Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования Петрозаводский государственный университет Факультет математики и информационных технологий
Кафедра информатики и математического обеспечения

Отчет по дисциплине «Верификация ПО»

Выполнил:
Студент группы 22608 С.А. Сущевич

_____

подпись

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

_____

Подпись

Петрозаводск
2016

СОДЕРЖАНИЕ

Описание объекта тестирования

Объектом тестирования является веб-приложение, предназначенное для учета рабочего времени работника, создания отчетов об их деятельности, и над их управлением.

Графически схему взаимодействия можно представить следующим образом, см.рисунок 1.

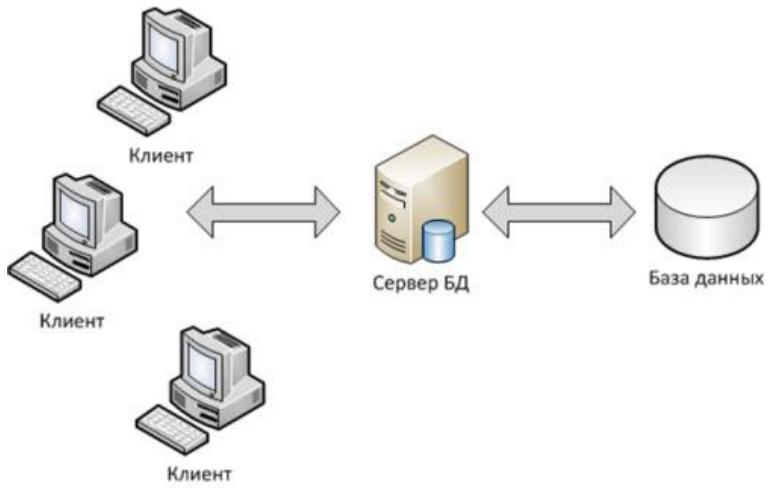

Рисунок 1 Графическая схема взаимодействия компонентов веб-приложения

Для реализации приложения использовались следующие технологии: Java, CSS, HTML5, MySQL.

Функциональность объекта тестирования

1. В системе допустимы два вида пользователей: работник и директор - требуется произвести проверку разграничения прав доступа, т.к. функционал, доступный директору, не должен быть доступен работнику;
2. Работник и директор могут логировать рабочие часы с указанием:
   - Даты;
   - Вида совершаемой деятельности;
   - Затраченного времени;
3. Работники имеют возможность просматривать итоговый отчет о затраченном времени, редактировать и удалять записи о виде деятельности.
4. Директор может управлять командой работников:

- Добавлять информацию о новых работниках (идентификатор, имя и роль работника);
- Редактировать информацию о существующих работниках;
- Удалять информацию о существующих работниках;

5. Директор имеют возможность управлять видами деятельности (создавать, редактировать и удалять виды деятельности);
6. Директор имеет возможность генерировать, просматривать и скачивать отчет в определенном формате за выбранный промежуток времени. *Каждая запись отчета включает в себя следующую информацию:*

- Имя работника;
- 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;

Стратегия тестирования

Архитектура приложения

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.

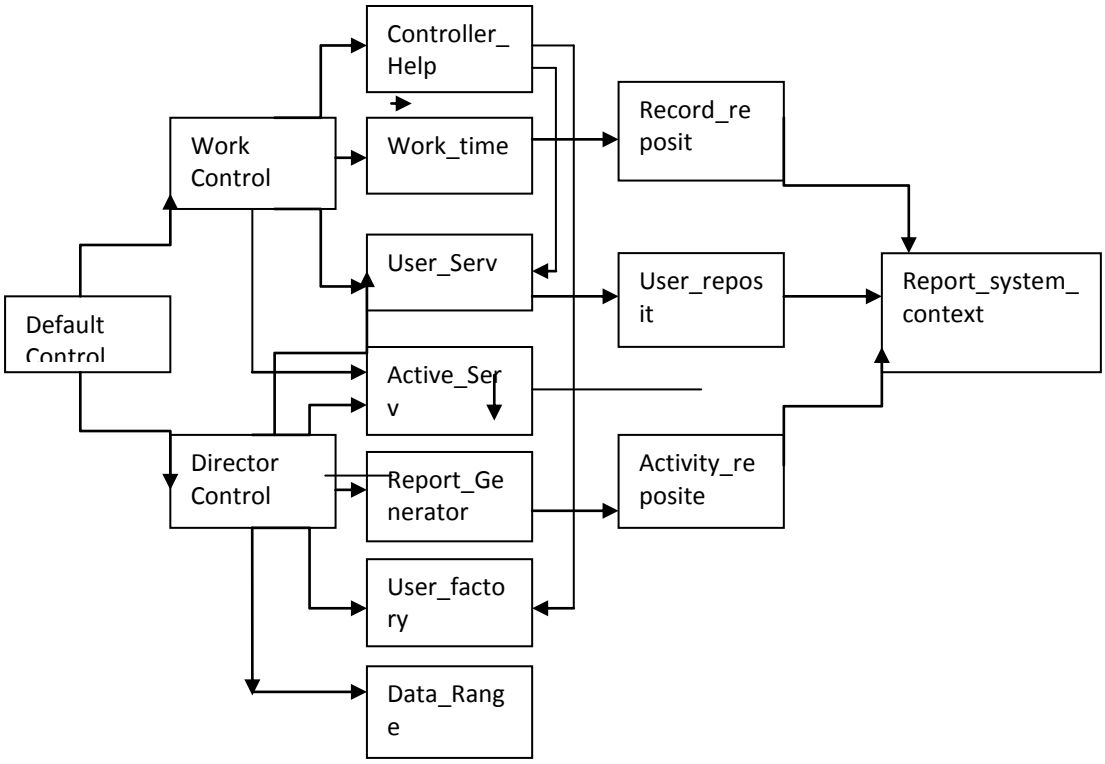

Рисунок 2 Графическая схема взаимодействия компонентов веб-приложения

Описание элементов системы

1) **Default_Controll** – точка входа в систему. На данном этапе производится проверка роли пользователя и происходит перенаправление на **Work_Control** (если пользователь является работником) или на **Director_Control** (если пользователь является директором).

Метод:

RedirectToRouteResult Index() – получает роль пользователя и производит перенаправление на **Work** или на **Director**

***Тестирование**: модульное, интеграционное, аттестационное (проверить взаимодействие с Work_Control(1).*

2) **Work_Control** – обращается к сервисам, позволяющим работникам добавлять, редактировать и удалять записи, а также просматривать отчет затраченного времени.

Методы:

RedirectToRouteResult Delete( ) - обращается к **Work_time** для удаления записи.

ActionResult Edit( ) - обращается к **Work_time** для получения записи для редактирования.

ActionResult EditRecord( ) – если модель валидна, то обращается к **Work_time** для сохранения изменений, иначе производит перенаправление на заполнение формы**.**

***Тестирование**: модульное, аттестационное.*

3) **Director_Control** – обращается к сервисам, позволяющим директору управлять работниками, и видами деятельности, создавать, скачивать и просматривать отчеты за определенный промежуток времени.

***Тестирование**: невозможно провести тестирование, так как элемент недоступен.*

4) **Control_Help** – предоставляет информацию о текущем пользователе, взаимодействующим с системой.

***Тестирование**: модульное, интеграционное (проверить взаимодействие с Work_Control,User_serv).*

5) **Work_time** – обращается к **Work_Control** с запросом получить все (или определенную) записи текущего работника, добавить, удалить или отредактировать запись.

Методы:

void Log( ) – обращается к репозиторию для сохранения записи.

void Delete( ) – обращается к репозиторию для удаления записи.

Record GetRecord( ) – обращается к репозиторию для получения свойств записи.

IEnumerable<Record> GetUsersRecords( ) – обращается к репозиторию для получения записей пользователя.

IEnumerable<Record> Get( ) – обращается к репозиторию для получения всех записей и возвращает их.

Тестирование*: модульное, интеграционное (проверить взаимодействие с Work_Control, Record_reposit).*

6) **User_Serv** – обращается к **User_Reposit** с просьбой получить всех пользователей системы, добавить нового пользователя, отредактировать или удалить информацию о существующих пользователях.

Тестирование *осуществляется обоюдно с тестированием других элементов.*

7) **Activite_serv -** обращается к **Active_reposit** с просьбой получить все виды деятельности, добавить новый, отредактировать или удалить информацию о существующих видах деятельности.

Тестирование *осуществляется обоюдно с тестированием других элементов.*

8) **Report_Generator** – позволяет генерировать финальный отчет за определенный промежуток времени.

Тестирование*: невозможно провести тестирование, так как элемент недоступен.*

9) **User_factory** – позволяет создавать экземпляр профиля работника.

Тестирование*: невозможно провести тестирование, так как элемент недоступен.*

10) **Date_Range**– позволяет сформировать начальную и конечную дату для создания финального отчета.

Тестирование*: невозможно провести тестирование, так как элемент недоступен.*

11)**Record_reposit** – через контекст обращается к БД для работы с данными о записях пользователя.

Тестирование *осуществляется обоюдно с тестированием других элементов.*

12) **User_reposit** – через контекст обращается к БД для работы с данными о пользователях.

Тестирование *осуществляется обоюдно с тестированием других элементов.*

13)**Activity_reposit** – через контекст обращается к БД для работы с данными о видах деятельности.

Тестирование *осуществляется обоюдно с тестированием других элементов.*

14) Report System Context – контекст работы с БД.

15) **AllowAccessAttribute –** перенаправляет пользователя на соответствующую страницу.

Метод:

bool IsInRole( ) – проверяет имеет ли пользователь права доступа к запрашиваемой странице.

Тестирование: *модульное (проверить взаимодействие с Work_Control, Director_Control).*

Стратегия тестирования

Корректное значение переменных, участвующих в тестировании:

Числовое значение 0-50; дата в формате DD.MM.YY; пустое значение;

Модульное тестирование.

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

Объектами модульного тестирования выбраны:

Класс: AllowAccessAttribute

Метод: bool IsInRole.

Класс: Work_time

Метод: IEnumerable<Record> Get(),Record GetRecord(),IEnumerable<Record> GetUsersRecords.

Класс: Active_serv

Метод: Get(), GetUsersRecords ()

Класс: User_service

Метод: Get(), GetUsersRecords()

Интеграционное тестирование.

Для проведения интеграционного тестирования применяется стратегия нисходящего тестирования.

Класс DefaultControl + класс Direct_Controll + класс Data_Range;

Класс DefaultControl + класс Direct_Controll + класс User_factory;

Класс DefaultControl + класс Direct_Controll + класс Report_Generation + класс Activity_reposite + класс Report_system_context;

Класс DefaultControl + класс Direct_Controll + класс Active_serv + класс Activity_reposite + класс Report_system_context;

Класс DefaultControl + класс Direct_Controll + класс User_serv + класс User_reposite + класс Report_system_context;

Класс DefaultControl + класс Work_Controll + класс User_serv + класс User_reposite + класс Report_system_context;
Класс DefaultControl + класс Work_Controll + класс Work_time + класс Record_reposit + класс Report_system_context;
Класс DefaultControl + класс ControllerHelp + класс Work_time + класс Record_reposit + класс Report_system_context;

- Функция RedirectToRouteResult метод Index() класс DefaultControl + класс Work_Controll метод Index() + класс Controller_Help метод GetCurrentUser()
- Функция RedirectToRouteResult метод Index() класс DefaultControl + класс Director_Controll метод Index() + класс Controller_Help метод GetCurrentUser()
- Функция ActionResult метод AddRecord () класс Work_Control + Work_time + класс Controller_Help функция User GetCurrentUser()
- Функция ActionResult метод Edit() класса Work_Control + класс Work_time метод Log ()+ класс Controller_Help метод GetCurrentUser ()
- Функция ActionResult метод EditRecord () класса Work_Control + класс HttpException + класс Work_time метод GetRecord ()
- метод Delete () класса Work_Control + класс Record_reposit метод Delete ()
- Функция Record метод Log() класса UserService+класс Record_reposit метод Add()+User_reposit метод Add()
- метод Log() класса Activity_serv+класс Record_reposit метод Add()+Activity_reposite метод Add()
- метод EditRecord() класса Work_time+класс Record_reposit метод Edit()+Record_reposit метод Edit()
- метод GetUsersRecords () класса Work_time+класс Record_reposit метод GetRecords ()+Record_reposit метод GetRecords ()

- Функция RedirectToRouteResult метод Delete() класс Work_Control + класс Work_Time метод GetRecord() + класс Work_time метод GetRecord ()
- Функция ActionResult метод Edit() класс Work_Control + класс Work_Time метод GetRecord() + класс Work_time метод GetRecord ()+класс Controller_Help метод GetCurrentUser()

Аттестационное тестирование.

Аттестационное тестирование проводится для всей системы, что подразумевает выполнение действий в пользовательском интерфейсе.

Для совершения процедуры тестирования необходимы:

Браузер (Google Chrome)

Java 8.0

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.

Детальный план тестов

Модульное тестирование

1. Тестирование метода **IsInRole** класса **AllowAccessAttribute.**

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

Способ использования: Mock-объект

Возвращаемое значение: коллекция объектов класса **User**

Ожидаемый результат: Пользователь не найден, создан новый пользователь

2. Тестирование метода **IsInRole** класса **AllowAccessAttribute.**

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

Способ использования: Mock-объект

Возвращаемое значение: коллекция объектов класса **User**

Ожидаемый результат: Пользователь найден, возвращаемое значение true

3. Тестирование метода **IsInRole** класса **AllowAccessAttribute.**

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:

Использования: Mock-объект

Возвращаемое значение: коллекция объектов класса **User**

Ожидаемый результат: Пользователь найден, HttpException 403 (Forbidden)

4. Тестирование метода **Get()** класса **Work_time.**

Описание: проверка создания коллекции записей, если записи есть

Тип теста: общий

Входные элементы: -

Выходные элементы: коллекция записей

Способ использования: Mock-объект

Возвращаемое значение: коллекция объектов класса **Record**

Ожидаемый результат: Коллекция объектов класса **Record**; Коллекция из 2 записей

4.1 Тестирование метода **Get()** класса **User_service.**

Описание: проверка создания коллекции записей, если записи есть

Тип теста: общий

Входные элементы: -

Выходные элементы: коллекция записей

Способ использования: Mock-объект

Возвращаемое значение: коллекция объектов класса **Record**

Ожидаемый результат: Коллекция объектов класса **User**; Коллекция из 2 записей

4.2 Тестирование метода **Get()** класса **Active_Serv.**

Описание: проверка создания коллекции записей, если записи есть

Тип теста: общий

Входные элементы: -

Выходные элементы: коллекция записей

Способ использования: Mock-объект

Возвращаемое значение: коллекция объектов класса **Record**

Ожидаемый результат: Коллекция объектов класса **Activity**; Коллекция из 2 записей

5. Тестирование метода **Get()** класса **Work_time.**

Описание: проверка создания пустой коллекции, если записей нет

Тип теста: специальный

Входные элементы: -

Выходные элементы: коллекция записей

Способ использования: Mock-объект

Возвращаемое значение: пустая коллекция объектов класса **Record**

Ожидаемый результат: Пустая коллекция

5.1 Тестирование метода **Get()** класса **Active_Serv.**

Описание: проверка создания пустой коллекции, если записей нет

Тип теста: специальный

Входные элементы: -

Выходные элементы: коллекция записей

Способ использования: Mock-объект

Возвращаемое значение: пустая коллекция объектов класса **Record**

Ожидаемый результат: Пустая коллекция

6. Тестирование метода **GetRecord ()** класса **Work_time.**

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: Record_reposit

Метод: IQueryable<Record> GetRecords ()

Способ использования: Mock-объект

Возвращаемое значение: коллекция объектов класса **Record**

Ожидаемый результат: Возвращаемое значение – запись с идентификатором “8”.

6.1 Тестирование метода **GetRecord ()** класса **User_serv.**

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

Способ использования: Mock-объект

Возвращаемое значение: коллекция объектов класса **Record**

Ожидаемый результат: Возвращаемое значение – запись с идентификатором “8”.

6.2 Тестирование метода **GetRecord ()** класса **Active_serv.**

Описание: проверка создания объекта, если запись с указанным идентификатором существует

Тип теста: общий

Входные элементы: идентификатор записи **recordId** (существующий)

Выходные элементы: объект класса Active с указанным идентификатором

Зависимости: Класс: Record_reposit

Метод: IQueryable<Record> GetRecords ()

Способ использования: Mock-объект

Возвращаемое значение: коллекция объектов класса **Record**

Ожидаемый результат: Возвращаемое значение – запись с идентификатором “8”.

7. Тестирование метода **GetRecord ()** класса **Work_time.**

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

Способ использования: Mock-объект

Возвращаемое значение: коллекция объектов класса **Record**

Ожидаемый результат: Возвращаемое значение null.

7.1 Тестирование метода **GetRecord ()** класса **User_serv.**

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 User с указанным идентификатором

Способ использования: Mock-объект

Возвращаемое значение: коллекция объектов класса **Record**

Ожидаемый результат: Возвращаемое значение null.

7.2 Тестирование метода **GetRecord ()** класса **Active_Serv.**

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 Activity с указанным Идентификатором

Способ использования: Mock-объект

Возвращаемое значение: коллекция объектов класса **Record**

Ожидаемый результат: Возвращаемое значение null.

8. Тестирование метода **GetUsersRecords ()** класса **Work_time.**

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

Способ использования: Mock-объект

Возвращаемое значение: коллекция объектов класса **Record**

Ожидаемый результат: Возвращаемое значение – коллекция записей (с идентификатором пользователя – trestarita).

9. Тестирование метода **GetUsersRecords ()** класса **Work_time.**

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

Способ использования: Mock-объект

Возвращаемое значение: коллекция объектов класса **Record**

Ожидаемый результат: Возвращаемое значение – пустая коллекция записей.

Интеграционное тестирование

1. Проверка взаимодействия RedirectToRouteResult Index() класса **DefaultControl** к методу Index класса **Work_Controll**.

Зависимости:

Класс: DefaultControlHelp

Метод: User GetCurrentUser(IRecordControl controller)

Возвращаемое значение: объект User, UserRole = Working

Описание	Входные данные	Выходные данные	Ожидаемый результат	Тип теста
Обращение к методу Index, если роль пользователя Working.	объект – **User** (роль – **Working**)	объект класса **RedirectToRouteResult**	Возвращает значение **RedirectToRouteResult** класса Work_Controll.	Общий

2. Проверка взаимодействия RedirectToRouteResult Index() класса **DefaultControl** к методу Index класса Director_Controll.

Зависимости:

Класс: DefaultControlHelp

Метод: User GetCurrentUser(IRecordControl controller) **Возвращаемое значение:** объект User, UserRole = Director

Описание	Входные данные	Выходные данные	Ожидаемый результат	Тип теста
Обращение к методу Index, если роль пользователя **Director**.	объект – **User** (роль – **Director**)	объект класса **RedirectToRouteResult**	Возвращает значение **RedirectToRouteResult** класса Director_Controll.	Общий

3. Проверка работоспособности **ActionResult AddRecord** (**RecordFormModel recordFormModel**) класса **Work_Control.**

Зависимости: Класс: RecordControllerHelper

Метод: User GetCurrentUser(IRecordController controller)

Возвращаемое значение: объект User

Класс: WorkingTimeService

Метод: void Log(Record record)

Описание	Входные данные	Выходные данные	Ожидаемый результат	Тип теста
Проверка перенаправления на главную страницу (вне зависимости от валидности модели)	объект – **RecordFormModel (валидный)**	объект класса **ActionResult**	Возвращает значение **ActionResult** класса Work_Controll.	Общий

4. Проверка взаимодействия метода **ActionResult AddRecord(RecordFormModel recordFormModel)** класса **Work_Control** с классом Work_time.

Зависимости:

Класс: RecordControlHelp

Метод: GetCurrentUser(IRecordControl controller)

Возвращаемое значение: объект User, UserRole = Director

Класс: Work_time

Метод: void Log(Record record)

Описание	Входные данные	Выходные данные	Ожидаемый результат	Тип теста
Проверка	объект –	объект класса	вызов метода Log	Общий

обращения к **Work_time** если модель валидна	**RecordFormModel** (**ModelState** == Valid)	**ActionResult**	класса **Work_time** с параметром – новой записью пользователя.	

5. Проверка взаимодействия метода **RedirectToRouteResult Delete(int recordId)** класса **Work_Control** с методом **GetRecord** класса **Work_time.**

Зависимости:

Класс: Work_time

Метод: Record GetRecord(int recordId))

Класс: RecordControlHelp

Метод: User GetCurrentUser(IRecordControl controller)

Возвращаемое значение: объект User

Описание	Входные данные	Выходные данные	Ожидаемый результат	Тип теста
Проверка обращения к **Work_time** если модель валидна	Идентификатор записи (recordId) (существующий)	объект класса **ActionResult**	производится обращение к **Work_time** методу **GetRecord** с параметром “0”	Общий

6. Проверка взаимодействия метода **RedirectToRouteResult Delete(int recordId)** класса **Work_Control** с методом **Index**, класса **Work_Control.**

Зависимости:

Класс: Work_time

Метод: Record GetRecord(int recordId))

Класс: RecordControlHelp

Метод: User GetCurrentUser(IRecordControl controller)

Возвращаемое значение: объект **User**

Описание	Входные	Выходные	Ожидаемый	Тип теста

	данные	данные	результат	
Проверка обращения к методу **Index**, класса **RecordControl,** если запись не найдена (record == null)	Идентификатор записи (recordId) (существующий)		возвращаемое значение ActionResult	негативный

7. Проверка взаимодействия метода **RedirectToRouteResult Delete(int recordId)** класса **Work_Control** с методом **Index**, класса **Work_Control.**

Зависимости:

Класс: Work_time

Метод: Record GetRecord(int recordId))

Класс: RecordControlHelp

Метод: User GetCurrentUser(IRecordControl controller)

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

Описание	Входные данные	Выходные данные	Ожидаемый результат	Тип теста
проверка обращения к методу **Index**, класса **RecordControl,** если работник не является создателем записи	Идентификатор записи (recordId) (существующий)		возвращаемое значение ActionResult	общий

8. Проверка взаимодействия метода **ActionResult Edit(int recordId)** класса **Work_Control** с методом **GetRecord**, класса **Work_time.**

Зависимости:

Класс: Work_time

Метод: Record GetRecord(int recordId))

Класс: Controller_Help

Метод: User GetCurrentUser(IRecordControl controller)

Возвращаемое значение: объект User

Описание	Входные данные	Выходные данные	Ожидаемый результат	Тип теста
проверка обращения к **Work_time**, методу **GetRecord**	Идентификатор записи (recordId) (существующий)		производится обращение к **Workin_time** методу **GetRecord** с параметром “0”	общий

9. Проверка взаимодействия метода **ActionResult Edit(int recordId)** класса **Work_Control** с классом **HttpException.**

Зависимости:

Класс: Work_time

Метод: Record GetRecord(int recordId))

Возвращаемое значение: объект **Record** (null)

Класс: RecordControlHelp

Метод: User GetCurrentUser(IRecordControl controller)

Возвращаемое значение: объект **User**

Описание	Входные данные	Выходные данные	Ожидаемый результат	Тип теста
проверка вызова **HttpException** с параметром	Идентификатор записи (recordId) (не существу	объект класса **ActionResult**	“Forbidden” (403)”	негативный

“Forbidden”, если запись не найдена (record == null)	ющий)			

10. Проверка взаимодействия метода **ActionResult Edit(int recordId)** класса **Work_Control** с классом **HttpException.**

Зависимости:

Класс: Work_time

Метод: Record GetRecord(int recordId))

Возвращаемое значение: объект **Record** (null)

Класс: RecordControlHelp

Метод: User GetCurrentUser(IRecordControl controller) Возвращаемое значение: объект **User** (record.UserId != user.UserId)

Описание	Входные данные	Выходные данные	Ожидаемый результат	Тип теста
проверка вызова **HttpException** с параметром “Forbidden”, если работник не является создателем записи	Идентификатор записи (recordId) (существующий)		“Forbidden” (403)”	негативный

11. Проверка взаимодействие метода **ActionResult Edit(int recordId)** класса **Work_Control с** объектом класса **ActionResult.**

Зависимости:

Класс: Work_time

Метод: Record GetRecord(int recordId))

Класс: RecordControlHelp

Метод: User GetCurrentUser(IRecordControl controller)

Возвращаемое значение: объект **User**

Описание	Входные данные	Выходные данные	Ожидаемый результат	Тип теста
проверка обращения к **EditView**, если запись найдена и модель валидна	объект – **RecordFormModel** (**ModelState** == Valid)	объект класса **ActionResult**	Объект класса **ActionResult** (передаваемый параметр **EditViewModel**)	общий

12. Проверка взаимодействия метода **EditRecord(RecordFormModel recrdFormModel)** класса **Work_Control** с классом **HttpException.**

Зависимости:

Класс: Work_time

Метод: Record GetRecord(int recordId))

Возвращаемое значение: объект **Record**

Класс: RecordControlHelp

Метод: User GetCurrentUser(IRecordControl controller) Возвращаемое значение: объект **User** (record.UserId != user.UserId)

Описание	Входные данные	Выходные данные	Ожидаемый результат	Тип теста
проверка вызова **HttpException** с параметро	объект – **RecordFormModel** (**ModelState** ==	объект класса **ActionResult**	Сообщение «Forbidden” 403	негативный

м “Forbidden ”, если работник не является создателем записи	Valid)			

13. Проверка взаимодействия метода **ActionResult EditRecord(RecordFormModel recrdFormModel)recrdFormModel)** класса **Work_Control** с классом **Work_time** методу **EditRecord.**

Зависимости:

Класс: Work_time

Метод: Record GetRecord(int recordId))

Возвращаемое значение: объект **Record**

Класс: RecordControlHelp

Метод: User GetCurrentUser(IRecordControl controller)

Возвращаемое значение: объект **User**

Описание	Входные данные	Выходные данные	Ожидаемый результат	Тип теста
проверка обращения к **Work_time** , методу **EditRecord,** если модель валидна	объект – **RecordFormModel** (**ModelState** == Valid)	объект класса **ActionResult**	производится обращение к **Work_time** методу **EditRecord** с правильно сформированным параметром Record	негативный

14. Проверка взаимодействия метода **ActionResult EditRecord(RecordFormModel recrdFormModel)** класса **Work_Control** с классом **Work_Control** методу **Edit.**

Зависимости:

Класс: Work_time

Метод: Record GetRecord(int recordId))

Возвращаемое значение: объект **Record**

Класс: RecordControlHelp

Метод: User GetCurrentUser(IRecordControl controller)

Возвращаемое значение: объект **User**

Описание	Входные данные	Выходные данные	Ожидаемый результат	Тип теста
проверка перенаправления на **EditView**, если модель не валидна	объект – **RecordFormModel** (**ModelState** == Valid)	объект класса **ActionResult**	Возвращаемое значение ActionResult метода Edit	общий

15. Проверка взаимодействия метода **ActionResult Index()** класса **Work_Control** с классом **Work_time** методу **GetRecords.**

Зависимости:

Класс: Work_time

Метод: Record GetRecords(string developerId)) Возвращаемое значение: коллекция объектов класса **Record**

Класс: RecordControlHelp

Метод: User GetCurrentUser(IRecordControl controller)

Возвращаемое значение: объект **User**

Описание	Входные данные	Выходные данные	Ожидаемый результат	Тип теста
проверка обращения к **Work_time** методу **GetRecords**	идентификатор (существующий)	коллекция объектов класса **Record**	Производится обращение к **Work_time** методу **GetRecords** с параметром “trestarita” Коллекция содержит две записи	общий

16. Проверка взаимодействия метода **void Delete(int recordId)** класса **Work_time** с классом **Record_reposit** методу **Delete.**

Зависимости:

Класс: Record_reposit

Метод: void Delete (int recordId)

Возвращаемое значение: коллекция объектов класса **Record**

Описание	Входные данные	Выходные данные	Ожидаемый результат	Тип теста
проверка обращения к репозиторию с целью удаления записи	Идентификатор записи RecordId (суще		Произошло обращение к классу **Record_reposit**, методу **Delete**	общий

	ствую щий)			

17. Проверка взаимодействия метода **void Log(Record record)**класса **Work_time** с классом **Record_reposit** методу **Add.**

Зависимости:

Класс: Record_reposit

Метод: void Add (Record record)

Описание	Входные данные	Выходные данные	Ожидаемый результат	Тип теста
проверка обращения к репозиторию с целью добавления новой записи	объект класса **Record**		Произошло обращение к классу **Record_reposit**, методу **Add**	общий

18. Проверка взаимодействия метода **void Log(Record record)**класса **UserService** с классом **Record_reposit** методу **Add.**

Зависимости:

Класс: User_reposit

Метод: void Add (Record record)

Описание	Входные	Выходные	Ожидаемый	Тип теста

	данные	данные	результат	
проверка обращения к репозиторию с целью добавления новой записи	объект класса **Record**	Объект класса User с указанным идентификатором	Произошло обращение к классу **Record_reposit**, методу **Add**	общий

19. Проверка взаимодействия метода **void Log(Record record)**класса **Activity_serv** с классом **Record_reposit** методу **Add.**

Зависимости:

Класс: Activity_repositore

Метод: void Add (Record record)

Описание	Входные данные	Выходные данные	Ожидаемый результат	Тип теста
проверка обращения к репозиторию с целью добавления новой записи	объект класса **Record**	Объект класса Activity с указанным идентификатором	Произошло обращение к классу **Record_reposit**, методу **Add**	общий

20. Проверка взаимодействия метода **void EditRecord(Record record)** класса **Work_time** с классом **Record_reposit** методу **Edit.**

Зависимости:

Класс: Record_reposit

Метод: void Edit (Record record)

Описание	Входные данные	Выходные данные	Ожидаемый результат	Тип теста
проверка обращения к репозиторию с целью добавления новой записи	объект класса **Record**	Объект класса Activity с указанным идентификатором	Произошло обращение к классу **Record_reposit**, методу **Edit**	общий

21. Проверка взаимодействия метода **void EditRecord(Record record)** класса **UserService** с классом **Record_reposit** методу **Edit.**

Зависимости:

Класс: Record_reposit

Метод: void Edit (Record record)

Описание	Входные данные	Выходные данные	Ожидаемый результат	Тип теста
проверка обращения	объект класса User	Объект класса Activity с указанным	Произошло обращение к	общий

к репозиторию с целью добавления новой записи		идентификатором	классу **Record_reposit**, методу **Edit**	

22. Проверка взаимодействия метода **void EditRecord(Record record)** класса **Activity_serv** с классом **Record_reposit** методу **Edit.**

Зависимости:

Класс: Activity_reposit

Метод: void Edit (Record record)

Описание	Входные данные	Выходные данные	Ожидаемый результат	Тип теста
проверка обращения к репозиторию с целью добавления новой записи	объект класса Activity	Объект класса Activity с указанным идентификатором	Произошло обращение к классу **Record_reposit**, методу **Edit**	общий

23. Проверка взаимодействия метода **Record GetRecord(int recordId)** класса **Work_time** с классом **Record_reposit** методу **GetRecords.**

Зависимости:

Класс: Record_reposit

Метод: IQueryable<Record> GetRecords ()

Возвращаемое значение: коллекция объектов класса **Record**

Описание	Входные данные	Выходные данные	Ожидаемый результат	Тип теста
проверка обращение к репозиторию с целью получения записей	Идентификатор записи RecordId (существующий)	Объект класса Record с указанным идентификатором	Произошло обращение к классу **Record_reposit**, методу **GetRecords.**	общий

24. Проверка взаимодействия метода **Record GetRecord(int recordId)** класса **UserService** с классом **Record_reposit** методу **GetRecords.**

Зависимости:

Класс: User_reposit

Метод: IQueryable<Record> GetRecords ()

Возвращаемое значение: коллекция объектов класса **Record**

Описание	Входные данные	Выходные данные	Ожидаемый результат	Тип теста
проверка обращение к репозиторию с целью получения	объект класса User с указанным идентификатором	Объект класса Record с указанным идентификатором	Произошло обращение к классу **Record_reposit**, методу **GetRecords.**	общий

записей				

25. Проверка взаимодействия метода **Record GetRecord(int recordId)** класса **Activity_serv** с классом **Record_reposit** методу **GetRecords.**

Зависимости:

Класс: Activity_repositore

Метод: IQueryable<Record> GetRecords ()

Возвращаемое значение: коллекция объектов класса **Record**

Описание	Входные данные	Выходные данные	Ожидаемый результат	Тип теста
проверка обращение к репозиторию с целью получения записей	объект класса Activity с указанным идентификатором	Объект класса Record с указанным идентификатором	Произошло обращение к классу **Record_reposit**, методу **GetRecords.**	общий

26. Проверка взаимодействия метода **IEnumerable<Record>GetUsersRecords(string directorId)** класса **Work_time** с классом **Record_reposit** методу **GetRecords.**

Зависимости:

Класс: Record_reposit

Метод: IQueryable<Record> GetRecords ()

Возвращаемое значение: коллекция объектов класса **Record**

Описание	Входные данные	Выходные данные	Ожидаемый результат	Тип теста
проверка обращение	идентификатор директора	Объект класса Record с указанным	Произошло обращение к	общий

к репозиторию с целью получения записей	**directorId**	идентификатором	классу **Record_reposit**, методу **GetRecords.**	

Аттестационное тестирование

№	Описание	Входные элементы	Ожидаемый результат
1	создание записи рабочего	введена дата за прошлый месяц	модель не валидна, сообщение об ошибке
2	создание записи рабочего	введена дата за следующий месяц	модель не валидна, сообщение об ошибке
3	создание записи рабочего	введена дата за текущий месяц	модель валидна, запись сохранена
4	создание записи рабочего	Введена дата последнего дня текущего месяца	модель валидна, запись сохранена
5	создание записи рабочего	введена дата первого дня текущего месяца	модель валидна, запись сохранена
6	создание записи рабочего	введена дата: 20 мая 2016	модель не валидна, уведомление об ошибке
7	создание записи рабочего	значение даты пусто	модель не валидна, сообщение об ошибке
8	создание записи рабочего	введено отрицательное значение рабочих часов	Модель не валидна, сообщение об ошибке
9	создание записи рабочего	введено значение рабочих часов больше 12	модель не валидна, уведомление об ошибке
10	создание записи рабочего	введено значение рабочих часов равное 0	модель валидна, запись сохранена

11	создание записи рабочего	введено значение рабочих часов равное 12	модель валидна, запись сохранена
12	создание записи рабочего	введено значение рабочих часов равное 8	модель валидна, запись сохранена
13	создание записи рабочего	значение рабочих часов пусто	не валидна, уведомление об ошибке
14	открытие страницы рабочего, используя аккаунт директора		Отображение страницы рабочего
15	открытие страницы директора, используя аккаунта рабочего		Отображение страницы директора
16	Выход из аккаунта		Сообщение об удачном выходе из аккаунта
17	Авторизация пользователя	Пустое значение	Сообщение об ошибке авторизации
18	Авторизация пользователя	Неверное значение логина и пароля	Сообщение об ошибке авторизации
19	Авторизация пользователя	Верное значение логина и пароля	Сообщение об успешной авторизации
20	Удаление записи рабочего		Запись удалена
21	Редактирование записи рабочего		Запись отредактирована
22	Просмотр отчета рабочего		Генерирование отчета

23	Переход на любую другую страницу приложения		Страницы отображаются корректно, сообщение об ошибках не появляется

Покрытие исполняемого кода тестами

Количество строк кода тестируемых

модулей: 1345;

Количество строк кода, покрытых тестами:

957;

Покрытие = (957 / 1345) * 100% = 71,1 %

Примеры реализации тестов

Для запуска тестов использовалась платформа модульных тестов Microsoft.

1. Модульное тестирование

```
/// Work_time should require records of current user from Record_reposit.
  [TestMethod]
  public void Work_timeShouldRequireRecordsOfCurrentUserfromRecord_reposit()
  {
      // Arrange
      Work_time = new Work_time(this.reposit); List<Record> records = new List<Record>()
      {
          new Record()
          {
              RecordId = 0, DirectorId =
              "user"
        }
      };
      var expectedRecord = new Record()
      {
          RecordId = 0, DirectorId =
          "user"
```

```
    };

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

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

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

2. Интеграционное

```
/// EditRecord() action should send recordId to Record_reposit.
 [TestMethod]
public void EditRecordShouldSendRecordIdToRepository()
{
    // Arrange
    Work_time = new Work_time(this.reposit); Record record = new Record
    {
        RecordId = 0,
        Date = new DateTime(2016, 05, 20),
        Work_eime = 5,

        ProjectId = 6, };
    ActivityId = 7

    // Act service.EditRecord(record);

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

Результаты модульного тестирования

№ теста	Класс	Метод	Кол-во тестов	Кол-во ошибок	Дата проведения
1	**AllowAccessAttribute**	IsInRole	1	0	31.05.2016
2	**AllowAccessAttribute**	IsInRole	1	0	31.05.2016
3	**AllowAccessAttribute**	IsInRole	1	0	31.05.2016
4	**Work_time**	Get	1	0	31.05.2016
4.1	**Work_time**	Get	1	0	31.05.2016
5	**User_service**	Get	1	0	31.05.2016
5.1	**User_service**	Get	1	0	31.05.2016
6	**Work_time**	GetUsersRecords	1	1	31.05.2016
6.1	**Work_time**	GetUsersRecords	1	1	31.05.2016
7	**Active_serv**	Get	1	0	31.05.2016
7.1	**Active_serv**	Get	1	0	31.05.2016
7.2	**Active_serv**	GetUsersRecords	1	0	31.05.2016
8	**User_service**	GetUsersRecords	1	0	31.05.2016
9	**User_service**	GetUsersRecords	1	0	31.05.2016

Результаты интеграционного тестирования

1	**DefaultControl**	Index	1	0	31.05.2016
2	**DefaultControl**	Index	1	0	31.05.2016
3	**RecordControl**	AddRecord	1	0	31.05.2016
4	**RecordControl**	AddRecord	1	0	31.05.2016
5	**RecordControl**	Delete	1	0	31.05.2016
6	**RecordControl**	Delete	1	0	31.05.2016
7	**RecordControl**	Delete	1	0	31.05.2016
8	**RecordControl**	Edit	1	0	31.05.2016
9	**RecordControl**	Edit	1	0	31.05.2016
10	**RecordControl**	Edit	1	0	31.05.2016
11	**RecordControl**	Edit	1	0	31.05.2016
12	**RecordControl**	EditRecord	1	0	31.05.2016
13	**RecordControl**	EditRecord	1	0	31.05.2016
14	**RecordControl**	EditRecord	1	0	31.05.2016
15	**RecordControl**	Index	1	0	31.05.2016
16	**Work_time**	Delete	1	0	31.05.2016
18	**Work_time**	Log	1	0	31.05.2016
19	**Work_time**	EditRecord	1	0	31.05.2016
20	**Work_time**	GetRecord	1	0	31.05.2016
21	**Work_time**	GetUsersRecords	1	0	31.05.2016
22	**UserService**	Log	1	0	31.05.2016

23	**UserService**	Delete	1	0	31.05.2016
24	**UserService**	EditRecord	1	0	31.05.2016
25	**Activity_serv**	Delete	1	0	31.05.2016
26	**Activity_serv**	EditRecord	1	0	31.05.2016
27	**Activity_serv**	Log	1	0	31.05.2016

Аттестационное тестирование

№	Результат	Дата проведения
1	Провален(2)	31.05.2016
2	Провален(2)	31.05.2016
3	Успех	31.05.2016
4	Успех	31.05.2016
5	Успех	31.05.2016
6	Успех	31.05.2016
7	Успех	31.05.2016
8	Успех	31.05.2016
9	Успех	31.05.2016
10	Успех	31.05.2016
11	Успех	31.05.2016
12	Успех	31.05.2016
13	Успех	31.05.2016
14	Успех	31.05.2016
15	Успех	31.05.2016
16	Успех	31.05.2016
17	Успех	31.05.2016
18	Успех	31.05.2016
19	Успех	31.05.2016
20	Успех	31.05.2016
21	Успех	31.05.2016
22	Успех	31.05.2016
23	Успех	31.05.2016

Найденные ошибки.

1. Класс **Work_time** метод **GetUsersRecords:** в качестве возвращаемого 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.