion под всеми версиями Android >= 4.4.0.

Входные данные: произвольные, на усмотрение тестировщика.

Результат: поведение программы, ожидаемое в соответствии со спецификацией.

Тест 4.2.5.

Цель: контрольное испытание приложения в присутствии заказчика.

Тип: аттестационный.

Кем и как выполняется: тест выполняется тестировщиком с помощью мобильного устройства в присутствии заказчика.

Входные данные: Найти ярлык на рабочем столе устройства и запустить приложение, перейти на страницу расписания групп, в поле

поиска группы ввести значение “2250” (без кавычек). Из списка групп выбрать группу 22509. Прокрутить список расписания вниз до конца. Выбрать первое занятие, щелкнуть на аудиторию в первом занятии.

Результат: приложение должно запуститься без ошибок, после запуска должна отобразиться страница home.html. Должно загрузиться расписание группы 22509. При щелчке на аудиторию, должна отобразиться страница карты с корпусом в котором находится аудитория.

5. Бета-тестирование

Тест 5.0.0.

Цель: бета-тестирование приложения.

Тип: специальный.

Кем и как выполняется: выполняется пользователями по желанию.

Входные данные: нет

Результат: отзывы пользователей о работе ПО.

6. Специальные тесты

6.1. Нагрузочные испытания сервера

Тест 6.1.1.

Цель: нагрузочное испытание сервера.

Тип: специальный.

Кем и как выполняется: выполняется тестировщиком.

Входные данные: скрипт с параметрами connections = 20, timeout = 1000.

Результат: все ответы с сервера должны быть success.

Требования к тестировщикам [к содержанию]

Для тестирования достаточно одного специализированного тестировщика, обладающего большей частью нижеприведенного списка качеств:

- Последовательность
- Нацеленность на качество
- Психология экспериментатора
- Высшее образование
- Опыт программирования
- Опыт работы с ПО подобного класса
- Знание комбинаторики
- Умение выражать мысли
- Талант поиска ошибок
- Абстрактное мышление
- Эффективное использование времени
- Способность быстро переключаться между задачами
- Навыки планирования
- Терпение
- Способность представить себя на месте другого
- Умение читать и писать спецификации

Пример исходного кода [к содержанию]

Пример функции вывода списка студенческих групп по шаблону номера

```
$scope.searchGroups = function() {


    if(dataService.isCorrectQueryTemplate($scope.data.search.template))
{

        $scope.data.groups = [];
        $scope.isLoadingGroups = true;

        $http.get(prefixAPEX + 'groupsSearch/' +
$scope.data.search.template + '/10', {}).error(function (data, status)
{
            onNetworkError(status);
        }).then(function (response) {
            $scope.data.groups = response.data.items;
            $scope.isLoadingGroups = false;
        });

    }else{
        $scope.data.groups = [];
        $scope.isLoadingGroups = false;
```

```
        }
};
```

Методы покрытия [к содержанию]

Расчёт покрытия тестами относительно исполняемого кода производится по формуле:

$$Covering = tested\_length / code\_length$$

$tested\_length$ - количество строк кода, покрытых тестами.
$code\_length$ - общее количество строк кода в программе.

- JS: всего 978 строк, покрыто 751
- HTML: 418 строк, покрыто 385
- SQL: 116 строк, покрыто 116
- Всего: 1512 строк, покрыто 1252.

Коэффициент покрытия: 0,828

Процент покрытия: 82,8%

Отчет о проведении тестирования [к содержанию]

Каждый тест запускается один раз.

Первоначальный запуск тестов:

Номер	Дата	Результат
1.1.1	2016-01-15 20:40	**пройден**
1.1.2	2016-01-15 20:41	**пройден**
1.1.3	2016-01-15 20:42	**1 ошибка**
1.1.4	2016-01-15 20:44	**пройден**
1.2.1	2016-01-15 20:50	**пройден**
1.2.2	2016-01-15 20:51	**пройден**
1.3.1	2016-01-15 20:53	**пройден**
1.3.2	2016-01-15 20:54	**пройден**
1.3.3	2016-01-15 20:56	**пройден**
1.3.4	2016-01-15 21:01	**пройден**
1.4.1	2016-01-15 21:05	**пройден**
1.4.2	2016-01-15 21:06	**пройден**
1.4.3	2016-01-15 21:07	**пройден**
1.5.1	2016-01-15 21:15	**пройден**
1.6.1	2016-01-15 21:18	**пройден**
1.6.2	2016-01-15 21:23	**пройден**

1.6.3	2016-01-15 21:25	**пройден**
1.6.4	2016-01-15 21:26	**пройден**
1.6.5	2016-01-15 21:30	**пройден**
1.6.6	2016-01-15 21:34	**пройден**
1.6.7	2016-01-15 21:37	**пройден**
2.1.1	2016-01-15 21:45	**1 ошибка**
2.1.2	2016-01-15 21:46	**пройден**
2.1.3	2016-01-15 21:47	**пройден**
2.2.1	2016-01-15 21:48	**пройден**
2.2.2	2016-01-15 21:49	**пройден**
2.2.3	2016-01-15 21:50	**пройден**
2.3.1	2016-01-15 21:57	**1 ошибка**
2.3.2	2016-01-15 22:01	**пройден**
2.3.3	2016-01-15 22:02	**пройден**
2.4.1	2016-01-15 22:03	**пройден**
2.4.2	2016-01-15 22:04	**пройден**
2.4.3	2016-01-15 22:05	**пройден**
2.4.4	2016-01-15 22:06	**пройден**
2.5.1	2016-01-15 22:13	**пройден**
2.5.2	2016-01-15 22:15	**пройден**
2.5.3	2016-01-15 22:16	**пройден**
2.5.4	2016-01-15 22:20	**пройден**
2.6.1	2016-01-15 22:25	**пройден**
2.6.2	2016-01-15 22:26	**пройден**
2.6.3	2016-01-15 22:27	**пройден**
2.6.4	2016-01-15 22:29	**пройден**
3.1.1	2016-01-15 22:31	**пройден**
3.1.2	2016-01-15 22:34	**пройден**
3.2.1	2016-01-15 22:35	**пройден**
3.2.2	2016-01-15 22:36	**пройден**
3.3.1	2016-01-15 22:37	**пройден**
3.3.2	2016-01-15 22:48	**пройден**
4.1.1	2016-01-15 22:50	**1 ошибка**
4.2.1	2016-01-15 22:53	**пройден**
4.2.2	2016-01-15 22:54	**пройден**
4.2.3	2016-01-15 22:57	**пройден**
4.2.4	2016-01-15 23:05	**пройден**
4.2.5	2016-01-15 23:07	**пройден**
5.0.0	с 2015-12-20 23:01	**2 ошибки**
6.1.1	2016-01-15 23:25	**пройден**

Отчет об ошибках [к содержанию]

Ошибки на тесте 1.1.3.

Ошибка 1.1.3.1.

На странице groupSchedule.html при вводе текста в поле поиска группы, вводимый текст отображается неровно (вылезает за рамку поля). Мобильное устройство Lenovo A530 Android 4.4.0.

Ожидаемый результат: Текст должен быть полностью в поле ввода.

Фактический результат: Текст вылезает за рамки поля ввода.

Алгоритм воспроизведения ошибки:

Зайти на страницу group_schedule.html и ввести в поле текст 22509.

Решение обновление библиотеки Ionic на night-версию.

Ошибки на тесте 2.1.1.

Ошибка 2.1.1.1.

В браузере мобильного устройства невозможно подключиться к сервисам, с браузера компьютера при этом все работает.

Ожидаемый результат: JSON список расписания для группы 22509.

Фактический результат: ошибка сертификата безопасности.

Алгоритм воспроизведения ошибки: ввести в адресную строку браузера на мобильном устройстве (встроенного или google chrome): serveraddress/api/SchedulesForGroup?groupCode=22509&specialityCode=01.04.02

Проблема связана с тем, что мобильные браузеры (на момент тестирования) не поддерживают последние протоколы шифрования: TLS v 1.0, TLS v 1.1, TLS v1.2, а поддерживают только SSLv2, SSLv3. Браузер в телефоне не обновлен.

Ошибки на тесте 2.3.1.

Ошибка 2.3.1.1.

Не найден сервис.

Ожидаемый результат: JSON список занятий преподавателя Кузнецова В. А.

Фактический результат: ошибка 404, страница не найдена.

Алгоритм воспроизведения ошибки:

Для воспроизведение ввести в адресную строку браузера: serveraddres/api/SchedulesForLecturer?lecturerFullName=Кузнецов В. А.

Проблема в описании параметра lecturerFullName

Ошибки на тесте 4.1.1.

Ошибка 4.1.1.1.

Ярлык приложения слишком мал, по сравнению с остальными ярлыками.

Ожидаемый результат: Размеры ярлыка приложения соответствуют остальным ярлыкам.

Фактический результат: Размеры ярлыка меньше чем остальные.

Алгоритм воспроизведения ошибки:

Установить приложение, посмотреть ярлык.

Ошибки на тесте 5.0.0. [6]

Ошибка 5.0.0.1.

Комментарий пользователя в Google Play:

«При поиске расписания преподавателей надо сделать поиск либо без учета регистра, либо чтобы принудительно первая буква печаталась в верхнем регистре.»

Ошибка 5.0.0.2.

Комментарий пользователя в Google Play:

«Сделать более понятное разделение по числ/знам»

Повторный запуск тестов:

Номер	Дата	Результат
1.1.1	2016-01-15 20:40	**пройден**
1.1.2	2016-01-15 20:41	**пройден**
1.1.3	2016-01-15 20:42	**пройден**
1.1.4	2016-01-15 20:44	**пройден**
1.2.1	2016-01-15 20:50	**пройден**
1.2.2	2016-01-15 20:51	**пройден**
1.3.1	2016-01-15 20:53	**пройден**
1.3.2	2016-01-15 20:54	**пройден**
1.3.3	2016-01-15 20:56	**пройден**

1.3.4	2016-01-15 21:01	**пройден**
1.4.1	2016-01-15 21:05	**пройден**
1.4.2	2016-01-15 21:06	**пройден**
1.4.3	2016-01-15 21:07	**пройден**
1.5.1	2016-01-15 21:15	**пройден**
1.6.1	2016-01-15 21:18	**пройден**
1.6.2	2016-01-15 21:23	**пройден**
1.6.3	2016-01-15 21:25	**пройден**
1.6.4	2016-01-15 21:26	**пройден**
1.6.5	2016-01-15 21:30	**пройден**
1.6.6	2016-01-15 21:34	**пройден**
1.6.7	2016-01-15 21:37	**пройден**
2.1.1	2016-01-15 21:45	**пройден**
2.1.2	2016-01-15 21:46	**пройден**
2.1.3	2016-01-15 21:47	**пройден**
2.2.1	2016-01-15 21:48	**пройден**
2.2.2	2016-01-15 21:49	**пройден**
2.2.3	2016-01-15 21:50	**пройден**
2.3.1	2016-01-15 21:57	**пройден**
2.3.2	2016-01-15 22:01	**пройден**
2.3.3	2016-01-15 22:02	**пройден**
2.4.1	2016-01-15 22:03	**пройден**
2.4.2	2016-01-15 22:04	**пройден**
2.4.3	2016-01-15 22:05	**пройден**
2.4.4	2016-01-15 22:06	**пройден**
2.5.1	2016-01-15 22:13	**пройден**
2.5.2	2016-01-15 22:15	**пройден**
2.5.3	2016-01-15 22:16	**пройден**
2.5.4	2016-01-15 22:20	**пройден**
2.6.1	2016-01-15 22:25	**пройден**
2.6.2	2016-01-15 22:26	**пройден**
2.6.3	2016-01-15 22:27	**пройден**
2.6.4	2016-01-15 22:29	**пройден**
3.1.1	2016-01-15 22:31	**пройден**
3.1.2	2016-01-15 22:34	**пройден**
3.2.1	2016-01-15 22:35	**пройден**
3.2.2	2016-01-15 22:36	**пройден**
3.3.1	2016-01-15 22:37	**пройден**
3.3.2	2016-01-15 22:48	**пройден**
4.1.1	2016-01-15 22:50	**пройден**
4.2.1	2016-01-15 22:53	**пройден**
4.2.2	2016-01-15 22:54	**пройден**

4.2.3	2016-01-15 22:57	**пройден**
4.2.4	2016-01-15 23:05	**пройден**
4.2.5	2016-01-15 23:07	**пройден**
5.0.0	с 2015-12-20 23:01	**Условно пройден**
6.1.1	2016-01-15 23:25	**пройден**

Перечень отложенных ошибок

Исправление двух некритичных ошибок (5.0.0.1 и 5.0.0.2) отложено вследствие нехватки имеющихся ресурсов.

- Ошибка 5.0.0.1
- Ошибка 5.0.0.2

Текущие результаты [к содержанию]

Разработана программа, реализующая все требуемые функции и удовлетворяющая всем требованиям. Разработано более 50 различных тестов.

Программа успешно прошла тестирование. Было найдено 6 ошибок, из них 4 ошибки уже исправлены. Проведенное тестирование позволяет надеяться на невысокую вероятность возникновения новых серьезных ошибок в работе текущей версии приложения.

Внешние источники [к содержанию]

[1] Ionic framework. http://ionicframework.com/

[2] Google play market. https://play.google.com/store?hl=ru

[3] Доля Lollipop на Android-рынке. https://www.3dnews.ru/924889

[4] Ionic framework. Обзор экосистемы. http://habrahabr.ru/company/simpleweek/blog/254681/

[5] Эмулятор GenyMotion. https://www.genymotion.com/#!/

[6] Отзывы пользователей. https://play.google.com/store/apps/details?id=com.ionicframework.myapp259509&hl=ru