ПЕТРОЗАВОДСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
КАФЕДРА ИНФОРМАТИКИ И МАТЕМАТИЧЕСКОГО ОБЕСПЕЧЕНИЯ

Отчёт по курсу
«Верификация ПО»

Выполнила:
студентка группы 22608 Т.В. Старикова

Преподаватель:
к.ф-м.н., доцент К. А. Кулаков

Петрозаводск
2014

СОДЕРЖАНИЕ

Объект тестирования

Report System – это система учета рабочего времени и генерации отчетов. Реализована данная система в виде веб-приложения с использованием технологии ASP.NET (на основе фреймворка MVC 4).

Report System позволяет работникам вести учет затраченного на различные проекты времени и подводить итоги о проведенной работе. Кроме этого система позволяет менеджеру управлять командой и генерировать отчеты за выбранный промежуток времени. Отчеты имеют строгий, заранее определенный и принятый формат.

Основные особенности

1. В системе допустимы два вида пользователей: разработчики и менеджеры - требуется произвести проверку разграничения прав доступа, т.к. функционал, доступный менеджеру, не должен быть доступен разработчику (применяется тестирование: блочное, аттестационное);
2. Разработчики и менеджеры (далее именуются как работники) имеют возможность логировать рабочий часы с указанием:
   - Даты;
   - Проекта, над которым велась работа;
   - Вида совершаемой деятельности над данным проектом;
   - Затраченного времени;

   Данный функционал должен пройти проверку. Применяется тестирование: блочное, интеграционное, аттестационное.
3. Работники имеют возможность просматривать итоговый отчет о затраченном времени, редактировать и удалять записи из отчета. Данный функционал должен пройти проверку. Применяется тестирование: блочное, интеграционное, аттестационное;

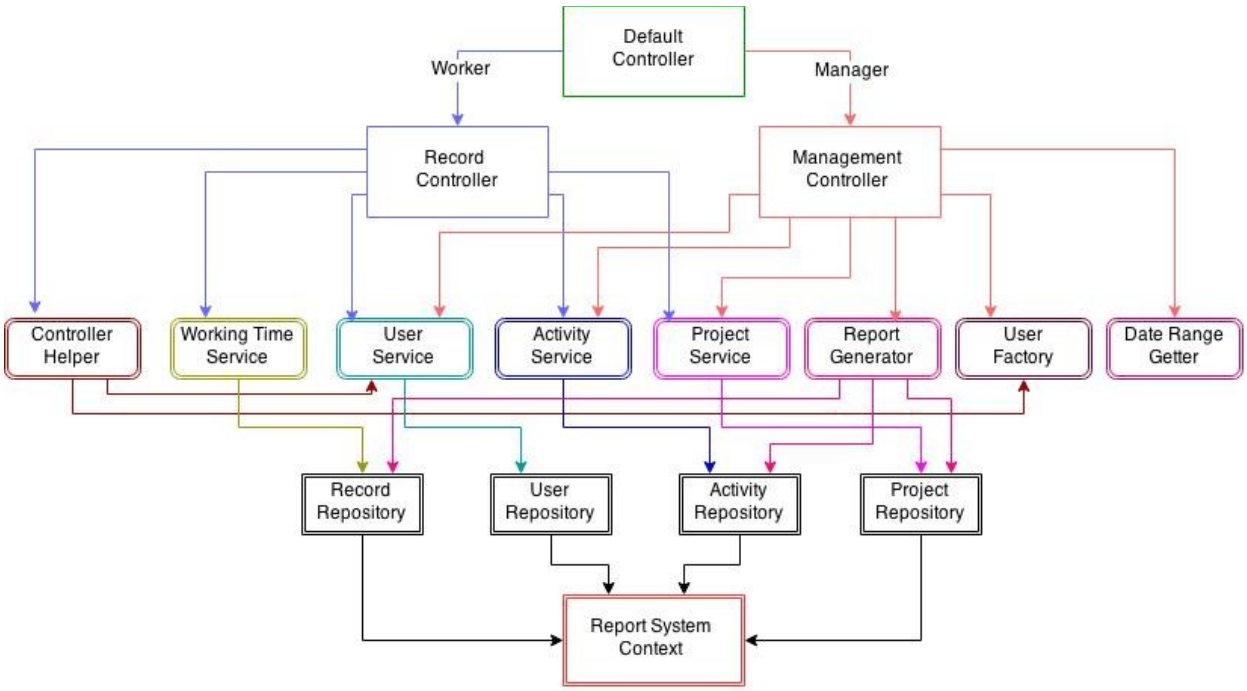
Функционал доступный менеджеру (пункты 4 – 7) проходить проверку не будет, поскольку представляет собой отдельный модуль, доступ к которому не предоставляется.

4. Менеджеры имеют возможность управлять командой:
   - Добавлять информацию о новых работниках (идентификатор, имя и роль работника);
   - Редактировать информацию о существующих работниках;
   - Удалять информацию о существующих работниках;

   Данный функционал проходить проверку не будет, поскольку представляет собой отдельный модуль, доступ к которому не предоставляется.
5. Менеджеры имеют возможность управлять проектами (создавать, редактировать и удалять проекты);
6. Менеджеры имеют возможность управлять видами деятельности (создавать, редактировать и удалять виды деятельности);
7. Менеджер имеет возможность генерировать, просматривать и скачивать отчет в определенном формате за выбранный промежуток времени. Каждая запись отчета включает в себя следующую информацию:
   - Имя работника;
   - Проект, над которым велась работа;
   - Деятельность, которая осуществлялась над проектом;
   - Первый день недели (дата);
   - Затраченные часы за каждый день указанной недели (отдельные столбцы для каждого дня недели);
   - Общее время, потраченное на указанный проект, указанный вид деятельности за указанную неделю;

Диаграмма классов

На приведенном ниже рисунке показаны зависимости между классами, отражены основные компоненты, используемые в системе: контроллеры, сервисы, генераторы, фабрики, репозитории и контексты (для работы с БД).



Описание элементов системы

1) **Default Controller** – точка входа в систему. На данном этапе производится проверка роли пользователя и происходит перенаправление на Record Controller (если пользователь является разработчиком) или на Management Controller (если пользователь является менеджером). Тестирование: произвести функциональное, аттестационное, а также интеграционное тестирование (проверить взаимодействие с **Record Controller (1)**).
2) **Record Controller** – обращается к сервисам, позволяющим работникам добавлять, редактировать и удалять записи, а также просматривать отчет затраченного времени. Тестирование: произвести функциональное и аттестационное тестирование.
3) **Management Controller** – обращается к сервисам, позволяющим менеджеру управлять работниками, проектами и видами деятельности, генерировать, скачивать и просматривать отчеты за определенный промежуток времени. Тестирование не производится, так как модуль не доступен.
4) **Controller Helper** – предоставляет информацию о текущем пользователе, взаимодействующим с системой. Тестирование:

произвести функциональное, а также интеграционное тестирование (проверить взаимодействие с **Record Controller (2), User Service (4)**).

5) **Working Time Service** – обращается к Record Repository с просьбой получить все (или определенную) записи текущего работника, добавить, удалить или отредактировать запись. Тестирование: произвести функциональное, а также интеграционное тестирование (проверить взаимодействие с **Record Controller (3), Record Repository(5)**).
6) **User Service** – обращается к User Repository с просьбой получить всех пользователей системы, добавить нового пользователя, отредактировать или удалить информацию о существующих пользователях. Тестирование производится обоюдно с тестированием других сервисов.
7) **Activity Service** - обращается к Activity Repository с просьбой получить все виды деятельности, добавить новый, отредактировать или удалить информацию о существующих видах деятельности. Тестирование производится обоюдно с тестированием других сервисов.
8) **Project Service** - обращается к Project Repository с просьбой получить все проекты, добавить новый, отредактировать или удалить информацию о существующих проектах. Тестирование производится обоюдно с тестированием других сервисов.
9) **Report Generator** – позволяет генерировать финальный отчет за определенный промежуток времени. Тестирование не производится, так как модуль не доступен.
10) **User Factory** – позволяет создавать экземпляр профиля работника. Тестирование не производится, так как модуль не доступен.
11) **Date Range Getter** – позволяет сформировать начальную и конечную дату для создания финального отчета. Тестирование не производится, так как модуль не доступен.
12) **Record Repository** – через контекст обращается к БД для работы с данными о записях пользователя. Тестирование: произвести

интеграционное тестирование (проверить взаимодействие с **Working Time Service(6)**.

13) **User Repository** – через контекст обращается к БД для работы с данными о пользователях. Тестирование не производится, так как модуль не доступен. Тестирование производится обоюдно с тестированием других репозиториев.
14) **Project Repository** – через контекст обращается к БД для работы с данными о проектах. Тестирование не производится, так как модуль не доступен. Тестирование производится обоюдно с тестированием других репозиториев.
15) **Activity Repository** – через контекст обращается к БД для работы с данными о видах деятельности. Тестирование не производится, так как модуль не доступен. Тестирование производится обоюдно с тестированием других репозиториев.
16) **Report System Context** – контекст для работы с БД. Тестирование не производится, так как модуль не доступен.
17) **Allow Access Attribute** – перенаправляет пользователя на соответствующую ему страницу. Тестирование: произвести функциональное, а также интеграционное тестирование (проверить взаимодействие с **Record Controller, Management Controller**).

В скобках указывается очередность прохождения интеграционного тестирования.

Стратегия проведения тестирования

Тестирование будут проходить следующие классы и функции:

1) **DefaultController : Controller, IDefaultController**
   - RedirectToRouteResult Index() – получает роль пользователя и производит перенаправление на **Record** или на **Management**
2) **RecordController : Controller, IRecordController**
   - **ActionResult AddRecord(RecordFormModel recordFormModel) -** если модель валидна, то обращается к **WorkingTimeService** для логирования времени, иначе производит перенаправление на заполнение формы**.**
   - **RedirectToRouteResult Delete(int recordId) -** обращается к **WorkingTimeService** для удаления записи.

- **ActionResult Edit(int recordId)** - обращается к **WorkingTimeService** для получения записи для редактирования.
- **ActionResult EditRecord(RecordFormModel recordFormModel)** – если модель валидна, то обращается к **WorkingTimeService** для сохранения изменений, иначе производит перенаправление на заполнение формы**.**
- **ActionResult Index()**

3) **Working Time Service : IWorkingTimeService**
   - **void Log(Record record)** – обращается к репозиторию для сохранения записи.
   - **IEnumerable<Record> Get() –** обращается к репозиторию для получения всех записей и возвращает их.
   - **void Delete(int recordId)** – обращается к репозиторию для удаления записи.
   - **Record GetRecord(int recordId)** – обращается к репозиторию для получения свойст записи.
   - **IEnumerable<Record> GetUsersRecords(string developerId)** – обращается к репозиторию для получения записей пользователя.
4) **AllowAccessAttribute : AuthorizeAttribute**
   - **bool IsInRole(string username, string roles)** – проверяет имеет ли пользователь права доступа к запрашиваемой странице.

Для валидации введенных пользователем данных в системе используется механизм присваивания атрибутов валидации. Такие атрибуты позволяют устанавливать различные требования к данным в поле (обязательно заполнение или нет, разрешены ли отрицательные значения, какие разрешены символы, максимальная длина введенных в поле данных и т.п.)

Блочное тестирование.

Для проведения блочного тестирования необходимо определить обозначить все возможные входные данные, соответствующие им выходные данные, а также все зависимости от других модулей.

Для совершения процедуры тестирования необходимы:

- Microsoft Visual Studio 2010 (и выше)
- Библиотека Moq (для создания заглушек)

Для создания заглушек используется класс Mock, позволяющий перехватывать вызовы к реальным классам, а также подменять возвращаемые значения.

Интеграционное тестирование.

Для проведения интеграционного тестирования применяется стратегия нисходящего тестирования.

Для каждого модуля необходимо определить зависимые модули и проверить правильность их взаимодействия.

Тестирование производится в порядке описанном в «Основные элементы системы».

Для совершения процедуры тестирования необходимы:

- Microsoft Visual Studio 2010 (и выше)

Аттестационное тестирование.

Для проведения аттестационного тестирования необходимо изучить требования к функционалу системы, определить возможные пути выполнения программы.

Для совершения процедуры тестирования необходимы:

- Браузер (Google Chrome, Mozilla Firefox, Internet Explorer)
- Java 7.0 и выше

Критерий прохождения тестов (применим для всех видов тестов)

Любой тест считается успешно пройденным, если ожидаемый и фактический результаты совпадают. Если тест завершается неудачей, то перед принятием решения целесообразно проверить правильность самого теста.

Если тест завершился неудачей и тест реализован правильно, то производится заключение о найденной ошибке.

Критерий приостановления тестирования (применим для всех видов тестов)

Тестирование должно быть приостановлено, если количество не пройдённых тестов превысит 10% от общего количества.

Критерий возобновления работы (применим для всех видов тестов)

Необходимо заново начать тестирование при получении уведомления, что найденные при тестировании ошибки исправлены.

Результаты тестирования должны быть представлены в январе 2015 года.

Детальный план тестов

Модульное тестирование

1.
Класс: AllowAccessAttribute
Метод: bool IsInRole(string username, string roles)
Описание: проверка создания нового пользователя с параметрами по умолчанию в случае, если пользователь не найден
Тип теста: специальный
Входные элементы: несуществующий username
Выходные элементы: true, если пользователю разрешен доступ, exception если не разрешен
Зависимости:
Класс: UserService
Метод: IEnumerable<User> Get ()
Способ использования: Mock-объект
Возвращаемое значение: коллекция объектов класса **User**
Ожидаемый результат: Пользователь не найден, создан новый пользователь

2.
Класс: AllowAccessAttribute
Метод: bool IsInRole(string username, string roles)
Описание: проверка доступа пользователя к заращиваемой странице, если пользователь найден
Тип теста: общий
Входные элементы: существующий username, допустимые роли пользователей (содержат роль пользователя)
Выходные элементы: true, если пользователю разрешен доступ, exception если не разрешен
Зависимости:
Класс: UserService
Метод: IEnumerable<User> Get ()
Способ использования: Mock-объект
Возвращаемое значение: коллекция объектов класса **User**
Ожидаемый результат: Пользователь найден, возвращаемое значение true

3.
Класс: AllowAccessAttribute
Метод: bool IsInRole(string username, string roles)
Описание: проверка доступа пользователя к заращиваемой странице, если пользователь найден
Тип теста: негативные
Входные элементы: существующий username, допустимые роли пользователей (не содержат роль пользователя)

Выходные элементы: true, если пользователю разрешен доступ, exception если не разрешен
Зависимости:
Класс: UserService
Метод: IEnumerable<User> Get ()
Способ использования: Mock-объект
Возвращаемое значение: коллекция объектов класса **User**
Ожидаемый результат: Пользователь найден, HttpException 403 (Forbidden)

4.
Класс: WorkingTimeService
Метод: IEnumerable<Record> Get()
Описание: проверка создания коллекции записей ,если записи есть
Тип теста: общий
Входные элементы: -
Выходные элементы: коллекция записей
Зависимости:
Класс: RecordRepository
Метод: IQueryable<Record> GetRecords ()
Способ использования: Mock-объект
Возвращаемое значение: коллекция объектов класса **Record**
Ожидаемый результат: Коллекция объектов класса **Record**; Коллекция из 2 записей
Аналогично для других сервисов (далее приведены различия):
4.1
Класс: UserService
Зависимости: UserRepository
Ожидаемый результат: Коллекция объектов класса **User**; Коллекция из 2 записей

4.2
Класс: ProjectService
Зависимости: ProjectRepository
Ожидаемый результат: Коллекция объектов класса **Project**; Коллекция из 2 записей

4.3
Класс: ActivityService
Зависимости: ActivityRepository
Ожидаемый результат: Коллекция объектов класса **Activity**; Коллекция из 2 записей

5.
Класс: WorkingTimeService
Метод: IEnumerable<Record> Get()

Описание: проверка создания пустой коллекции, если записей нет
Тип теста: специальный
Входные элементы: -
Выходные элементы: коллекция записей
Зависимости:
Класс: RecordRepository
Метод: IEnumerable<Record> GetRecords ()
Способ использования: Mock-объект
Возвращаемое значение: пустая коллекция объектов класса **Record**
Ожидаемый результат: Пустая коллекция

Аналогично для других сервисов (далее приведены различия):
5.1
Класс: UserService
Зависимости: UserRepository
5.2
Класс: ProjectService
Зависимости: ProjectRepository
5.3
Класс: ActivityService
Зависимости: ActivityRepository

6.
Класс: WorkingTimeService
Метод: Record GetRecord(int recordId)
Описание: проверка создания объекта, если запись с указанным идентификатором существует
Тип теста: общий
Входные элементы: идентификатор записи **recordId** (существующий)
Выходные элементы: объект класса Record с указанным идентификатором
Зависимости:
Класс: RecordRepository
Метод: IQueryable<Record> GetRecords ()
Способ использования: Mock-объект
Возвращаемое значение: коллекция объектов класса **Record**
Ожидаемый результат: Возвращаемое значение – запись с идентификатором “8”.

Аналогично для других сервисов (далее приведены различия):
6.1
Класс: UserService
Зависимости: UserRepository

Выходные элементы: объект класса User с указанным идентификатором

6.2
Класс: ProjectService
Зависимости: ProjectRepository
Выходные элементы: объект класса Project с указанным идентификатором

6.3
Класс: ActivityService
Зависимости: ActivityRepository
Выходные элементы: объект класса Activity с указанным идентификатором

7.
Класс: WorkingTimeService
Метод: Record GetRecord(int recordId)
Описание: проверка возвращаемого значения на null, если запись с указанным идентификатором не существует.
Тип теста: негативный
Входные элементы: идентификатор записи **recordId** (несуществующий)
Выходные элементы: объект класса Record с указанным идентификатором
Зависимости:
Класс: RecordRepository
Метод: IQueryable<Record> GetRecords ()
Способ использования: Mock-объект
Возвращаемое значение: коллекция объектов класса **Record**
Ожидаемый результат: Возвращаемое значение null.

Аналогично для других сервисов (далее приведены различия):
7.1
Класс: UserService
Зависимости: UserRepository
Выходные элементы: объект класса User с указанным идентификатором

7.2
Класс: ProjectService
Зависимости: ProjectRepository
Выходные элементы: объект класса Project с указанным идентификатором

7.3

Класс: ActivityService
Зависимости: ActivityRepository
Выходные элементы: объект класса Activity с указанным идентификатором

8.
Класс: WorkingTimeService
Метод: **IEnumerable<Record>** **GetUsersRecords(string developerId)**
Описание: проверка возврата коллекции записей данного пользователя, если БД содержит записи пользователя.
Тип теста: общий
Входные элементы: идентификатор разработчика **developerId** (существущий)
Выходные элементы: коллекция объектов класса Record
Зависимости:
Класс: RecordRepository
Метод: IQueryable<Record> GetRecords ()
Способ использования: Mock-объект
Возвращаемое значение: коллекция объектов класса **Record**
Ожидаемый результат: Возвращаемое значение – коллекция записей (с идентификатором пользователя – trestarita).

9.
Класс: WorkingTimeService
Метод: **IEnumerable<Record>** **GetUsersRecords(string developerId)**
Описание: проверка возврата пустой коллекции, если БД не содержит записи пользователя
Тип теста: негативный
Входные элементы: идентификатор разработчика **developerId**
Выходные элементы: коллекция объектов класса Record
Зависимости:
Класс: RecordRepository
Метод: IQueryable<Record> GetRecords ()
Способ использования: Mock-объект
Возвращаемое значение: коллекция объектов класса **Record**
Ожидаемый результат: Возвращаемое значение – пустая коллекция записей.

Интеграционое тестирование

1.
Класс: DefaultController
Метод: RedirectToRouteResult Index()
Описание: проверка обращения к методу **Index**, класса **RecordController,** если роль пользователя - **Developer**

Тип теста: общий
Входные элементы: объект – **User** (роль – **Developer**)
Выходные элементы: объект класса **RedirectToRouteResult**
Зависимости:
Класс: DefaultControllerHelper
Метод: User GetCurrentUser(IRecordController controller)
Возвращаемое значение: объект **User**, **UserRole** = Developer
Ожидаемый результат:
Тип возвращаемого значения: RedirectToRouteResult
Класс: RecordController
Метод: Index

2.
Класс: DefaultController
Метод: RedirectToRouteResult Index()
Описание: проверка обращения к методу к методу **Index**, класса **ManagementController** ,если роль пользователя - **Manager**
Тип теста: общий
Входные элементы: объект – **User** (роль – **Manager**)
Выходные элементы: объект класса **RedirectToRouteResult**
Зависимости:
Класс: DefaultControllerHelper
Метод: User GetCurrentUser(IRecordController controller)
Возвращаемое значение: объект **User**, **UserRole** = Manager
Ожидаемый результат:
Тип возвращаемого значения: RedirectToRouteResult
Класс: ManagementController
Метод: Index

3.
Класс: RecordController
Метод: ActionResult AddRecord(RecordFormModel recordFormModel)
Описание: проверка перенаправления на главную страницу (вне зависимости от валидности модели)
Тип теста: общий
Входные элементы: объект – **RecordFormModel (валидный)**
Выходные элементы: объект класса **ActionResult**
Зависимости:
Класс: RecordControllerHelper
Метод: User GetCurrentUser(IRecordController controller)
Возвращаемое значение: объект **User**

Класс: WorkingTimeService
Метод: void Log(Record record)
Ожидаемый результат:
Тип возвращаемого значения: ActionResult
Класс: RecordController

Метод: Index

3.1 Тест производится по аналогии, за исключением входных данных:
Входные элементы: объект – **RecordFormModel (не валидный)**

4.
Класс: RecordController
Метод: ActionResult AddRecord(RecordFormModel recordFormModel)
Описание: проверка обращения к **WorkingTimeService**если модель валидна
Входные элементы: объект – **RecordFormModel** (**ModelState** == Valid)
Выходные элементы: объект класса **ActionResult**
Зависимости:
Класс: RecordControllerHelper
Метод: GetCurrentUser(IRecordController controller)
Возвращаемое значение: объект **User**, **UserRole** = Manager

Класс: WorkingTimeService
Метод: void Log(Record record)
Ожидаемый результат: вызов метода Log класса **WorkingTimeService** с параметром – новой записью пользователя.

5.
Класс: RecordController
Метод: RedirectToRouteResult Delete(int recordId)
Описание: проверка обращения к **WorkingTimeService** методу **GetRecord** с целью удаления записи
Тип теста: общий
Входные элементы: идентификатор записи (recordId) (существующий)
Выходные элементы: объект класса **ActionResult**
Зависимости:
Класс: WorkingTimeService
Метод: Record GetRecord(int recordId))

Класс: RecordControllerHelper
Метод: User GetCurrentUser(IRecordController controller)
Возвращаемое значение: объект **User**

Ожидаемый результат: производится обращение к **WorkingTimeService** методу **GetRecord** с параметром “0”

6.
Класс: RecordController
Метод: RedirectToRouteResult Delete(int recordId)

Описание: проверка обращения к методу **Index**, класса **RecordController,** если запись не найдена (record == null)
Тип теста: негативный
Входные элементы: идентификатор записи (recordId) (несуществующий)
Выходные элементы: объект класса **ActionResult**
Зависимости:
Класс: WorkingTimeService
Метод: Record GetRecord(int recordId))

Класс: RecordControllerHelper
Метод: User GetCurrentUser(IRecordController controller)
Возвращаемое значение: объект **User**
Ожидаемый результат:
Тип возвращаемого значения: ActionResult
Класс: RecordController
Метод: Index

7.
Класс: RecordController
Метод: RedirectToRouteResult Delete(int recordId)
Описание: проверка обращения к методу **Index**, класса **RecordController,** если работник не является создателем записи
Тип теста: общий
Входные элементы: идентификатор записи (recordId) (существующий)
Выходные элементы: объект класса **ActionResult**
Зависимости:
Класс: WorkingTimeService
Метод: Record GetRecord(int recordId))

Класс: RecordControllerHelper
Метод: User GetCurrentUser(IRecordController controller)
Возвращаемое значение: объект **User** (record.UserId != user.UserId)
Ожидаемый результат:
Тип возвращаемого значения: ActionResult
Класс: RecordController
Метод: Index

8.
Класс: RecordController
Метод: ActionResult Edit(int recordId)
Описание: проверка обращения к **WorkingTimeService**, методу **GetRecord**
Тип теста: общий
Входные элементы: идентификатор записи (recordId) (существующий)

Выходные элементы: объект класса **ActionResult**
Зависимости:
Класс: WorkingTimeService
Метод: Record GetRecord(int recordId))

Класс: RecordControllerHelper
Метод: User GetCurrentUser(IRecordController controller)
Возвращаемое значение: объект **User**

Ожидаемый результат: производится обращение к **WorkingTimeService** методу **GetRecord** с параметром “0”

9.
Класс: RecordController
Метод: ActionResult Edit(int recordId)
Описание: проверка вызова **HttpException** с параметром “Forbidden”, если запись не найдена (record == null)
Тип теста: негативный
Входные элементы: идентификатор записи (recordId) (несуществующий)
Выходные элементы: объект класса **ActionResult**
Зависимости:
Класс: WorkingTimeService
Метод: Record GetRecord(int recordId))
Возвращаемое значение: объект **Record** (null)

Класс: RecordControllerHelper
Метод: User GetCurrentUser(IRecordController controller)
Возвращаемое значение: объект **User**
Ожидаемый результат:
Класс: HttpException
Сообщение: “Forbidden” (403)

10.
Класс: RecordController
Метод: ActionResult Edit(int recordId)
Описание: проверка вызова **HttpException** с параметром “Forbidden”, если работник не является создателем записи
Тип теста: негативный
Входные элементы: идентификатор записи (recordId) (существующий)
Выходные элементы: объект класса **ActionResult**
Зависимости:
Класс: WorkingTimeService
Метод: Record GetRecord(int recordId))

Класс: RecordControllerHelper
Метод: User GetCurrentUser(IRecordController controller)

Возвращаемое значение: объект **User** (record.UserId != user.UserId)

Ожидаемый результат:

Класс: HttpException

Сообщение: “Forbidden” (403)

11.

Класс: RecordController

Метод: ActionResult Edit(int recordId)

Описание: проверка обращения к **EditView**, если запись найдена и модель валидна

Тип теста: общий

Входные элементы: объект – **RecordFormModel** (**ModelState** == Valid)

Выходные элементы: объект класса **ActionResult**

Зависимости:

Класс: WorkingTimeService

Метод: Record GetRecord(int recordId))

Класс: RecordControllerHelper

Метод: User GetCurrentUser(IRecordController controller)

Возвращаемое значение: объект **User**

Ожидаемый результат: объект класса **ActionResult** (передаваемый параметр **EditViewModel**)

12.

Класс: RecordController

Метод: ActionResult EditRecord(RecordFormModel recrdFormModel)

Описание: проверка вызова **HttpException** с параметром “Forbidden”, если работник не является создателем записи

Тип теста: негативный

Входные элементы: объект – **RecordFormModel** (**ModelState** == Valid)

Выходные элементы: объект класса **ActionResult**

Зависимости:

Класс: WorkingTimeService

Метод: Record GetRecord(int recordId))

Возвращаемое значение: объект **Record**

Класс: RecordControllerHelper

Метод: User GetCurrentUser(IRecordController controller)

Возвращаемое значение: объект **User** (record.UserId != user.UserId)

Ожидаемый результат:

Класс: HttpException

Сообщение: “Forbidden” (403)

13.
Класс: RecordController
Метод: ActionResult EditRecord(RecordFormModel recrdFormModel)
Описание: проверка обращения к **WorkingTimeService**, методу **EditRecord,** если модель валидна
Входные элементы: объект – **RecordFormModel** (**ModelState** == Valid)
Выходные элементы: объект класса **ActionResult**
Зависимости:
Класс: WorkingTimeService
Метод: Record GetRecord(int recordId))
Возвращаемое значение: объект **Record**

Класс: RecordControllerHelper
Метод: User GetCurrentUser(IRecordController controller)
Возвращаемое значение: объект **User**
Ожидаемый результат: производится обращение к **WorkingTimeService** методу **EditRecord** с правильно сформированным параметром Record

14.
Класс: RecordController
Метод: ActionResult EditRecord(RecordFormModel recrdFormModel)
Описание: проверка перенаправления на **EditView**, если модель не валидна
Тип теста: общий
Входные элементы: объект – **RecordFormModel** (**ModelState** == NotValid)
Выходные элементы: объект класса **ActionResult**
Зависимости:
Класс: WorkingTimeService
Метод: Record GetRecord(int recordId))
Возвращаемое значение: объект **Record**

Класс: RecordControllerHelper
Метод: User GetCurrentUser(IRecordController controller)
Возвращаемое значение: объект **User**
Ожидаемый результат:
Тип возвращаемого значения: ActionResult
Класс: RecordController
Метод: Edit

16.
Класс: RecordController
Метод: ActionResult Index()

Описание: проверка обращения к **WorkingTimeService** методу **GetRecords**
Тип теста: общий
Входные элементы: идентификатор разработчика (существующий)
Выходные элементы: коллекция объектов класса **Record**
Зависимости:
Класс: WorkingTimeService
Метод: Record GetRecords(string developerId))
Возвращаемое значение: коллекция объектов класса **Record**

Класс: RecordControllerHelper
Метод: User GetCurrentUser(IRecordController controller)
Возвращаемое значение: объект **User**

Ожидаемый результат:
Производится обращение к **WorkingTimeService** методу **GetRecords** с параметром “trestarita”
Коллекция содержит две записи

17.
Класс: WorkingTimeService
Метод: void Delete(int recordId)
Описание: проверка обращения к репозиторию с целью удаления записи
Тип теста: общий
Входные элементы: идентификатор записи **recordId** (существующий)
Выходные элементы: -
Зависимости:
Класс: RecordRepository
Метод: void Delete (int recordId)
Возвращаемое значение: коллекция объектов класса **Record**
Ожидаемый результат: Произошло обращение к классу **RecordRepository**, методу **Delete**

Аналогично для других сервисов (далее приведены различия):
17.1
Класс: UserService
Зависимости: UserRepository

17.2
Класс: ProjectService
Зависимости: ProjectRepository

17.3

Класс: ActivityService
Зависимости: ActivityRepository

18.
Класс: WorkingTimeService
Метод: void Log(Record record)
Описание: проверка обращения к репозиторию с целью добавления новой записи
Тип теста: общий
Входные элементы: объект класса **Record**
Выходные элементы: -
Зависимости:
Класс: RecordRepository
Метод: void Add (Record record)
Ожидаемый результат: Произошло обращение к классу **RecordRepository**, методу **Add**

Аналогично для других сервисов (далее приведены различия):
18.1
Класс: UserService
Зависимости: UserRepository
Выходные элементы: объект класса User с указанным идентификатором

18.2
Класс: ProjectService
Зависимости: ProjectRepository
Выходные элементы: объект класса Project с указанным идентификатором

18.3
Класс: ActivityService
Зависимости: ActivityRepository
Выходные элементы: объект класса Activity с указанным идентификатором

19.
Класс: WorkingTimeService
Метод: void EditRecord(Record record)
Описание: проверка обращения к репозиторию с целью изменения записи
Тип теста: общий
Входные элементы: объект класса **Record**
Выходные элементы: -

Зависимости:
Класс: RecordRepository
Метод: void Edit (Record record)
Ожидаемый результат: Произошло обращение к классу **RecordRepository**, методу **Edit**

Аналогично для других сервисов (далее приведены различия):
19.1
Класс: UserService
Зависимости: UserRepository
Входные элементы: объект класса User

19.2
Класс: ProjectService
Зависимости: ProjectRepository
Входные элементы: объект класса Project

19.3
Класс: ActivityService
Зависимости: ActivityRepository
Входные элементы: объект класса Activity

20.
Класс: WorkingTimeService
Метод: Record GetRecord(int recordId)
Описание: проверка обращение к репозиторию с целью получения записей
Тип теста: общий
Входные элементы: идентификатор записи **recordId** (существующий)
Выходные элементы: объект класса Record с указанным идентификатором
Зависимости:
Класс: RecordRepository
Метод: IQueryable<Record> GetRecords ()
Возвращаемое значение: коллекция объектов класса **Record**
Ожидаемый результат: Произошло обращение к классу **RecordRepository**, методу **GetRecords.**

Аналогично для других сервисов (далее приведены различия):
20.1
Класс: UserService
Зависимости: UserRepository

Входные элементы: объект класса User с указанным идентификатором

20.2
Класс: ProjectService
Зависимости: ProjectRepository
Входные элементы: объект класса Project с указанным идентификатором

20.3
Класс: ActivityService
Зависимости: ActivityRepository
Входные элементы: объект класса Activity с указанным идентификатором

21.
Класс: WorkingTimeService
Метод: **IEnumerable<Record>** **GetUsersRecords(string developerId)**
Описание: проверка обращения к репозиторию с целью получения всех записей.
Тип теста: общий
Входные элементы: идентификатор разработчика **developerId**
Выходные элементы: коллекция объектов класса Record
Зависимости:
Класс: RecordRepository
Метод: IQueryable<Record> GetRecords ()
Возвращаемое значение: коллекция объектов класса **Record**
Ожидаемый результат: Произошло обращение к классу **RecordRepository**, методу **GetRecords.**

Аттестационное тестирование

1.
Описание: создание записи работника
Входные элементы: введена дата за прошлый месяц
Тип теста: негативный
Ожидаемый результат: модель не валидна, уведомление об ошибке

2.
Описание: создание записи работника
Входные элементы: введена дата за следующий месяц
Тип теста: негативный
Ожидаемый результат: модель не валидна, уведомление об ошибке

3.
Описание: создание записи работника
Входные элементы: введена дата за текущий месяц
Тип теста: общий
Ожидаемый результат: модель валидна, запись сохранена

4.
Описание: создание записи работника
Входные элементы: введена дата последнего дня текущего месяца
Тип теста: краевой
Ожидаемый результат: модель валидна, запись сохранена

5.
Описание: создание записи работника
Входные элементы: введена дата первого дня текущего месяца
Тип теста: краевой
Ожидаемый результат: модель валидна, запись сохранена

6.
Описание: создание записи работника
Входные элементы: введена дата: 29 февраля 2012
Тип теста: краевой
Ожидаемый результат: модель не валидна, уведомление об ошибке

7.
Описание: создание записи работника
Входные элементы: значение даты пусто
Тип теста: негативный
Ожидаемый результат: модель не валидна, уведомление об ошибке

8.
Описание: создание записи работника
Входные элементы: введено отрицательное значение рабочих часов
Тип теста: негативный
Ожидаемый результат: модель не валидна, уведомление об ошибке

9.
Описание: создание записи работника
Входные элементы: введено значение рабочих часов больше 12
Тип теста: негативный

Ожидаемый результат: модель не валидна, уведомление об ошибке

10.
Описание: создание записи работника
Входные элементы: введено значение рабочих часов равное 0
Тип теста: краевой
Ожидаемый результат: модель валидна, запись сохранена

11.
Описание: создание записи работника
Входные элементы: введено значение рабочих часов равное 12
Тип теста: краевой
Ожидаемый результат: модель валидна, запись сохранена

12.
Описание: создание записи работника
Входные элементы: введено значение рабочих часов равное 8
Тип теста: общий
Ожидаемый результат: модель валидна, запись сохранена

13.
Описание: создание записи работника
Входные элементы: значение рабочих часов пусто
Тип теста: негативный
Ожидаемый результат: модель не валидна, уведомление об ошибке

14.
Описание: открытие страницы менеджера, используя аккаунт разработчика
Тип теста: негативный
Ожидаемый результат: исключение 403

Покрытие исполняемого кода тестами

Количество строк кода тестируемых модулей: 1145;

Количество строк кода, покрытых тестами 946;

Покрытие = (946 / 1145) * 100% = 82,6 %

Примеры реализации тестов

Для запуска тестов использовалась платформа модульных тестов Microsoft.

1. Модульное тестирование

```
/// <summary>
/// Working time service should require records of current user from RecordRepository.
/// </summary>
[TestMethod]
public void WorkingTimeServiceShouldRequireRecordsOfCurrentUserfromREcordRepository()
{
    // Arrange
    WorkingTimeService service = new WorkingTimeService(this.repository);
    List<Record> records = new List<Record>()
    {
        new Record()
        {
            RecordId = 0,
            DeveloperId = "user"
        }
    };
    var expectedRecord = new Record()
    {
        RecordId = 0,
        DeveloperId = "user"
    };

    Mock.Get(this.repository)
        .Setup(r => r.GetRecords())
        .Returns(records.AsQueryable());

    // Act
    var record = service.GetUsersRecords("user");

    // Assert
    Mock.Get(this.repository).Verify(r => r.GetRecords());
    Assert.IsTrue(expectedRecord.RecordId == record.First().RecordId);
    Assert.AreEqual(1, record.Count());
}
```

2. Интеграционное

```
/// <summary>
/// EditRecord() action should send recordId to RecordRepository.
/// </summary>
[TestMethod]
public void EditRecordShouldSendRecordIdToRepository()
{
    // Arrange
    WorkingTimeService service = new WorkingTimeService(this.repository);
    Record record = new Record
    {
        RecordId = 0,
        Date = new DateTime(2013, 10, 9),
        WorkingTime = 5,
        ProjectId = 6,
        ActivityId = 7
    };

    // Act
    service.EditRecord(record);

    // Assert
    Mock.Get(this.repository).Verify(r => r.Edit(record));
}
```

Результаты тестирования

Номера тестов	Класс	Метод	Количество тестов	Количество ошибок	Дата проведения
	Модульное тестировние				
4,5	**WorkingTimeService**	Get	2	0	29.12.2014
6,7	**WorkingTimeService**	GetGetRecord	2	0	29.12.2014
8,9	**WorkingTimeService**	GetUsersRecords	2	1 (см. Найденные ошибки №1)	29.12.2014
4.1, 5.1	**UserService**	Get	1	0	29.12.2014
6.1,7.1	**UserService**	GetGetRecord	2	0	29.12.2014
4.2, 5.2	**ActivityService**	Get	1	0	29.12.2014
6.2,7.2	**ActivityService**	GetGetRecord	2	0	29.12.2014
4.3, 5.3	**ProjectService**	Get	1	0	29.12.2014
6.3,7.3	**ProjectService**	GetGetRecord	2	0	29.12.2014
1..3	**AllowAccessAttribute**	IsInRole	3	0	29.12.2014
	Интеграционное тестирование				
1,2	**DefaultController**	Index	2	0	29.12.2014
3,3.1,4	**DefaultController**	AddRecord	3	0	29.12.2014
5-7	**DefaultController**	Delete	3	0	29.12.2014
8-11	**DefaultController**	Edit	4	0	29.12.2014
12-15	**DefaultController**	EditRecord	5	0	29.12.2014
16	**DefaultController**	Index	1	0	29.12.2014
17	**WorkingTimeService**	Delete	1	0	29.12.2014
18	**WorkingTimeService**	Log	1	0	29.12.2014
19	**WorkingTimeService**	EditRecord	1	0	29.12.2014
20	**WorkingTimeService**	GetRecord	1	0	29.12.2014
21	**WorkingTimeService**	GetUsersRecords	1	0	29.12.2014
17.1	**UserService**	Delete	1	0	29.12.2014
18.1	**UserService**	Log	1	0	29.12.2014
19.1	**UserService**	EditRecord	1	0	29.12.2014
20.1	**UserService**	GetRecord	1	0	29.12.2014
17.2	**ActivityService**	Delete	1	0	29.12.2014
18.2	**ActivityService**	Log	1	0	29.12.2014
19.2	**ActivityService**	EditRecord	1	0	29.12.2014
20.2	**ActivityService**	GetRecord	1	0	29.12.2014
17.3	**ProjectService**	Delete	1	0	29.12.2014
18.3	**ProjectService**	Log	1	0	29.12.2014
19.3	**ProjectService**	EditRecord	1	0	29.12.2014
20.3	**ProjectService**	GetRecord	1	0	29.12.2014

Аттестационное тестирование		
Номер теста	**Результат**	**Дата проведения**
1	провален (см. Найденные ошибки №2)	29.12.2014
2	провален (см. Найденные ошибки №2)	29.12.2014
3	пройден	29.12.2014
4	пройден	29.12.2014
5	пройден	29.12.2014
6	пройден	29.12.2014
7	пройден	29.12.2014
8	пройден	29.12.2014
9	пройден	29.12.2014
10	пройден	29.12.2014
11	пройден	29.12.2014
12	пройден	29.12.2014
13	пройден	29.12.2014
14	пройден	29.12.2014

Найденные ошибки.

1. Класс **WorkingTimeService** метод **GetUsersRecords:** в качестве возврщаемого значения выступает коллекция записей. Если записей нет, то должна возвращаться пустая коллекция. В данной реализации производилось возвращение null, если коллекция пуста, что приводило к необработанной ошибке.
2. Одной из рекомендаций к валидации введенных пользователем данных была проверка даты. Публикуемая запись не может принадлежать прошлому или следующему месяцу. Такая проверка отсутствует в данной версии системы.

В целом система прошла тестирование успешно. Большинство модулей работают согласно требованиям. Модули, в которых наблюдается некорректное поведение, нуждаются в доработке.