1. [**Разработка технического задания**](#_30j0zll) **2**
   1. [Исходная постановка задачи проектирования](#_1fob9te) 2
   2. [Анализ условий предприятия](#_3znysh7) 2
      1. [Цели функционирования предприятия](#_2et92p0) 3
      2. [Организационная структура предприятия](#_tyjcwt) 4
      3. [Состав бизнес-процессов](#_y0pdkaa1a84s) 5
      4. [Содержание бизнес-процессов](#_1t3h5sf) 6
      5. [Классы-объекты предметной области](#_4d34og8) 8
      6. [Автоматизируемые элементы](#_2s8eyo1) 9
   3. [Формирование требований к системе](#_17dp8vu) 10
      1. [Состав требований](#_3rdcrjn) 10
      2. [Состав подсистем](#_26in1rg) 13
      3. [Варианты использования](#_lnxbz9) 15
      4. [Содержание сценариев](#_35nkun2) 17
2. [**Разработка рабочего проекта**](#_1ksv4uv) **27**
   1. [Определение классов анализа](#_44sinio) 27
   2. [Определение методов объекта](#_2jxsxqh) 31
   3. [Выбор технологии реализации](#_z337ya) 44
   4. [Проектирование хранилища данных](#_3j2qqm3) 45
3. [**Разработка программного кода**](#_ls90hda2m6rg) **53**
4. [**Развертывание**](#_2xcytpi) **66**
   1. [Интерфейс системы](#_1ci93xb) 66
5. [**Разработка сценариев тестирования**](#_1y810tw) **72**
6. [**Список литературы**](#_3whwml4) **78**

# 

# **Разработка технического задания**

## **Исходная постановка задачи проектирования**

Задачей проектирования является разработка системы мониторинга и исправления неисправностей оборудования Кемеровского областного медицинского колледжа (далее КОМК).

## **Анализ условий предприятия**

Полное официальное наименование: Кемеровский областной медицинский колледж.

Сокращенное наименование: КОМК.

Статус организации, форма юридического лица: государственное бюджетное профессиональное учреждение «Кемеровский областной медицинский колледж».

Общие принципы функционирования организации:

Деятельность КОМК проводится в соответствии со следующими общими принципами функционирования организаций:

* Скалярный принцип (scalaris - ступенчатый) - иерархичность построения организаций.
* Принцип функциональности - должностные обязанности на каждой ступени управления определяются настолько детально, насколько возможно.
* Принцип диапазона контроля – руководитель умственной деятельности имеет не более 8 человек.
* Принцип личной ответственности - ответственность руководителя за действия подчинённых ему людей в полном объёме.

Вид и профиль деятельности организации:

Основная деятельность организации включает:

1. Обучение абитуриентов по специальностям:
   * «Лечебное дело», профиль фельдшер;
   * «Акушерское дело», профиль акушерка;
   * «Сестринское дело», профиль мед. сестра/брат;
   * «Лабораторная диагностика», профиль медицинский лабораторный техник;
   * «Фармация», профиль фармацевт;
   * «Стоматология», профиль зубной техник.
2. Предоставление услуг по повышению квалификации младшего мед. персонала и людей смежных профессий (МЧС, Пожарные службы).

Физический адрес: 650064, Кемеровская область, г. Кемерово, пр. Советский, 58.

### **Цели функционирования предприятия**

Цели функционирования КОМК приведены на рисунке 1.

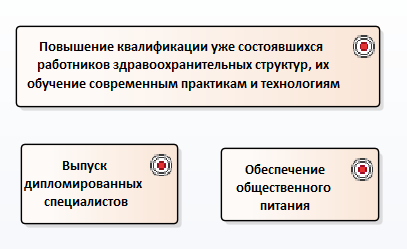


Рисунок 1. Цели функционирования КОМК

Описание целей функционирования колледжа:

* Повышение квалификации уже состоявшихся работников здравоохранительных структур, их обучение современным практикам и технологиям – обучение людей пришедших на повышение квалификации;
* Выпуск дипломированных специалистов – обучение студентов.

Для поддержки основных целей функционирования, выделим отдельную цель:

* Обеспечение бесперебойной работы колледжа - поддержание колледжа в работоспособном режиме.

### **Организационная структура предприятия**

Модель организационной структуры приведена на рисунке 2.

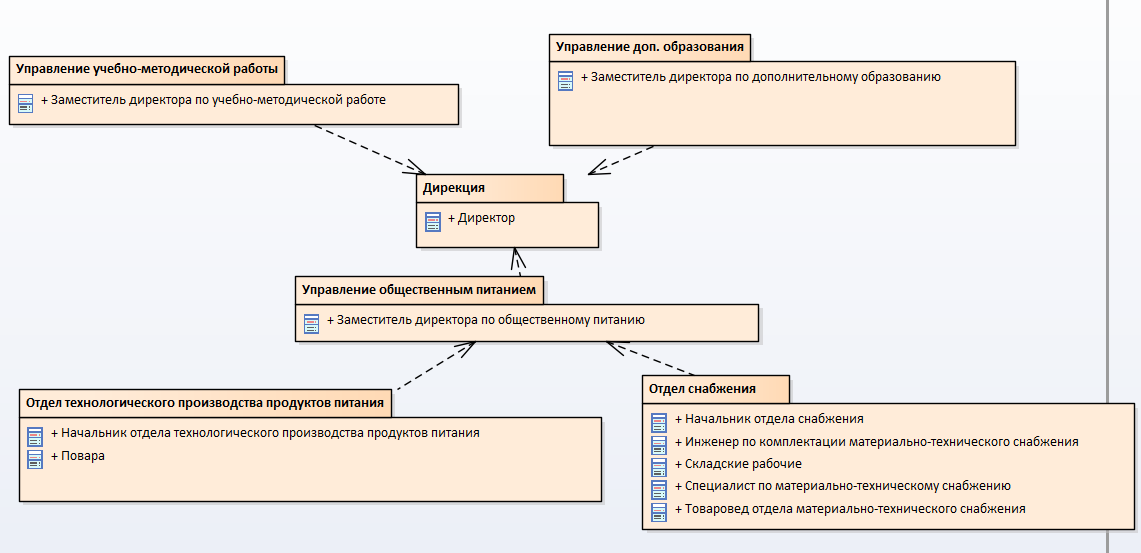


Рисунок 2. Организационная структура предприятия

Руководство деятельностью КОМК осуществляет: Дирекция. Блоку дирекция подчиняются:

* Управление учебно-методической работой – осуществление контроля учебного процесса.
* Управление доп. образованием – осуществляет контроль за дополнительным образованием;
* Управление общественным питанием - включает в себя: заместитель директора по общественному питанию; осуществляет координацию отделов:
  + Отдел технологического производства продуктов питания - состоит из: начальник отдела технологического производства продуктов питания; повара; осуществляет организацию питания колледжа и проверку его качества.
  + Отдел снабжения - состоит из: начальник отдела снабжения, складские рабочие, специалист по материально-техническому снабжению, инженер по комплектации материально-технического снабжения, товаровед отдела материально-технического снабжения; осуществляет контроль за оборотом товаров.

### **Состав бизнес-процессов**

Состав бизнес-процессов приведен на рисунке 3.

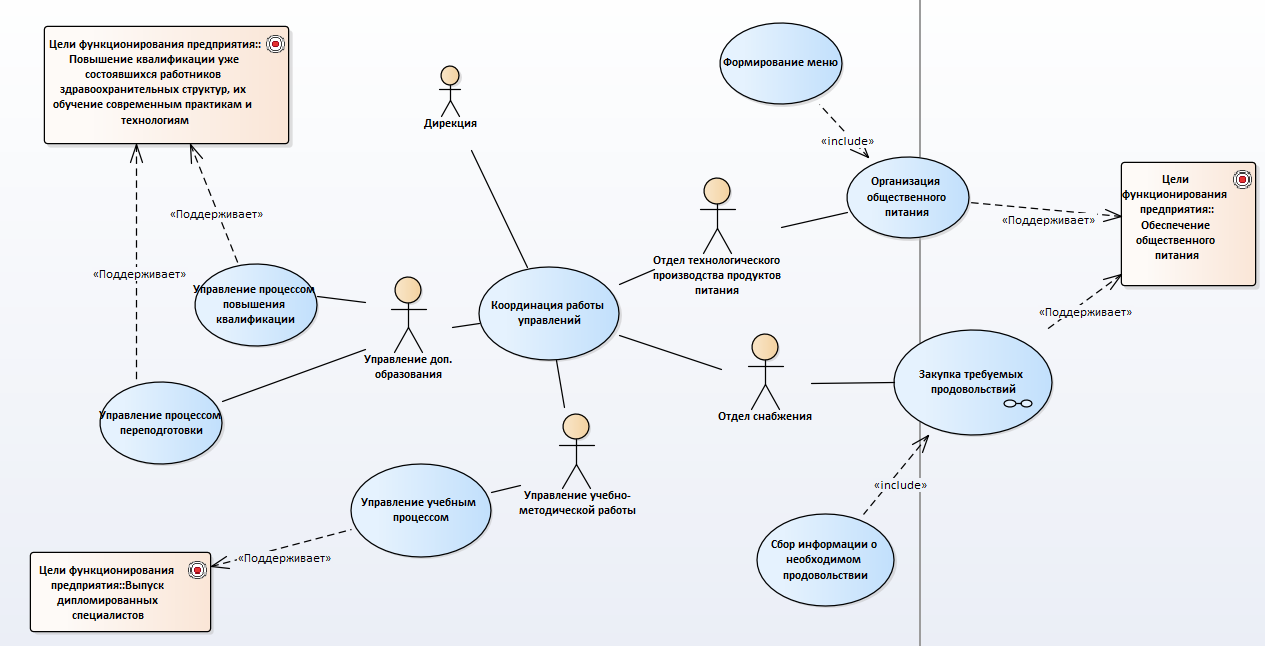


Рисунок 3. Состав бизнес-процессов

* Дирекция - осуществляет координацию работы управлений;
* Отдел технологического производства продуктов питания - осуществляет организацию общественного питания;
* Формирование меню;
* Отдел снабжения – осуществляет закупку требуемых продовольствий;
* Сбор информации о необходимом продовольствии;
* Отдел управление доп образованием осуществляет управление процессом переподготовки и процессом повышения квалификации;
* Отдел управление учебно-методической работой осуществляет управление учебным процессом.

### **Содержание бизнес-процессов**

Для подробного анализа выбран следующий процесс: “Формирование меню”.

Содержание бизнес-процесса “ Формирование меню” приведено на рисунке 4.

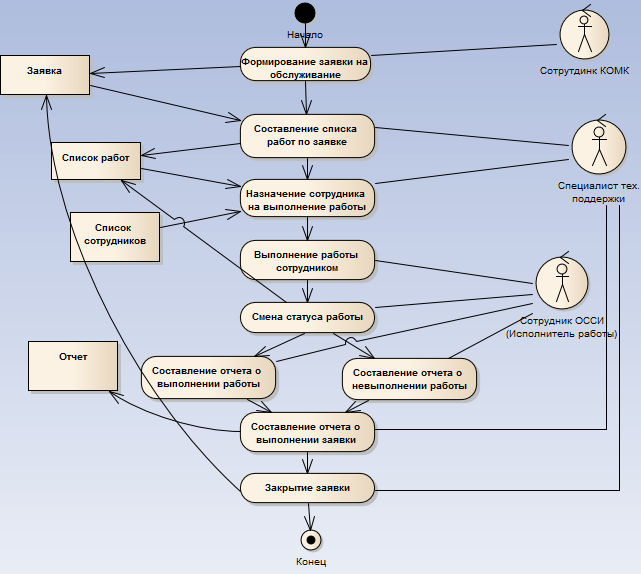


Рисунок 4. Сбор информации об тех. неисправностях и их исправление

Работы выполняемые сотрудником КОМК:

* Формирование заявки на обслуживание – сотрудник пишет заявку по поводу неисправности оборудования специалисту тех. поддержки;

Работы выполняемые специалистом тех поддержки:

* Составление списка работ по заявке;
* Назначение сотрудника на выполнение работы;
* Составление отчета о выполнении заявки;
* Закрытие заявки;

Работы выполняемые сотрудником ОССИ:

* Выполнение работы сотрудником;
* Смена статуса работы;
* Составление отчета по выполнению/невыполнению работы;
* Отчет о проверке направляется в соответствующие органы/отделы.

Описание объектов:

* Заявка – является выходным объектом формирования заявки пользователем; входным объектом для составления списка работ;
* Список работ – является выходным объектом для составления списка работ; входным объектом для назначения сотрудника на работу;
* Список сотрудников– является входным объектом для назначения сотрудника на работу;
* Отчет – выходным для закрытия заявки.

### **Классы-объекты предметной области**

Описание классов-объектов предметной области приведено на рисунке 5.

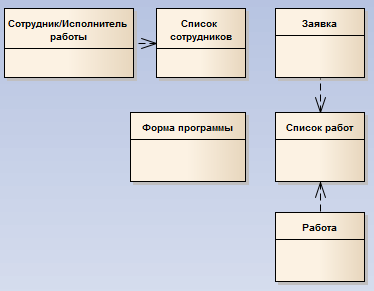


Рисунок 5. Описание классов-объектов предметной области

Выявленный классы:

* Работа – информационный объект, содержащий описание необходимых действий;
* Список работ – информационный объект, представляет собой список всех необходимых для заявки работ;
* Заявка – информационный объект, представляет собой запрос на обслуживание;
* Сотрудник/Исполнитель работы – информационный объект, представляет собой данные о сотруднике;
* Список сотрудников - информационный объект, представляет собой список всех исполнителе;
* Форма программы – информационный объект, представляет собой интерфейс программы.

### **Автоматизируемые элементы**

Предполагаемые автоматизируемые элементы приведены на рисунке 6.

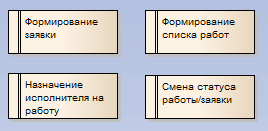


Рисунок 6. Автоматизируемые элементы бизнес-процессов

Формирование заявки соответствует функции «Формирование заявки». Автоматизация данного процесса позволит сократить время составления заявки и уменьшить количество ошибок.

Формирование списка работ соответствует функции «Составление работ». Автоматизация данного процесса позволит быстро составить план выполнения заявки .

Назначение исполнителей на работу «Управление исполнением необходимых работ». Автоматизация данного процесса позволит обеспечить более быстрый и точный подбор кадров для выполнения задачи.

Смена статуса работы/заявки соответствует функции «Изменение статуса работ». Автоматизация данного процесса позволит быстро получить информацию по статусу работы.

## **Формирование требований к системе**

### **Состав требований**

Соответствие функциональных требований автоматизируемым бизнес-решениям отобразим в виде матрицы трассировки (таблица 1).

Таблица 1. Требования

|  |  |  |
| --- | --- | --- |
| **№** | **Бизнес-процесс** | **Требование** |
| 1 | Формирование заявки | Обеспечение возможности создания заявки в фиксированной форме. |
| 2 | Формирование списка работ | Обеспечение возможности формирования списка работ на основе поступившей заявки;  Обеспечение формирования типов работ. |
| 3 | Назначение исполнителя на работу | Обеспечение возможности назначения на работу сотрудника;  Обеспечение возможности разграничения прав пользователей и формирования их списка умений. |
| 4 | Смена статуса работы/заявки | Обеспечение возможности смены статуса заявки с сохранением истории ее исполнения. |
| Обеспечение возможности смены статуса работы с сохранением истории ее исполнения. |

Диаграмма соответствия функциональных требований автоматизируемым бизнес-решениям приведена на рисунке 7.

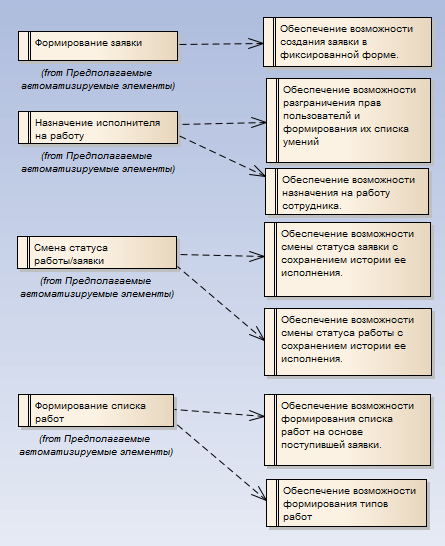


Рисунок 7. Диаграмма соответствия функциональных требований автоматизируемым бизнес-решениям

### **Состав подсистем**

Для отображения выявленных подсистем построим матрицу трассировки «подсистема – функциональные требования» (таблица 2).

Таблица 2. Подсистемы

|  |  |
| --- | --- |
| **Подсистема** | **Требование** |
| Подсистема работы с заявками | Обеспечение возможности формирования списка работ на основе поступившей заявки. |
| Обеспечение возможности назначения на работу сотрудника. |
| Обеспечение возможности смены статуса заявки с сохранением истории ее исполнения. |
| Подсистема создания заявок | Обеспечение возможности создания  заявки в фиксированной форме. |
| Подсистема работы с работами по заявке | Обеспечение возможности смены статуса работы с сохранением истории ее исполнения. |
| Обеспечение возможности формирования типов работ. |
| Обеспечение возможности разграничения прав пользователей и формирования их списка умений |

Диаграмма выявленных подсистем приведена на рисунке 8.

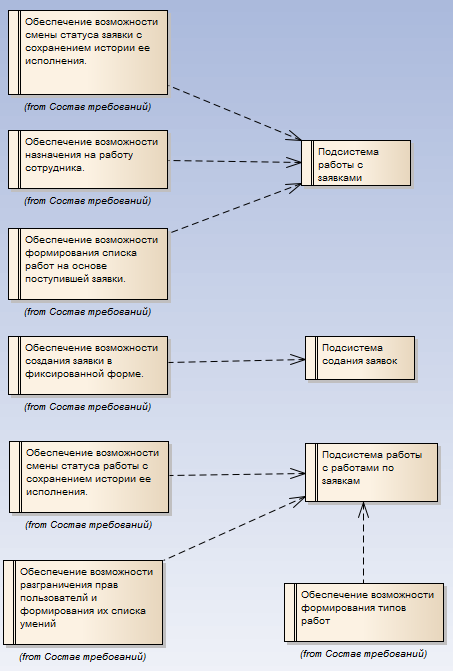


Рисунок 8. Диаграмма выявленных подсистем

### **Варианты использования**

Диаграмма вариантов использования, отражающая функциональные требования к системе и сценарии, их реализующие, приведены на рисунке 9.

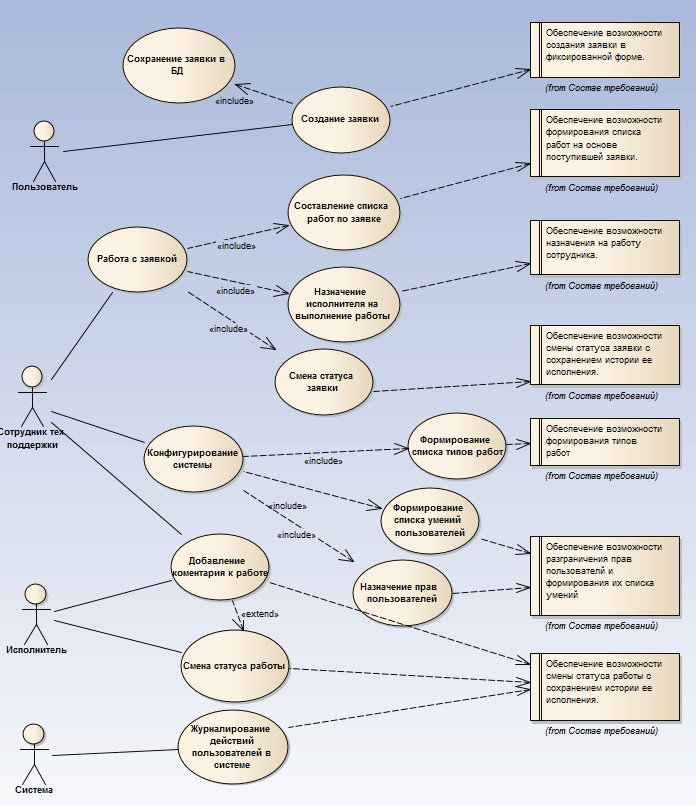


Рисунок 9. Диаграмма вариантов использования

Функции выполняемые пользователем:

* Создание заявки, включает в себя сохранение заявки в БД.

Функции выполняемые исполнителем:

* Смена статуса работы, которая расширяется возможность добавления комментария.

Функции выполняемые сотрудником тех. поддержки:

* Работа с заявкой, которая включает в себя: составление списка работ по заявке, назначение исполнителя на выполнение работы и смена статуса заявки;
* Конфигурирование системы, которая включает в себя: формирование списка типов работ, формирование списка умений пользователей, назначение прав пользователей;
* Добавление комментария к работе.

Функции выполняемые системой:

* Журналирование действий пользователей в системе.

### **Содержание сценариев**

Проведем анализ следующих сценариев: «Создание заявки», «Работа со справочником», «Составление отчета», «Авторизация пользователя», «Журналирование действий пользователя в системе».

Сценарии выполнения остальных ВИ в данном отчете рассмотрены не будут, т.к. они имеют косвенное отношение к работе с заявками пользователей.

Диаграмма деятельности сценария «Создание заявки» приведена на рисунке 10.

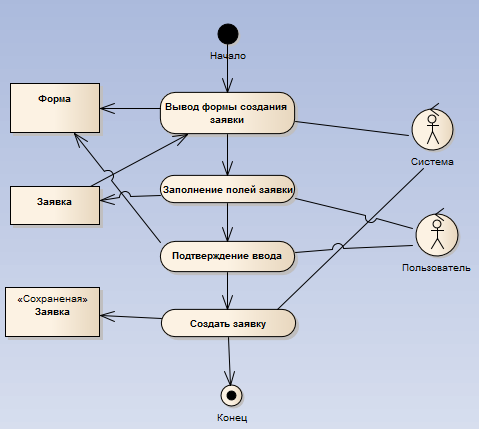


Рисунок 10. Создание заявки

В результате работы процесса выявлены следующие объекты:

* Граничные объекты:
* Форма – содержит формы «Форма создания заявки». Позволяет создавать, изменять, удалять, искать и выгружать документ в Word.
* Сущности:
* Заявка – содержит шаблон для создания заявки.
* Сохраненная заявка – содержит описание возникшей проблемы с указанием даты подачи заявки и заявителя

Сценарии выполняются пользователем и системой:

* Вывод формы заявки, открывается форма создания заявки с полем текста заявки и значениями текущей даты и описания заявителя (выполняется системой);
* Заполнение полей заявки (выполняется пользователем);
* Подтверждение ввода (выполняется пользователем);
* Сохранение заявки в БД (выполняется системой).

Диаграмма деятельности сценария «Работа с заявкой» приведена на рисунке 11.

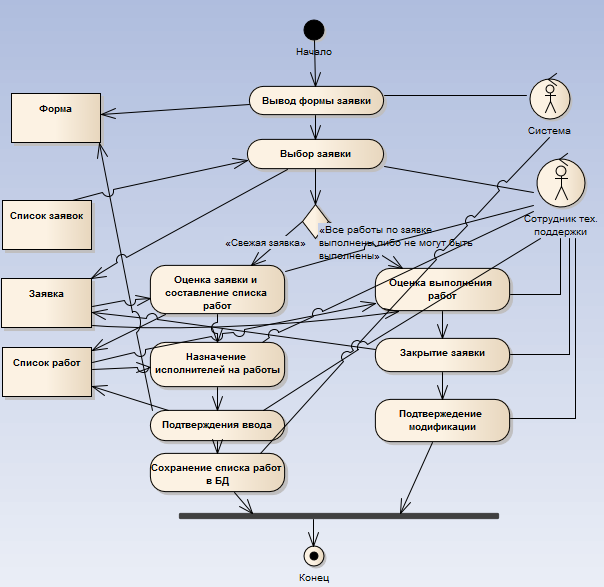


Рисунок 11. Работа с заявкой

В результате работы процесса выявлены следующие объекты:

* Граничные объекты:
* Форма – содержит формы «Добавить работу», «Работа с заявкой». Позволяет назначать исполнителям работы по заявке.
* Сущности:
* Список заявок – содержит заявки пользователей;
* Заявка – содержит данные о конкретной заявке;
* Список работ - содержит данные о работах по заявке.

В выполнении сценариев участвуют система и сотрудник тех. поддержки:

* Выводится форма работы с заявками;
* Сотрудник выбирает заявку:
  + Закрытие заявки;
    - Выполняется оценка выполнения работ назначенных на заявку;
    - В случае если все работы выполнены, либо не могут быть выполнены заявка закрывается;
    - Подтверждается закрытие заявки, данные о закрытии записываются в БД предприятия.
  + Обработка заявки;
    - Сотрудник просматривает заявку и составляет к ней список работ;
    - На назначенные работы выбирается исполнитель;
    - Если обработка заявки окончена, сотрудник подтверждает окончание работы с заявкой;
    - Данные о заявке и назначенных на нее работах сохраняются в БД.

Диаграмма деятельности сценария «Смена статуса работы» приведена на рисунке 12.

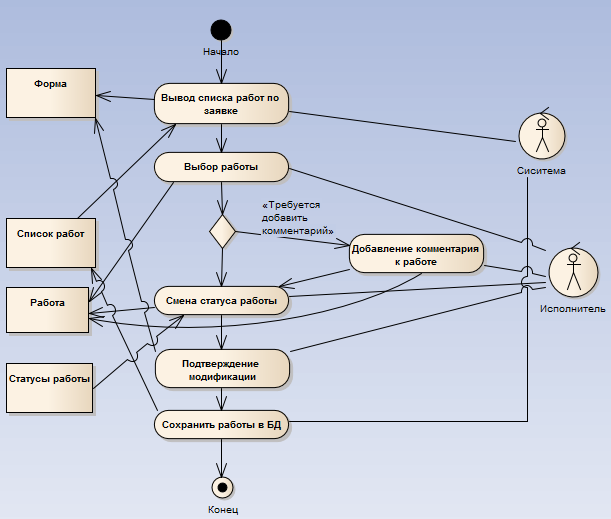


Рисунок 12. Смена статуса работы

В результате работы процесса выявлены следующие объекты:

* Граничные объекты:
* Форма – содержит формы «Смена статуса работы». Позволяет исполнителю работы изменять статус работы, а также комментировать её.
* Сущности:
* Заявка - содержит описание заявки, ее статус, а также список работ, требуемых для ее закрытия.
* Работа - содержит описание работы.
* Список работ - содержит все работы, требуемые к выполнению.
* Статусы работы- содержит все статусы возможное для работы.

В выполнении сценария участвуют система и исполнитель:

* Вывод списка работ по заявке (выполняется системой);
* Выбор работы (выполняется исполнителем);
  + Если исполнитель считает нужным, выполняется добавление комментария к работе (выполняется исполнителем);
* Подтверждение модификации (выполняется исполнителем):
* Обновление БД, изменения статуса работы по заявке фиксируется в БД (выполняется системой).

Диаграмма деятельности сценария «Авторизация пользователя» приведена на рисунке 13.

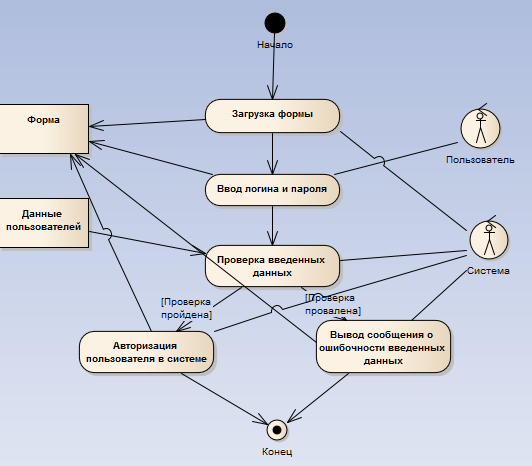


Рисунок 13. Авторизация пользователя

В результате работы процесса выявлены следующие объекты:

* Граничные объекты:
* Форма – содержит форму «Авторизация пользователя». Позволяет пользователю авторизоваться за свой аккаунт.
* Сущности:
* Данные пользователя - содержит данные о пользователях зарегистрированных в системе.

В выполнении сценария участвуют система и исполнитель:

* Вывод формы авторизации (выполняется системой);
* Ввод данных (выполняется исполнителем);
* Проверка данных (выполняется системой):
  + Если проверка пройдена: Авторизация пользователя в системе (выполняется системой);
  + Если проверка провалена: Вывод сообщения об ошибочности введенных данных (выполняется системой).

Диаграмма деятельности сценария «Журналирование действий пользователя в системе» приведена на рисунке 14.

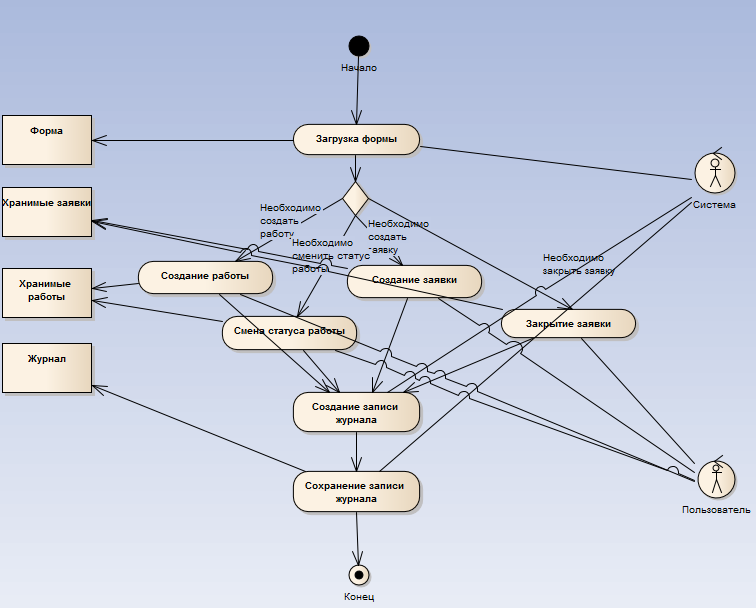


Рисунок 14. Журналирование действий пользователя в системе

В результате работы процесса выявлены следующие объекты:

* Граничные объекты:
* Форма – содержит формы «Смена статуса работы». Позволяет исполнителю работы изменять статус работы, а также комментировать её.
* Сущности:
* Хранимые заявки - содержит хранимые заявки;
* Хранимые работы - содержит хранимые работы;
* Журнал - содержит записи о действиях пользователей.

В выполнении сценария участвуют система и пользователь:

* Загрузка формы(выполняется системой);
  + Создание работы (выполняется пользователем);
  + Смена статуса работы (выполняется пользователем);
  + Создание заявки (выполняется пользователем);
  + Закрытие заявки (выполняется пользователем);
* Создание записи журнала (выполняется системой);
* Сохранение записи в журнал (выполняется системой).

Проект пользовательского интерфейса в виде диаграммы экранных форм приведены на рисунке 15.

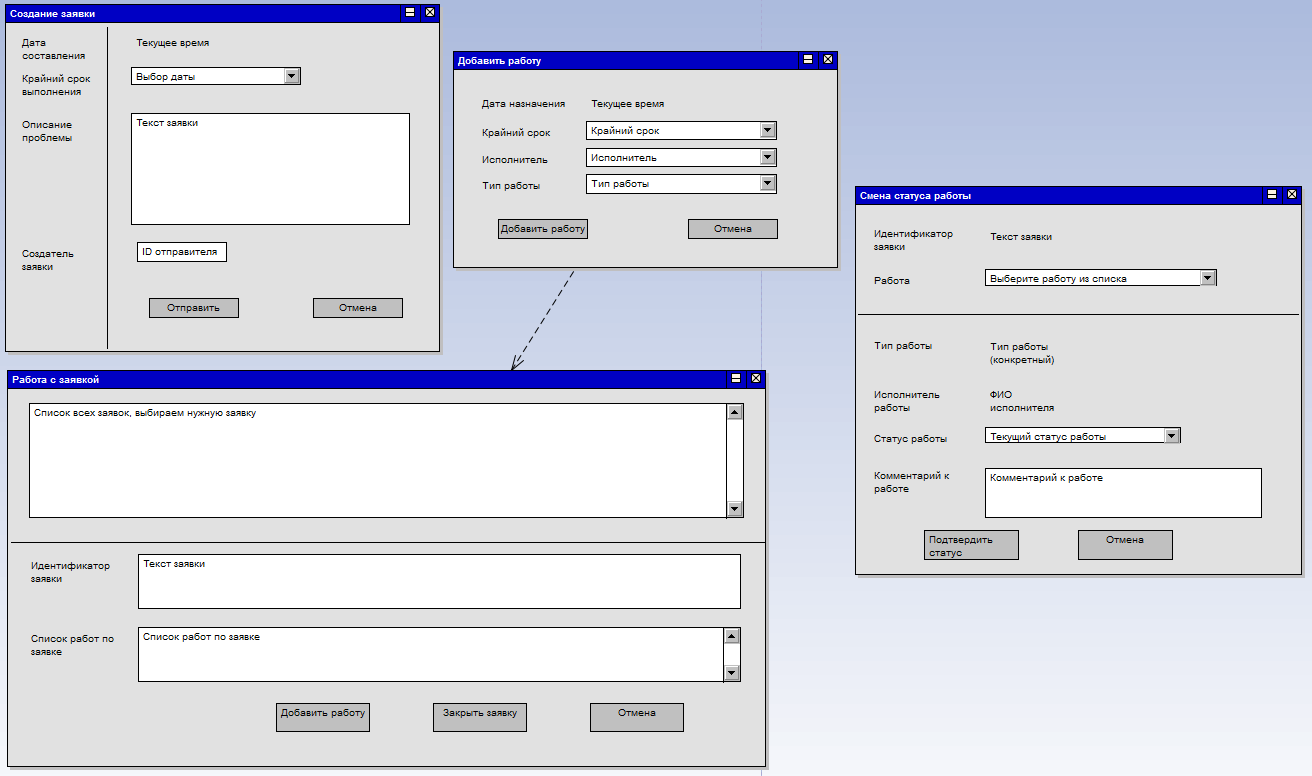


Рисунок 15. Интерфейс подсистемы «Внутренний документооборот ДОН Кемеровской области»

Требования к интерфейсу:

* Возможность составления заявок;
* Возможность обработки заявок сотрудником тех. поддержки;
* Возможность добавления работ к заявке;
* Возможность смены статуса работы;
* Возможность добавления комментария к работе;
* Возможность возможность смены статуса заявки;
* Возможность назначения исполнителей на работы.

Требования к СУБД: PostgreSQL Server. Должны быть реализованы сущности:

* «Исполнители работ» - обеспечивает хранение 40 записей.
* «Тип работы» - обеспечивает хранение 50 записей.
* «Заявители» - обеспечивает хранение 250 записей.
* «Статус работы» - обеспечивает хранение 7 записей.
* «Статус заявки» - обеспечивает хранение 7 записей.

Сущности «Заявки», «Работы» являются динамическими, должна быть осуществлена возможность хранения минимум 400 записей, каждой сущности.

# **Разработка рабочего проекта**

## **Определение классов анализа**

Диаграмма классов-анализа приведена на рисунке 16:

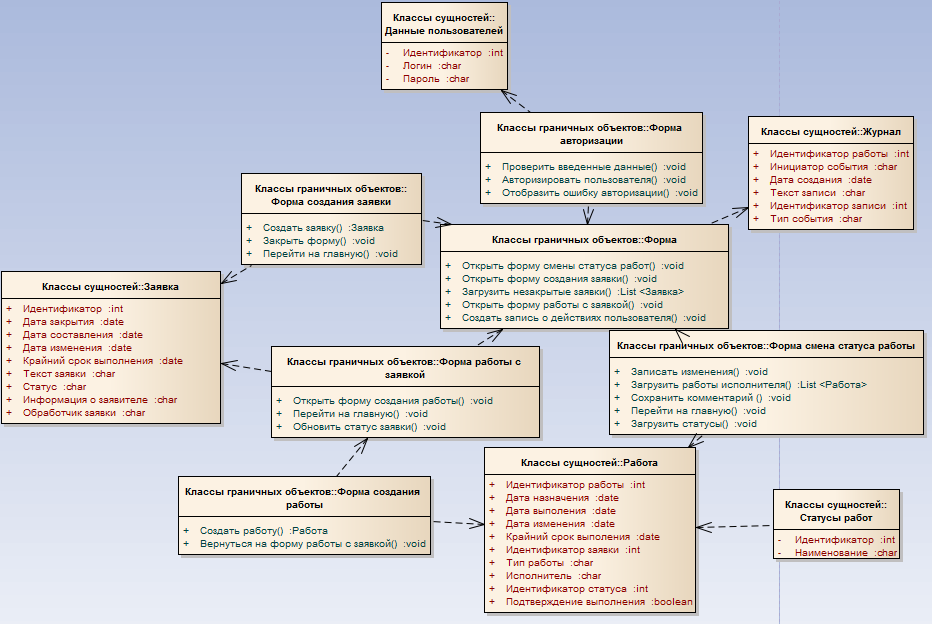


Рисунок 16. Диаграмма классов-анализа.

Выявлены классы:

Классы граничных объектов:

* Класс "Форма" – определён на основе объекта «Форма» сценариев «Создание заявки», «Смена статуса работы», «Работа с заявкой».
  + Открыть форму смены статуса работ, тип возврата void - создает граничный объект класса «Форма смены статуса работ»;
  + Открыть форму создания заявки, тип возврата void - создает граничный объект класса «Форма создания заявки»;
  + Открыть форму работы с заявкой, тип возврата void - создает граничный объект класса «Форма работы с заявкой»;
  + Загрузить незакрытые заявки, тип возврата List <Заявка> - возвращает список оставленных пользователями заявок.
* Класс "Форма создания заявки" – определён на основе объекта «Форма» сценария «Создание заявки», является подокном «Формы». Методы:
  + Создать заявку, тип возврата Заявка – создает заявку на основе введенных данных;
  + Закрыть форму, тип возврата void – завершает работу на форме;
  + Перейти на главную, тип возврата void - осуществляет переход на главную страницу приложения.
* Класс "Форма работы с заявкой" – определён на основе объекта «Форма» сценария «Работа с заявкой», является подокном «Формы». Методы:
  + Открыть форму создания работы, тип возврата void – создает граничный объект класса «Форма создания работы»;
  + Перейти на главную, тип возврата void - осуществляет переход на главную страницу приложения.
  + Обновить статус заявки, тип возврата void - записывает изменения произошедшие с заявкой в БД.
* Класс "Форма создания работы" – определён на основе объекта «Форма» сценария «Работа с заявкой», является подокном «Формы работы с заявкой». Методы:
  + Создать работу, тип возврата Работа – создает работу на основе введенных данных.
  + Вернуться на форму работы с заявкой, тип возврата void - осуществляет переход на страницу работы с заявкой.
* Класс "Форма смены статуса работы" – определён на основе объекта «Форма» сценария «Смена статуса работы», является подокном «Формы». Методы:
  + Записать изменения, тип возврата void – сохраняет изменения сделанные в работе;
  + Загрузить работы исполнителя, тип возврата List <Работа>– загружает из БД работы назначенные исполнителю;
  + Сохранить комментарий, тип возврата void - сохраняет оставленный исполнителем комментарий к работе;
  + Перейти на главную, тип возврата void - осуществляет переход на главную страницу приложения.
* Класс "Форма авторизации" - определен на основе объекта «Форма» сценария «Авторизация пользователя», является подокном «Формы». Методы:
  + Проверить введенные данные, тип возврата void - запрашивает проверку логина и пароля пользователя;
  + Авторизировать пользователя, тип возврата void - авторизирует пользователя в системе;
  + Отобразить ошибку авторизации, тип возврата void - отображает ошибку авторизации.

Классы сущностей:

* Класс "Заявка" – определён на основе объекта «Заявка» сценариев «Работа с заявкой», «Создание заявки». Атрибуты:
  + Идентификатор, тип int - искусственный атрибут, хранит в себе идентификатор объекта;
  + Дата закрытия, тип date - содержит в себе дату закрытия заявки;
  + Дата составления, тип date - содержит в себе дату создания заявки;
  + Дата изменения, тип date - содержит в себе дату последнего изменения заявки;
  + Крайний срок выполнения, тип date - содержит дату до которой заявка должна быть выполнена;
  + Текст заявки, тип char - содержит в себе описание проблемы, возникшей у заявителя;
  + Статус, тип char - содержит в себе текущий статус заявки;
  + Информация о заявителе, тип char - содержит в себе описание заявителя (ФИО);
  + Обработчик заявки, тип char - содержит в себе ФИО специалиста обработавшего заявку.
* Класс "Работа" – определён на основе объекта «Работа» сценария «Смена статуса работы», объекта «Список работ», сценария «Работа с заявкой». Атрибуты:
  + Идентификатор работы, тип int - искусственный атрибут, хранит в себе идентификатор объекта;
  + Дата назначения, тип date - содержит в себе дату создания работы;
  + Дата выполнения, тип date - содержит в себе дату выполнения работы;
  + Дата изменения, тип date - содержит в себе дату последнего изменения работы;
  + Крайний срок выполнения, тип date - содержит дату до которой работа должна быть выполнена;
  + Идентификатор заявки, тип int - искусственный атрибут, хранит в себе идентификатор заявки к которой работа относится;
  + Тип работы, тип char - содержит в себе тип работы;
  + Исполнитель, тип char - содержит в себе описание исполнителя(ФИО);
  + Статус работы, тип char - содержит в себе текущий статус работы;
  + Подтверждение выполнения, тип boolean - показывает, подтверждено ли выполнение работы.
* Данные пользователей - определен на основе объекта «Данные пользователей» сценария «Авторизация пользователя». Атрибуты:
  + Идентификатор, тип int - искусственный атрибут, хранит в себе идентификатор пользователя;
  + Логин, тип char - содержит в себе логин пользователя;
  + Пароль, тип char - содержит пароль пользователя.
* Статусы работ - определен на основе объекта «Статусы работ» сценария «Смена статуса работы». Атрибуты:
  + Идентификатор, тип int - искусственный атрибут, хранит в себе идентификатор статуса;
  + Наименование, тип char - содержит в себе наименование статуса.
* Журнал - определен на основе объекта «Журнал» сценария «Журналирование действий пользователя в системе». Атрибуты:
  + Идентификатор записи, тип int - искусственный атрибут, хранит в себе идентификатор записи;
  + Идентификатор работы, тип int - содержит идентификатор работы в которой сделаны изменения;
  + Инициатор события, тип char - содержит ФИО пользователя, спровоцировавшего создание записи;
  + Дата создания, тип date - содержит дату создания записи;
  + Тип события, тип char - содержит тип события.

## **Определение методов объекта**

Диаграммы последовательности сценариев: «Создание заявки», «Работа с заявкой», «Смена статуса работы».

Диаграмма последовательности сценария «Создание заявки» приведена на рисунке 17.

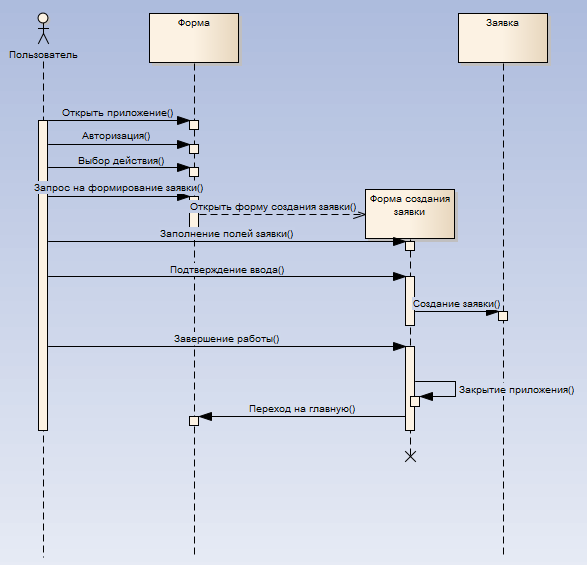


Рисунок 17. Создание заявки

В ходе построения объекты диаграммы выявлены следующие объекты:

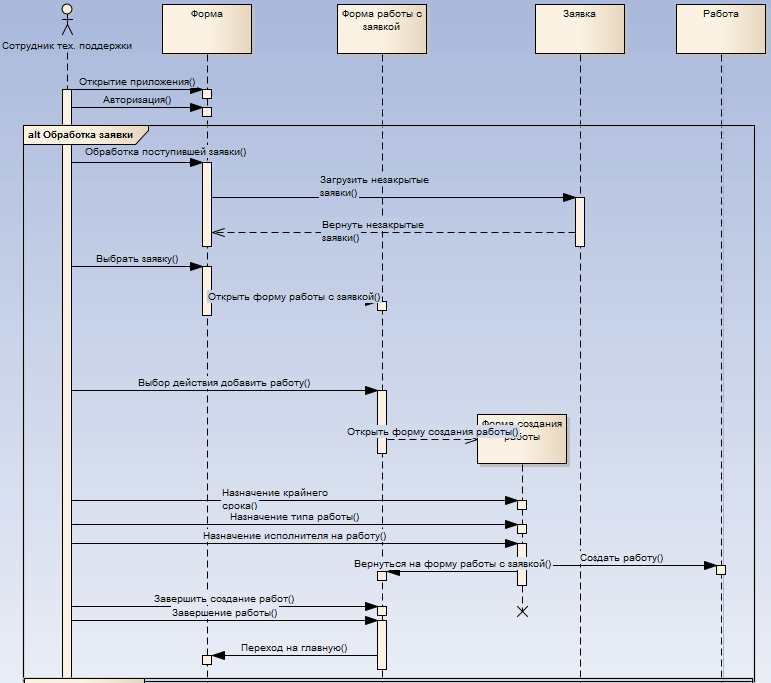
1. Пользователь
2. Форма
3. Форма создания заявки
4. Заявка

Выявлены следующие сообщения:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Вызывающий объект | Сообщение | Получатель сообщения | Примечание |
| 1 | Пользователь | Открыть приложение | Форма |  |
| 2 | Пользователь | Авторизация | Форма |  |
| 3 | Пользователь | Выбор действия | Форма |  |
| 4 | Пользователь | Запрос на формирование заявки | Форма |  |
| 5 | Форма | Открыть форму создания заявки | Форма создания заявки |  |
| 6 | Пользователь | Заполнить поля заявки | Форма создания заявки |  |
| 7 | Пользователь | Подтверждение ввода | Форма создания заявки |  |
| 8 | Форма создания заявки | Создать заявку | Заявка |  |
| 9 | Пользователь | Завершить работу | Форма создания заявки |  |
| 10 | Форма создания заявки | Закрыть форму | Форма создания заявки |  |
| 11 | Форма создания заявки | Перейти на главную | Форма |  |

|  |  |
| --- | --- |
| Операторы | Список сообщений |
| Форма | Открыть форму создания заявки. |
| Форма создания заявки | Создать заявку;  Закрыть форму;  Вернуться на главную. |

Диаграмма последовательности сценария «Работа с заявкой» приведена на рисунке 18.



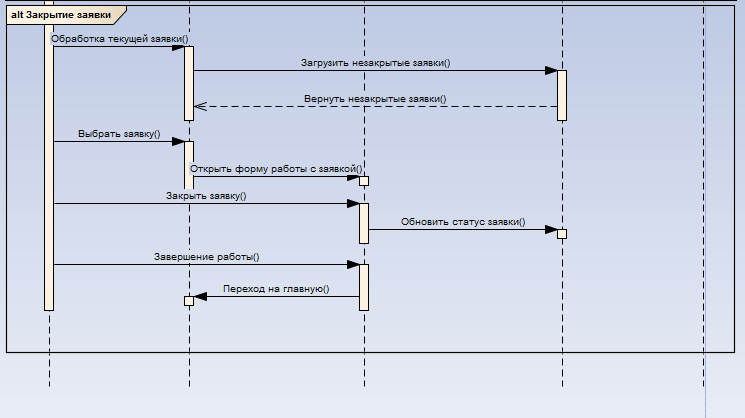


Рисунок 18. Работа с заявкой

В ходе построения объекты диаграммы выявлены следующие объекты:

1. Сотрудник тех. поддержки
2. Форма
3. Форма работы с заявкой
4. Заявка
5. Форма создания работы
6. Работа

Выявлены следующие сообщения:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Вызывающий объект | Сообщение | Получатель сообщения | Примечание |
| 1 | Сотрудник тех. поддержки | Открытие приложения | Форма |  |
| 2 | Сотрудник тех. поддержки | Авторизация | Форма |  |
| Сценарий добавления работ к заявке | | | | |
| 3.1 | Сотрудник тех. поддержки | Обработать поступившую заявку | Форма |  |
| 4.1 | Форма | Загрузить незакрытые заявки | Заявка |  |
| 5.1 | Заявка | Вернуть незакрытые заявки | Форма |  |
| 6.1 | Сотрудник тех. поддержки | Выбрать заявку | Форма |  |
| 7.1 | Форма | Открыть форму работы с заявкой | Форма работы с заявкой |  |
| 8.1 | Сотрудник тех. поддержки | Выбрать действие добавить работу | Форма работы с заявкой |  |
| 9.1 | Форма работы с заявкой | Открыть форму создания работы | Форма создания работы |  |
| 10.1 | Сотрудник тех. поддержки | Назначить крайний срок | Форма создания работы |  |
| 11.1 | Сотрудник тех. поддержки | Назначить тип работы | Форма создания работы |  |
| 12.1 | Сотрудник тех. поддержки | Назначить исполнителя на работу | Форма создания работы |  |
| 13.1 | Форма создания работы | Создать работу | Работа |  |
| 14.1 | Форма создания работы | Вернуться на форму работы с заявкой | Форма работы с заявкой |  |
| 15.1 | Сотрудник тех. поддержки | Завершить создание работ | Форма работы с заявкой |  |
| 16.1 | Сотрудник тех. поддержки | Завершить работу | Форма работы с заявкой |  |
| 17.1 | Форма работы с заявкой | Перейти на главную | Форма |  |
| Сценарий закрытия заявки | | | | |
| 3.2 | Сотрудник тех. поддержки | Обработать текущую заявку | Форма |  |
| 4.2 | Форма | Загрузить незакрытые заявки | Заявка |  |
| 5.2 | Заявка | Вернуть незакрытые заявки | Форма |  |
| 6.2 | Сотрудник тех. поддержки | Выбрать заявку | Форма |  |
| 7.2 | Форма | Открыть форму работы с заявкой | Форма работы с заявкой |  |
| 8.2 | Сотрудник тех. поддержки | Закрыть заявку | Форма работы с заявкой |  |
| 9.2 | Форма работы с заявкой | Обновить статус заявки | Заявка |  |
| 10.2 | Сотрудник тех. поддержки | Завершить работу | Форма работы с заявкой |  |
| 11.2 | Форма работы с заявкой | Перейти на главную | Форма |  |

|  |  |
| --- | --- |
| Операторы | Список сообщений |
| Форма | Загрузить закрытые заявки;  Открыть форму работы с заявкой. |
| Форма работы с заявкой | Открыть форму создания работы;  Обновить статус заявки;  Перейти на главную. |
| Форма создания работы | Создать работу;  Вернуться на форму работы с заявкой. |

Диаграмма последовательности сценария «Смена статуса работы» приведена на рисунке 19.

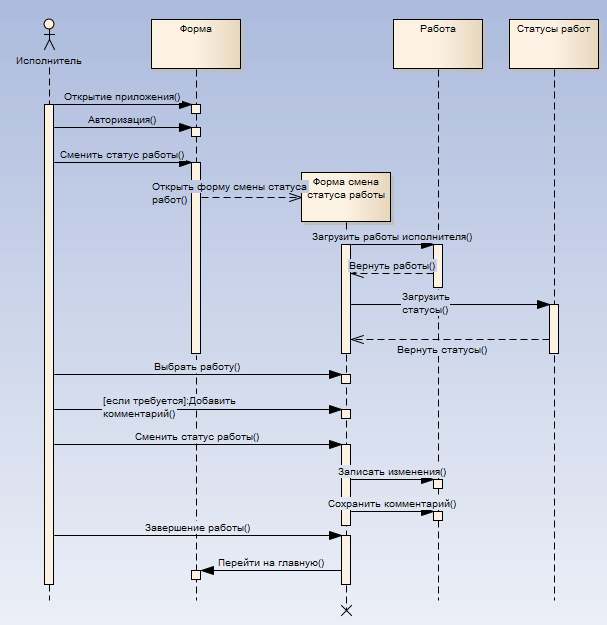


Рисунок 19. Смена статуса работы

В ходе построения объекты диаграммы выявлены следующие объекты:

1. Исполнитель
2. Форма
3. Форма смены статуса работы
4. Работа
5. Статусы работ

Выявлены следующие сообщения:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Вызывающий объект | Сообщение | Получатель сообщения | Примечание |
| 1 | Исполнитель | Открытие приложения | Форма |  |
| 2 | Исполнитель | Авторизация | Форма |  |
| 3 | Исполнитель | Сменить статус работы | Форма |  |
| 4 | Форма | Открыть форму смены статуса работы | Форма смены статуса работы |  |
| 5 | Форма смены статуса работы | Загрузить работы исполнителя | Работа |  |
| 6 | Работа | Вернуть работы | Форма смены статуса работы |  |
| 7 | Форма смены статуса работы | Загрузить статусы работ | Статусы работ |  |
| 8 | Статусы работ | Вернуть статусы | Форма смены статуса работы |  |
| 9 | Исполнитель | Выбрать работу | Форма смены статуса работы |  |
| 10 | Исполнитель | Добавить комментарий | Форма смены статуса работы | Если требуется |
| 11 | Исполнитель | Сменить статус работы | Форма смены статуса работы |  |
| 12 | Форма смены статуса работы | Записать изменения | Работа |  |
| 13 | Форма смены статуса работы | Сохранить комментарий | Работа |  |
| 14 | Исполнитель | Завершить работу | Форма смены статуса работы |  |
| 15 | Форма смены статуса работы | Перейти на главную | Форма |  |

|  |  |
| --- | --- |
| Операторы | Список сообщений |
| Форма | Открыть форму смены статуса работы. |
| Форма смены статуса работы | Загрузить работы исполнителя;  Загрузить статусы;  Записать изменения;  Сохранить комментарий;  Перейти на главную. |

Диаграмма последовательности сценария «Авторизация» приведена на рисунке 20.

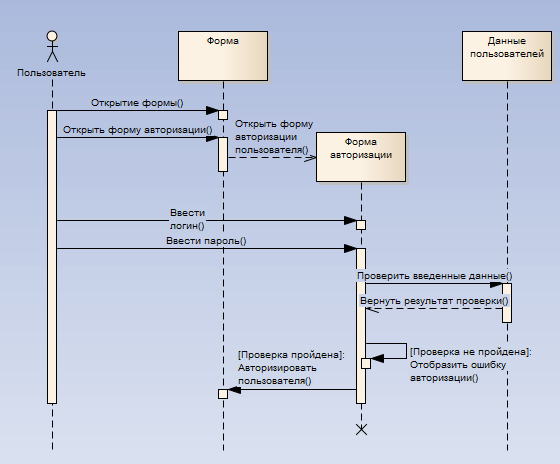


Рисунок 20. Авторизация

В ходе построения объекты диаграммы выявлены следующие объекты:

1. Исполнитель
2. Форма
3. Форма авторизации
4. Данные пользователя

Выявлены следующие сообщения:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Вызывающий объект | Сообщение | Получатель сообщения | Примечание |
| 1 | Исполнитель | Открытие приложения | Форма |  |
| 2 | Исполнитель | Открыть форму авторизации | Форма |  |
| 3 | Форма | Открыть форму авторизации пользователя | Форма авторизации |  |
| 4 | Исполнитель |  | Форма авторизации |  |
| 5 | Исполнитель | Загрузить работы исполнителя | Форма авторизации |  |
| 6 | Форма авторизации | Проверить введенные данные | Данные пользователей |  |
| 7 | Данные пользователей | Вернуть результаты проверки | Форма авторизации |  |
| 8 | Форма авторизации | Отобразить ошибку авторизации | Форма авторизации | Проверка не пройдена |
| 9 | Форма авторизации | Авторизовать пользователя | Форма | Проверка пройдена |

|  |  |
| --- | --- |
| Операторы | Список сообщений |
| Форма | Открыть форму авторизации. |
| Форма авторизации | Проверить введенные данные;  Отобразить ошибку авторизации;  Авторизовать пользователя. |

Диаграмма последовательности сценария «Журналирование пользователей в системе» приведена на рисунке 21.

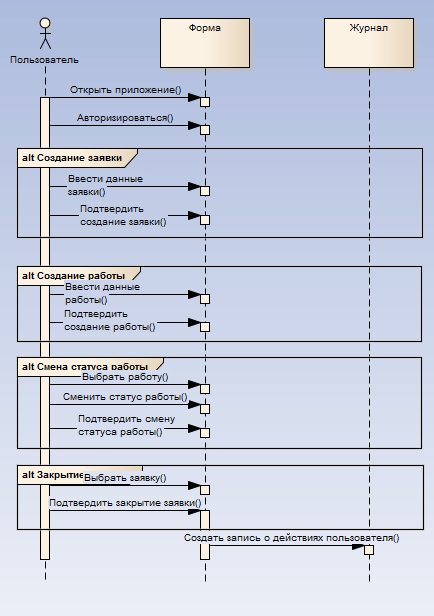


Рисунок 21. Авторизация

В ходе построения объекты диаграммы выявлены следующие объекты:

1. Пользователь
2. Форма
3. Журнал

Выявлены следующие сообщения:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Вызывающий объект | Сообщение | Получатель сообщения | Примечание |
| 1 | Пользователь | Открытие приложения | Форма |  |
| 2 | Пользователь | Авторизация | Форма |  |
| Создание заявки | | | | |
| 3.1 | Пользователь | Ввести данные заявки | Форма |  |
| 3.2 | Пользователь | Подтвердить создание заявки | Форма |  |
| Создание работы | | | | |
| 4.1 | Пользователь | Ввести данные работы | Форма |  |
| 4.2 | Пользователь | Подтвердить создание работы | Форма |  |
| Смена статуса работы | | | | |
| 5.1 | Пользователь | Выбрать работу | Форма |  |
| 5.2 | Пользователь | Сменить статус работы | Форма |  |
| 5.3 | Пользователь | Подтвердить смену статуса работы | Форма |  |
| Закрытие заявки | | | | |
| 6.1 | Пользователь | Выбрать заявку | Форма |  |
| 6.2 | Пользователь | Подтвердить закрытие заявки | Форма |  |
| 7 | Форма | Создать запись о действиях пользователя | Журнал |  |

|  |  |
| --- | --- |
| Операторы | Список сообщений |
| Форма | Создать запись о действиях пользователя. |

Выявленные методы отображены выше в диаграмме классов-анализа рисунок 17.

## **Выбор технологии реализации**

Система будет реализована на языке программирования C# в среде разработке Microsoft Visual Studio.

Пользователь будет взаимодействовать с системой через интерфейс в виде веб-страницы ASP.NET MVC приложения.

Реализация требования возможна в рамках операционных систем семейства Windows, которые установлены на предприятии.

Пользовательские компоненты будут взаимодействовать с данными с помощью технологии ADO.NET, которая будет реализована через соответствующие компоненты среды проектирования Visual Studio.

Для развертывания приложения будет использоваться:

* Клиентский узел для установки клиентского приложения. Характеристики:
  + CPU: AMD Athlon-5150 1,6Ghz;
  + ОЗУ: 2 ГБ;
  + HDD: 250 ГБ.

Серверный узел для установки базы данных. Характеристики:

* + CPU: Intel core i5 3Ghz;
  + ОЗУ: 6 ГБ;
  + HDD: 1000 ГБ.

Диаграмма начальной среды развертывания приведена на рисунке 22:

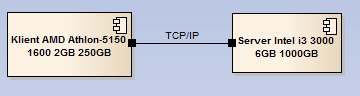


Рисунок 22. Начальная среда развертывания

## **Проектирование хранилища данных**

Предполагаемые состав сущностей, отображаемых в БД приведен на рисунке 23:

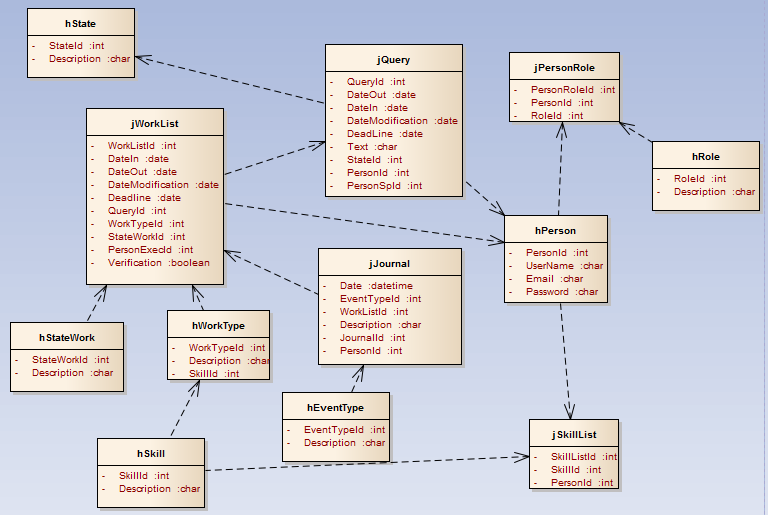


Рисунок 23. Состав сущностей

На основе состава сущностей, созданы таблицы БД:

* "hState" – хранит в себе возможные статусы заявки. Атрибуты:
  + StateId, тип int – искусственный атрибут, хранит в себе идентификатор статуса;
  + Description, тип char - содержит наименование статуса заявки.

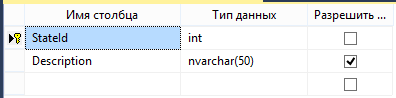


Рисунок 24. Таблица hState

* "hStateWork" – хранит в себе возможные статусы работы. Атрибуты:
  + StateWorkId, тип int – искусственный атрибут, хранит в себе идентификатор статуса;
  + Description, тип char - содержит наименование статуса работы.

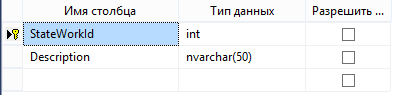


Рисунок 25. Таблица hStateWork

* "hWorkType" – хранит в себе возможные типы работы. Атрибуты:
  + WorkTypeId, тип int – искусственный атрибут, хранит в себе идентификатор типа работы;
  + Description, тип char - содержит наименование типа работы;
  + SkillId, тип int – хранит в себе идентификатор специальности исполнителя, способного выполнить работу.

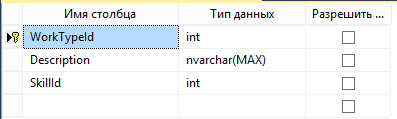


Рисунок 26. Таблица hWorkType

* "hSkill" - хранит в себе возможные типы специальностей исполнителей.
  + SkillId, тип int – искусственный атрибут, хранит в себе идентификатор специальности;
  + Description, тип char - содержит наименование специальности.

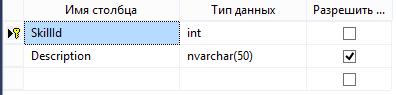


Рисунок 27. Таблица hSkill

* "jWorkList" - хранит в себе записи о работах по заявкам. Атрибуты:
  + WorkListId, тип int - искусственный атрибут, хранит в себе идентификатор записи;
  + DateIn, тип date - дата назначения работы на исполнителя;
  + DateOut, тип date - дата выполнения работы исполнителем;
  + DateModification, тип date - дата последнего изменения статуса работы;
  + Deadline, тип date - крайний срок выполнения работы исполнителем;
  + QueryId, тип int - хранит в себе идентификатор заявки к которой относится работа;
  + WorkTypeId, тип int – хранит в себе идентификатор типа назначенной работы;
  + StateWorkId, тип int – хранит в себе идентификатор текущего статуса работы;
  + PersonExecId, тип int - хранит в себе идентификатор исполнителя работы;
  + Verification, тип boolean - показывает подтвердил ли сотрудник тех. поддержки выполнение работы.

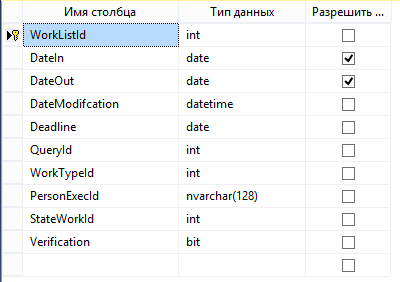


Рисунок 28. Таблица jWorkList

* jQuery - хранит в себе оставленные пользователями заявки. Атрибуты:
  + QueryId, тип int - искусственный атрибут, хранит в себе идентификатор заявки;
  + DateIn, тип date - дата составления заявки;
  + DateOut, тип date - дата закрытия заявки;
  + DateModification, тип date - дата последнего изменения статуса заявки;
  + Deadline, тип date - крайний срок выполнения заявки;
  + Text, тип char - содержит текст заявки;
  + StateId, тип int – хранит в себе идентификатор текущего статуса заявки;
  + PersonId, тип int - хранит в себе идентификатор составителя заявки;
  + PersonSpId, тип int - хранит в себе идентификатор сотрудника обработавшего заявку.

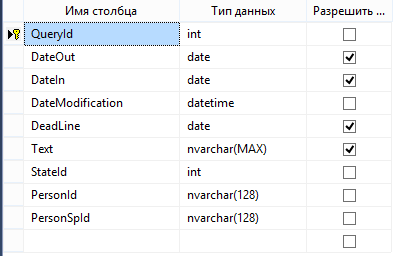


Рисунок 29. Таблица jQuery

* hPerson - хранит в себе информацию о персонах, работающих в КОМК. Атрибуты:
  + PersonId, тип int - искусственный атрибут, хранит в себе идентификатор персоны;
  + UserName, тип char - содержит ФИО сотрудника;
  + Email, тип char - содержит Email сотрудника;
  + Password, тип char - содержит пароль сотрудника.

Для облегчения разработки была автоматически сгенерирована таблица, выполняющая функции таблицы hPerson - AspNetUsers:

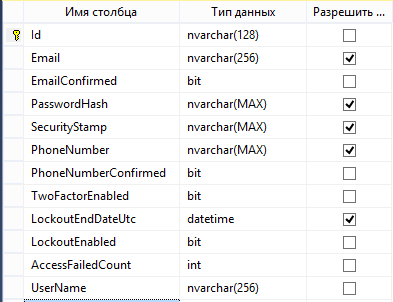


Рисунок 30. Таблица AspNetUsers

* hRole - хранит в себе информацию о ролях сотрудников, работающих в КОМК. Атрибуты:
  + RoleId, тип int - искусственный атрибут, хранит в себе идентификатор роли;
  + Description, тип char - содержит наименование роли.

Для облегчения разработки была автоматически сгенерирована таблица, выполняющая функции таблицы hRole - AspNetRoles:

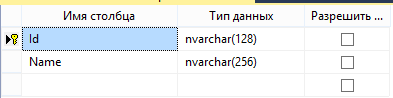


Рисунок 31. Таблица AspNetRoles

* jPersonRole - таблица, обеспечивающая связь многие к многим между ролями и пользователями. Атрибуты:
  + RoleId, тип int - искусственный атрибут, хранит в себе идентификатор роли;
  + PersonId, тип int - искусственный атрибут, хранит в себе идентификатор персоны;
  + PersonRoleId, тип int - искусственный атрибут, хранит в себе идентификатор записи.

Для облегчения разработки была автоматически сгенерирована таблица, выполняющая функции таблицы jPersonRole - AspNetUserRoles:

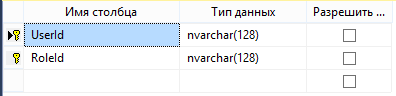


Рисунок 32. Таблица AspNetUserRoles

* jSkillList - хранит в себе информацию о специальностях сотрудника. Атрибуты:
  + SkillListId, тип int - искусственный атрибут, хранит в себе идентификатор записи;
  + PersonId, тип int - хранит в себе идентификатор сотрудника;
  + SkillId, тип int - хранит в себе идентификатор сотрудника специальности.

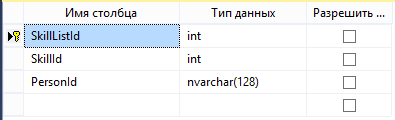


Рисунок 33. Таблица jSkillList

* "jJournal" – хранит в себе историю действий пользователей в системе. Атрибуты:
  + Date, тип datetime – содержит дату возникновения события;
  + EventTypeId, тип int - содержит идедификатор типа события;
  + WorkListId, тип int - содержит идентификатор работы вызвавшей событие;
  + Descrition, тип char - содержит описание произошедшего события;
  + JournalId, тип int - искусственный атрибут, хранит в себе идентификатор записи журнала;
  + PersonId, тип int - содержит идентификатор персоны вызвавшей событие.

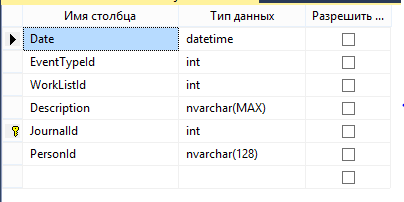


Рисунок 34. Таблица jJournal

* "hEventType" – хранит в себе типы событий журнала. Атрибуты:
  + EventTypeId, тип int – искусственный атрибут, хранит в себе идентификатор события;
  + Description, тип char - содержит наименование события.

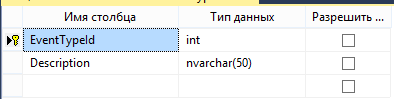
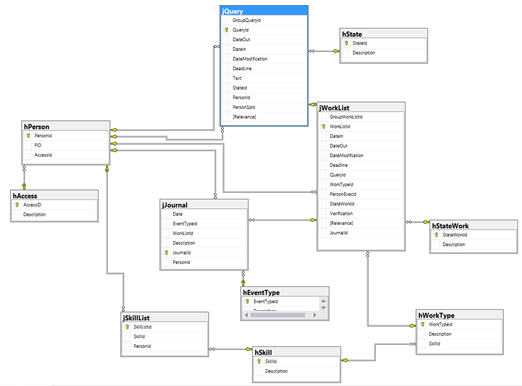


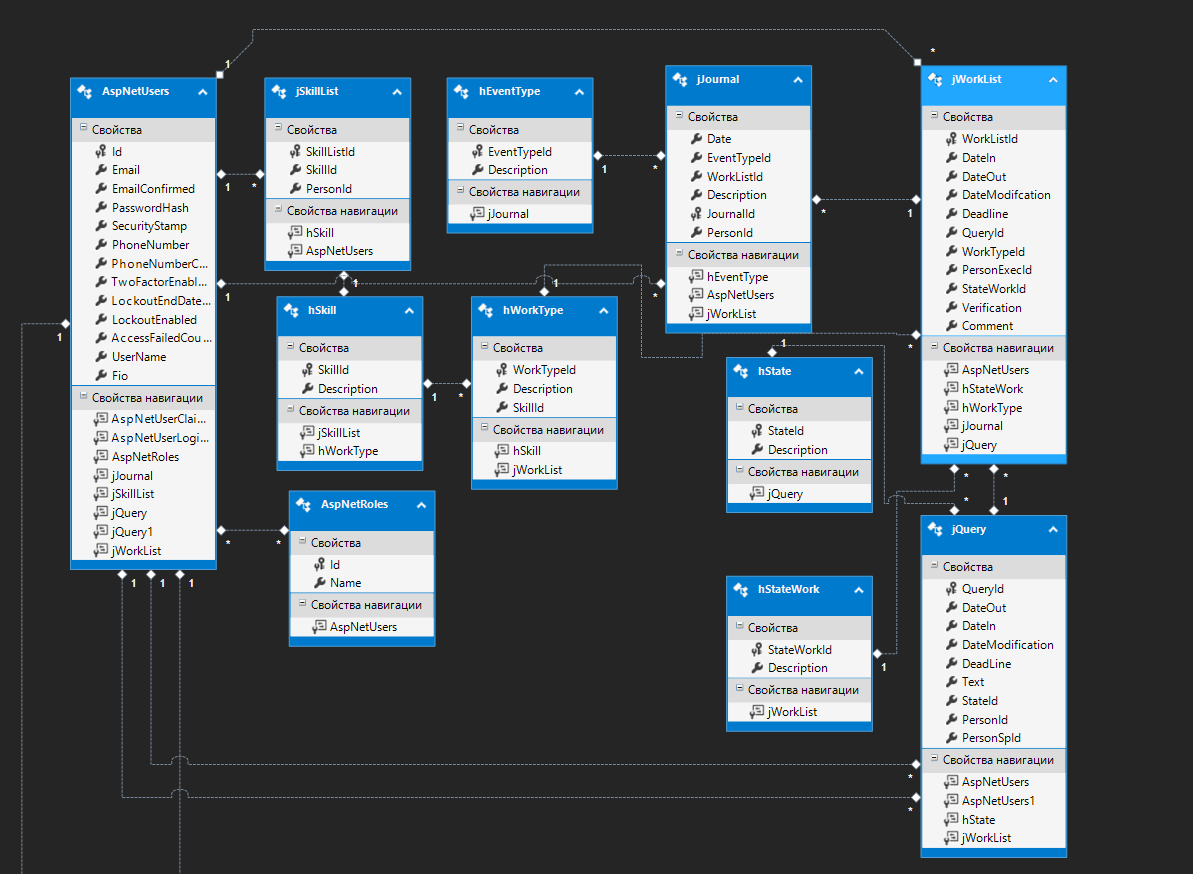
Рисунок 35. Таблица hEventType

# **Разработка программного кода**

Так как к моменту создания клиента база данных уже была разработана, то логичным было использовать технологию Entity Framework, а конкретнее Database First.



После создания модели подход Database First был более не актуален и мы перешли к подходу Code First, с помощью его мы добавили в бд, таблицы для работы с пользователями и все дальнейшие изменения производили непосредственно через код.

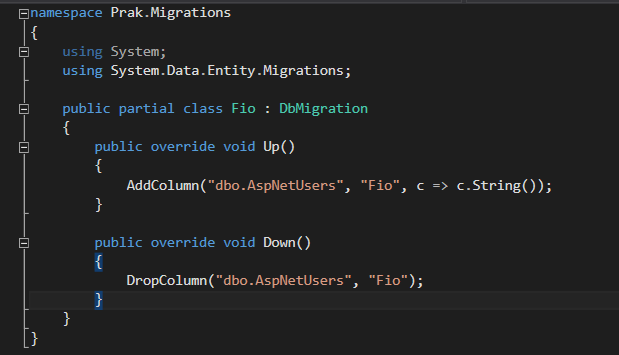


Для того чтобы вносить изменения в бд необходимо использовать миграции.

Создаем миграцию:



Смотрим созданную миграцию:

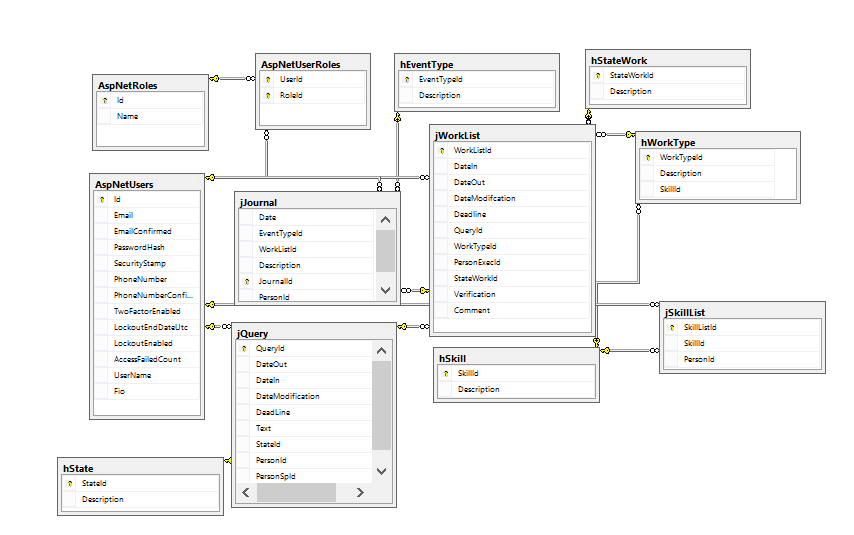


В данной миграции мы добавляем поле ФИО в таблицу описывающую пользователя.

Чтобы миграция прошла нужно запустить команду:



Теперь после всех модификаций проверяем нашу бд:



Состав компонентов, содержащих программный код представлен на рисунке 36:

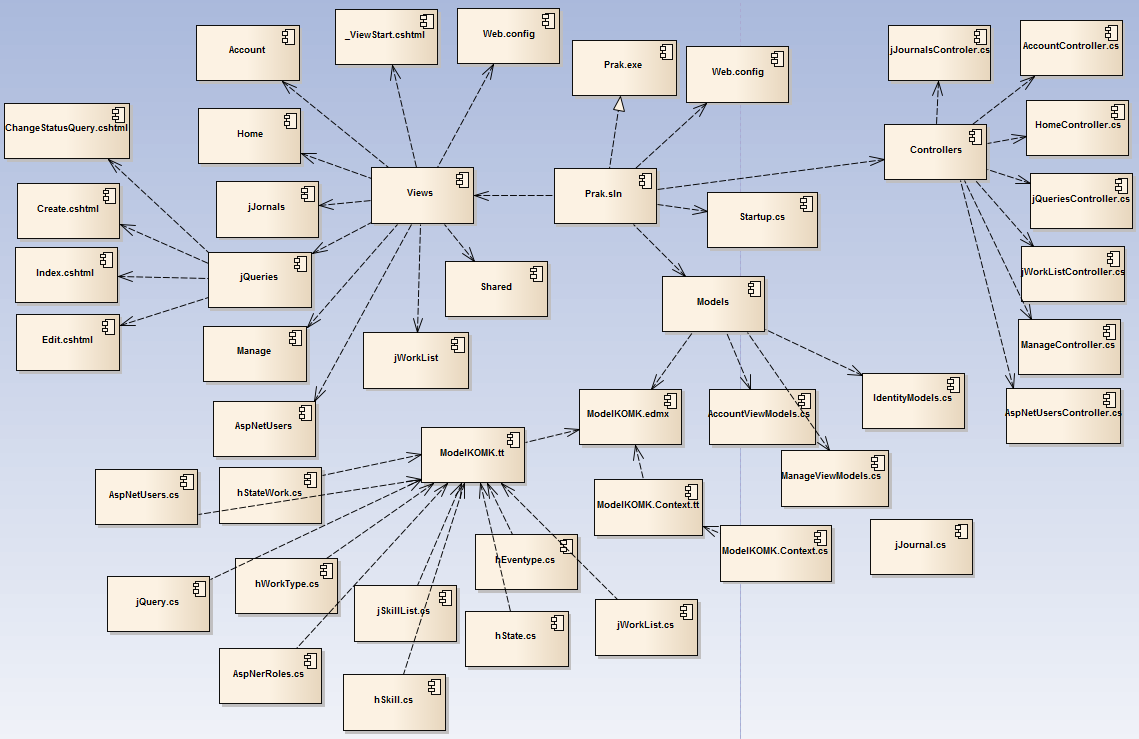
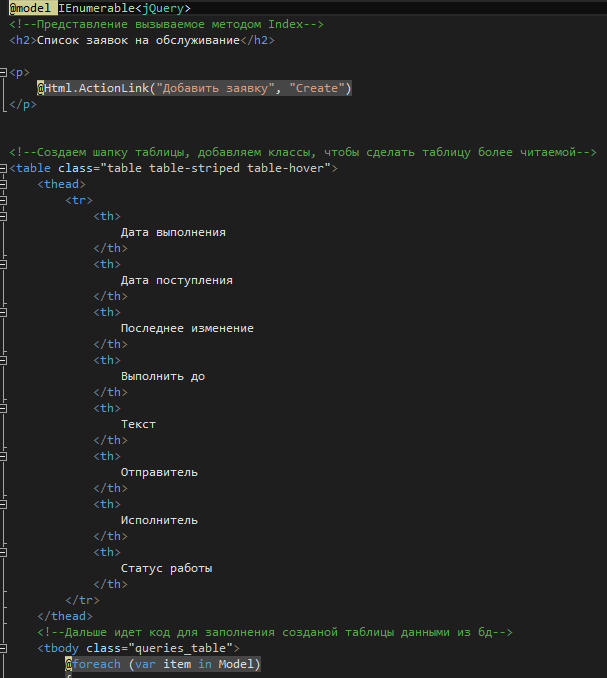


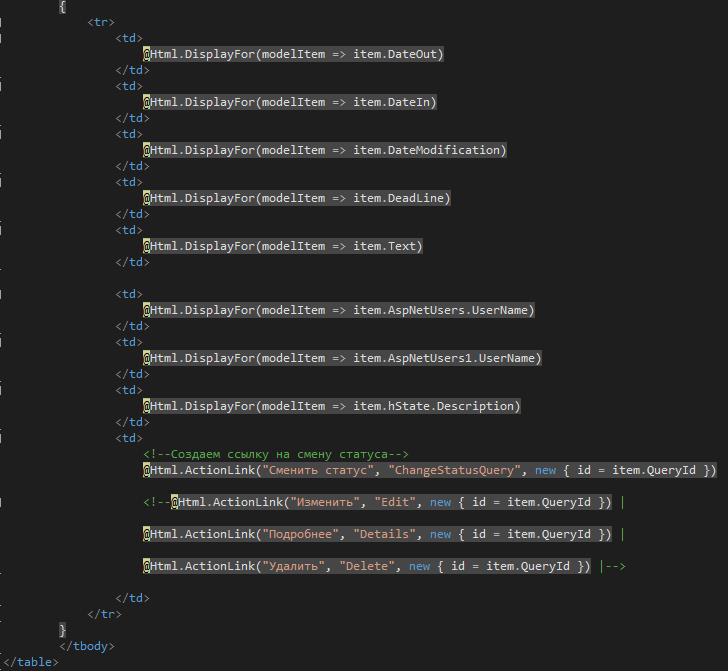
Рисунок 36. Состав компонентов программного кода

Проект реализован в виде

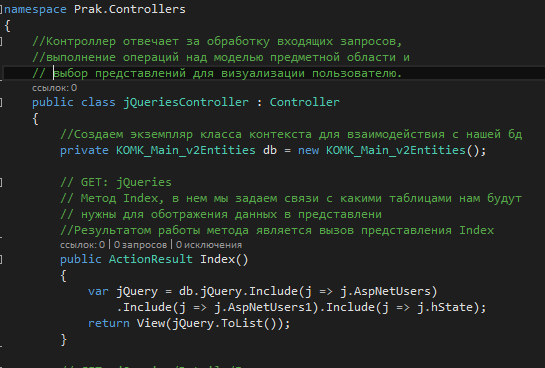
* Prak.sln, который организует проекты, элементы проектов и решений в решение;
* Prak.exe – исполняемый файл проекта.
* Startup.cs – запуска программы.
* Web.config – файл соединения и конфигурации приложения.
* Пакет Views – содержит отображаемые приложением веб-страницы
  + Account - пакет содержащий страницы, отвечающие за аккаунтнг пользователя;
  + AspNetUsers - пакет содержащий страницы, отображающие пользователей системы;
  + Home - пакет содержащий описание домашней страницы приложения;
  + jJournals - пакет содержащий страницы для работы с заявками пользователей;
  + jQueries - пакет содержащий страницы для работы с заявками пользователей;
    - ChangeStausQuery.cshtml;
    - Create.cshtml;
    - Edit.cshtml;
    - Index.cshtml.

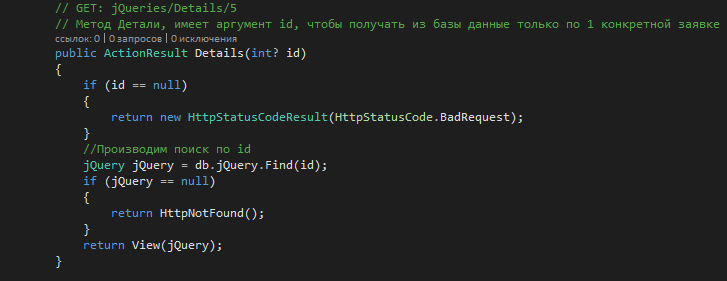
Пример кода представления на основе страницы Index.cshtml пакета jQueries представлен на рисунках ниже.

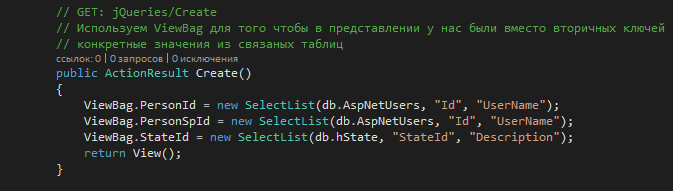


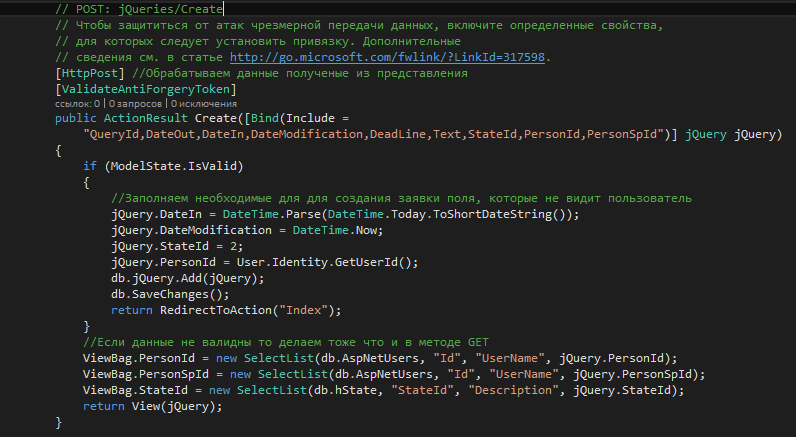


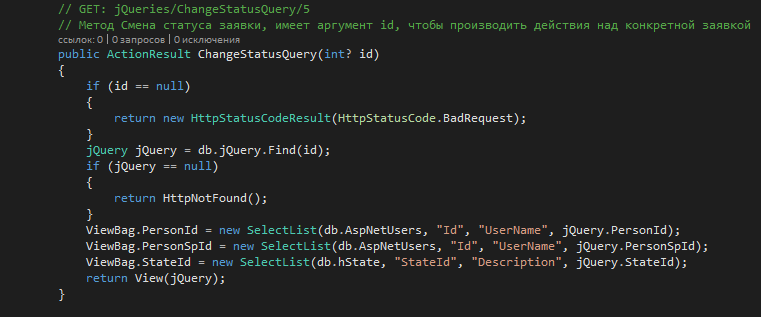
* + jWorkList - пакет содержащий страницы для работы с работами по заявкам;
  + Manage - пакет содержащий страницы, отвечающие за регистрацию пользователя в системе;
  + Shared - пакет содержащий элементы страниц, присущие всем страницам;
  + \_ViewStart.cshtml - содержит ссылку на настройки стартовой страницы приложения;
  + Web.config – файл соединения и конфигурации приложения.
* Controllers - пакет, содержащий контроллеры обрабатывающие действия пользователей со страницами, контроллеры обрабатывают действия на одноименных веб-страницах;
  + jQueriesController.cs - обрабатывает действия пользователя с моделью jQueries, в контроллере реализованы методы: ChangeStatusQuery(), Create(), Index(), Details(). Код описывающий контроллер приведен на рисунках ниже, код, описывающий остальные контроллеры, схож с приведенным;

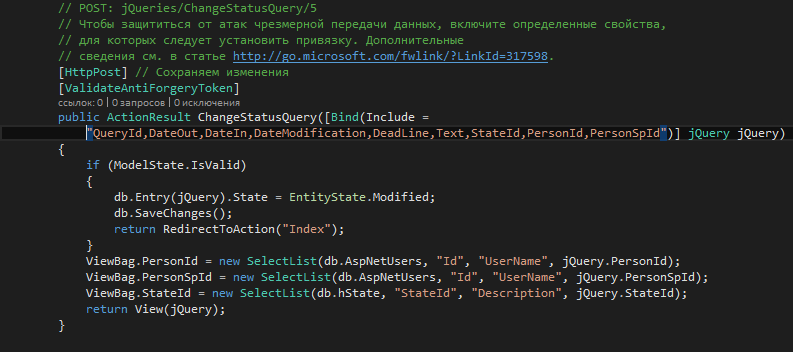




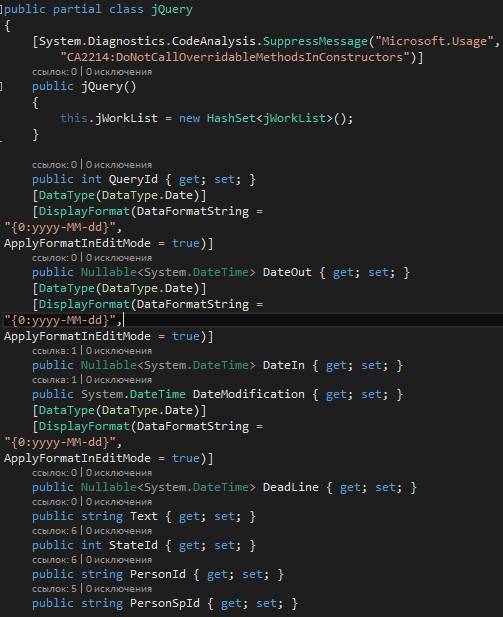


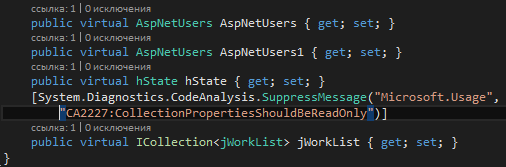






* + jWorkListController.cs - обрабатывает действия пользователя с моделью jWorkList, в контроллере реализованы методы: ChangeStateWork(), Create(), Index();
  + AccountController.cs - обрабатывает запросы пользователя по взаимодействию с моделью AccountViewModel, в контроллере реализованы методы: \_ExternalLoginsListPartial(), ConfirmEmail(), ExternalLoginConfirmation(), ExternalLoginFailure(), ForgotPassword(), ForgotPasswordConfirmation(), Login(), Register(), ResetPassword(), ResetPasswordConfirmation(), SendCode(), VerifyCode();
  + HomeController.cs - обрабатывает действия пользователя на главной странице приложения, в контроллере реализованы методы:About(), Contact(), Index();
  + ManageController.cs - обрабатывает действия пользователя с моделью ManageViewModels, в контроллере реализованы методы:AddPhoneNumber(), ChangePassword(), Index(), ManageLogins(), SetPassword(), VeryfyPhoneNumber();
  + jJournalsController.cs - обрабатывает действия пользователя с моделью jJournal, в контроллере реализованы методы: Create(), Index();
  + AspNetUsersController.cs - обрабатывает действия пользователя с моделью AspNetUsers, в контроллере реализованы методы: Index().
* Models - пакет, содержащий представления БД, через которые ведется работы с таблицами БД:
  + IdentityModels.cs;
  + ManageViewModels.cs;
  + AccountViewModels.cs - данные модели отвечают за описание поведения стандартной авторизации и управления авторизованными пользователями;
  + При формировании моделей использовался подход DataBase First. Была сформирована модель базы данных ModelKOMK.edmx. В сформированной модели присутствуют классы:
    - jQuery.cs - соответствует таблице БД jQuery, код автоматически сгенерированный средой разработки представлен на рисунках ниже, принцип формирования остальных классов схож с данным;





* + - jWorkList.cs - соответствует таблице БД jWorkList;
    - hEventType.cs - соответствует таблице БД hEventType;
    - hSkill.cs - соответствует таблице БД hSkill;
    - hState.cs - соответствует таблице БД hState;
    - hStateWork.cs - соответствует таблице БД hStateWork;
    - hWorkType.cs - соответствует таблице БД hWorkType;
    - jJournal.cs - соответствует таблице БД jJournal;
    - jSkillList.cs - соответствует таблице БД jSkillList;
    - AspNetUsers.cs - соответствует таблице БД AspNetUsers;
    - AspNerRoles.cs - соответствует таблице БД AspNetRoles.

Реализация работы журнала и конфигурирования системы в данном отчете рассмотрены не будут, т.к. они имеют косвенное отношение к работе с заявками пользователей.

# **Развертывание**

Диаграмма среды развертывания представлена на рисунке 37:

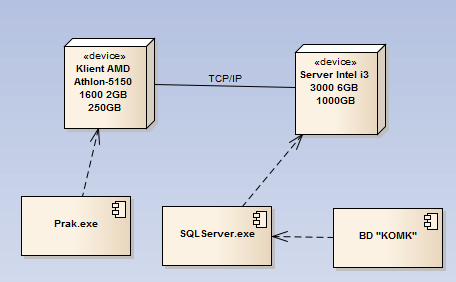


Рисунок 37. Среда развертывания

На серверной машине предприятия уже установлен сервер SqlServer. На сервер необходимо развернуть базу данных KOMK.

На клиентских машинах необходимо установить .NET Framework, а так же скопировать файл запуска программы (exe-файл).

После загрузки всех компонентов необходимо провести тестовый запуск программы.

## **Интерфейс системы**

Главная страница приложения представлена на рисунке 38. Реализованы возможности создания заявки, работы, смены статуса заявки, работы. Любые действия с заявками и работами возможны только после авторизации пользователя.

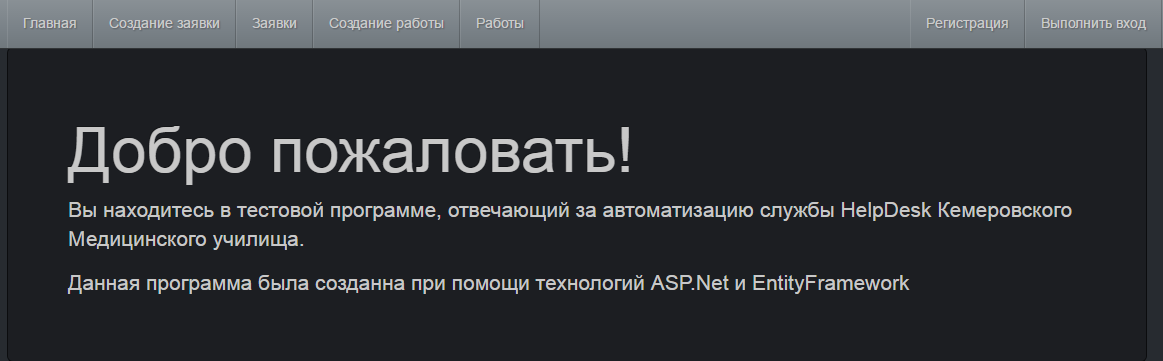


Рисунок 38. Главная страница приложения

Страница создания заявки представлена на рисунке 39. Реализованы возможность добавления заявки.

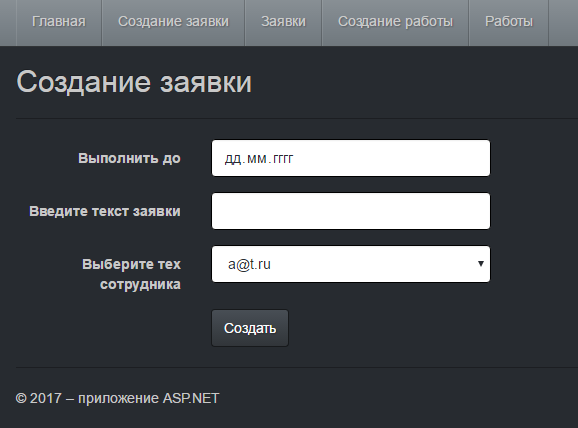


Рисунок 39. Страница создания заявки

Страница просмотра всех заявок представлена на рисунке 40, реализована возможность смены статуса заявки.

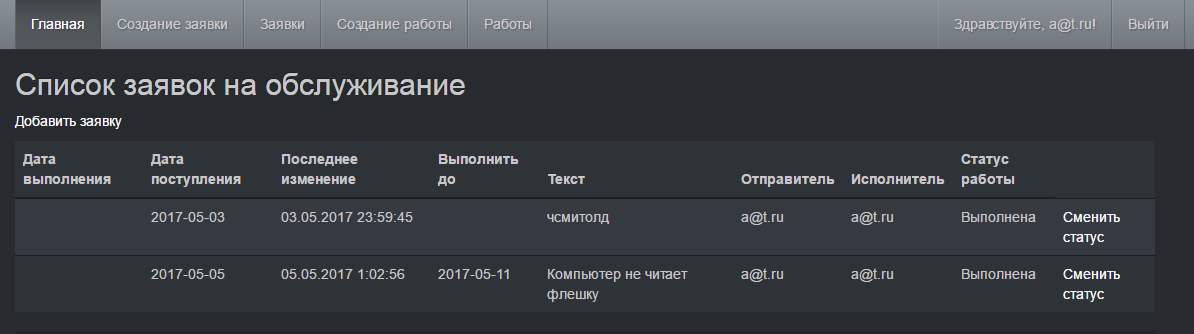


Рисунок 40. Страница просмотра всех заявок

Страница смены статуса заявки представлена на рисунке 41. Реализована возможность смены статуса заявки.

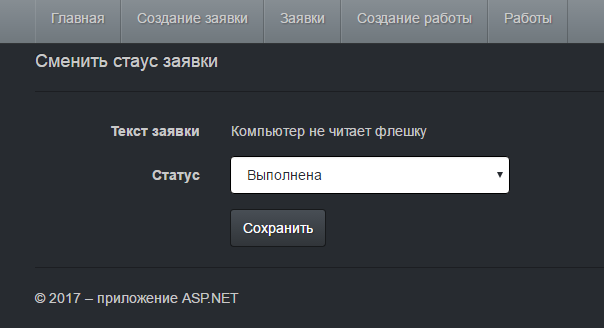


Рисунок 41. Страница смены статуса заявки

Страница создания работы представлена на рисунке 42. Реализована возможность создания работы.

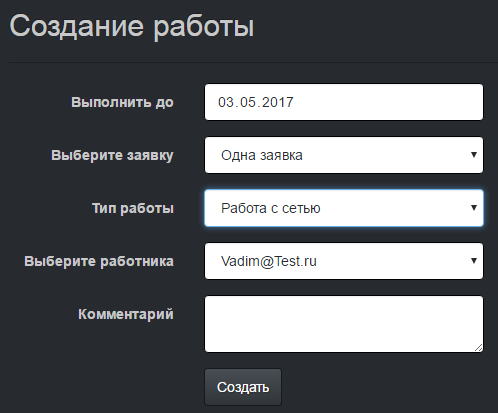


Рисунок 42. Страница создания работы

Страница просмотра работ представлена на рисунке 43. Реализована возможность смены статуса работы.

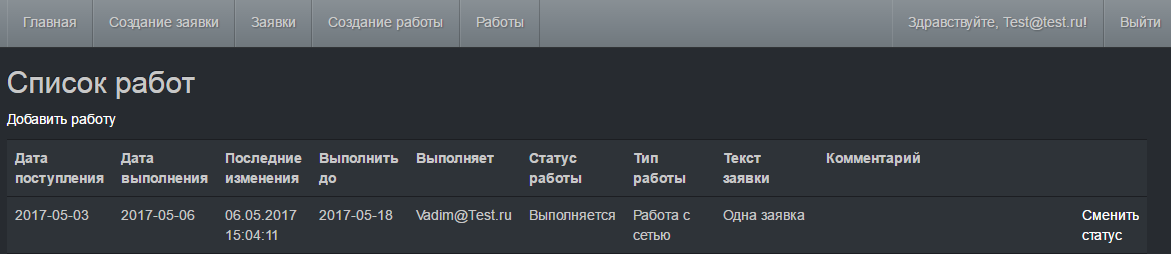


Рисунок 43. Страница просмотра работ.

Страница смены статуса работы представлена на рисунке 44. Реализована возможность смены статуса работы и добавления комментария к работе.

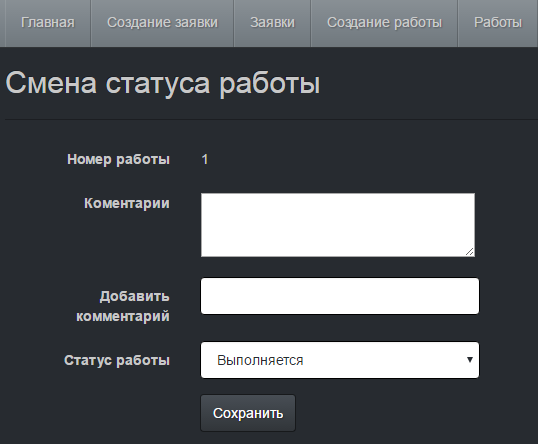


Рисунок 44. Страница смены статуса работы.

Страница авторизации пользователя представлена на рисунке 45. Реализована возможность авторизации пользователя.

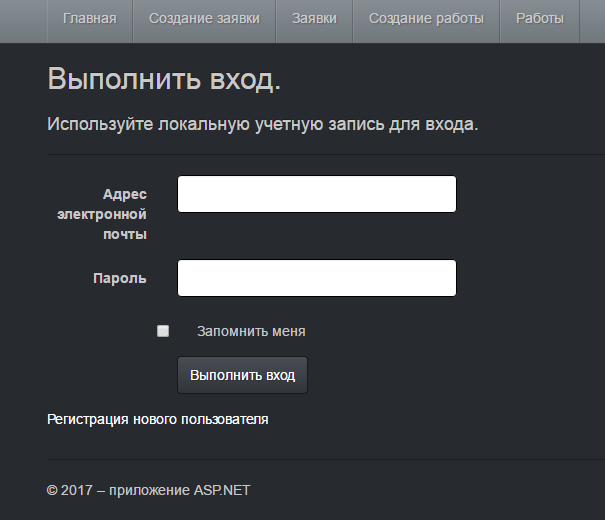


Рисунок 45. Страница авторизации пользователя

Страница регистрации пользователя представлена на рисунке 46. Реализована возможность регистрации пользователя.

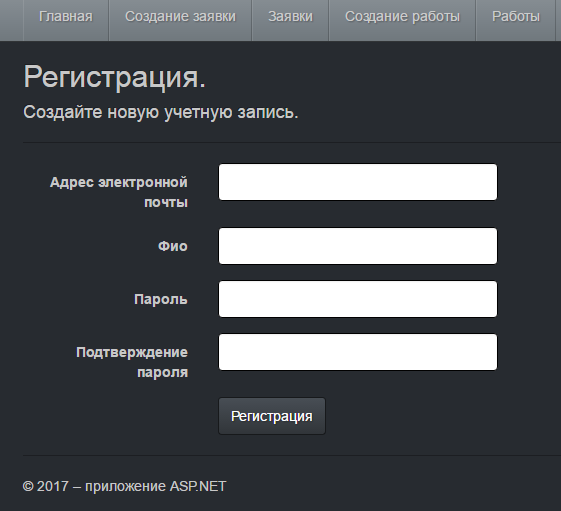


Рисунок 46. Страница регистрации пользователя

# **Разработка сценариев тестирования**

Для тестирования выбраны следующие сценарии «Создание заявки», «Создание работы», «Закрытие заявки», «Смена статуса работы».

Сценарий «Создание заявки»:

1. Пользователь авторизируется.

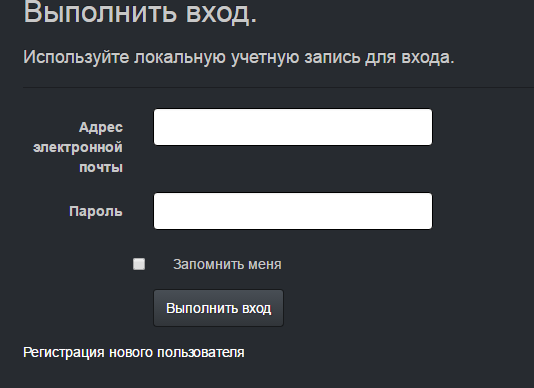


Рисунок 32. Вкладка «Авторизация»

1. Открывает вкладку «Создание заявки».

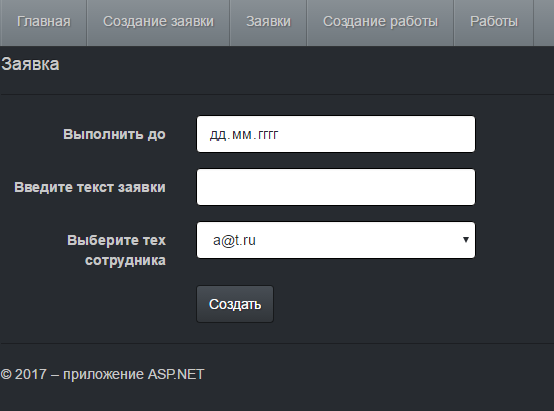


Рисунок 33. Вкладка «Создание заявки»

1. На вкладке «Создание заявки» необходимо заполнить поля данных заявки (срок выполнения, текст заявки, какому сотруднику послать заявку). После нажатия кнопки «Создать» в таблицу базы данных jQuery добавляется запись о созданной заявке, а на вкладке «Заявки» обновляется список заявок.

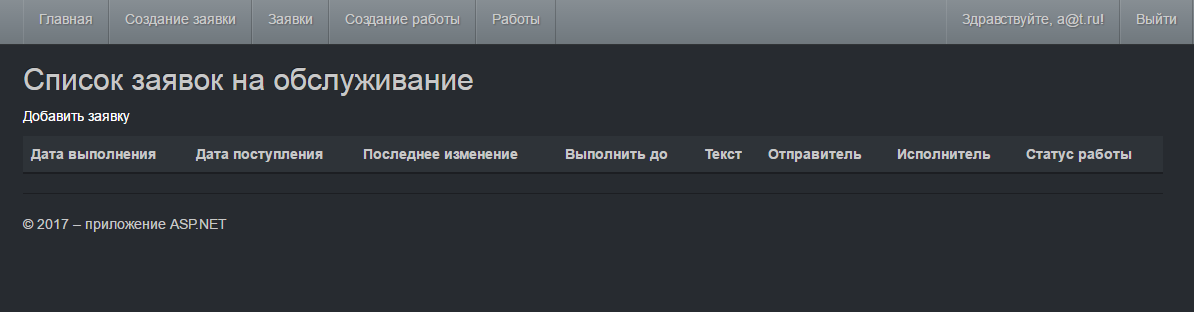


Рисунок 34. Вкладка «Заявки»

Сценарий «Создание работы»:

1. Пользователь авторизируется.

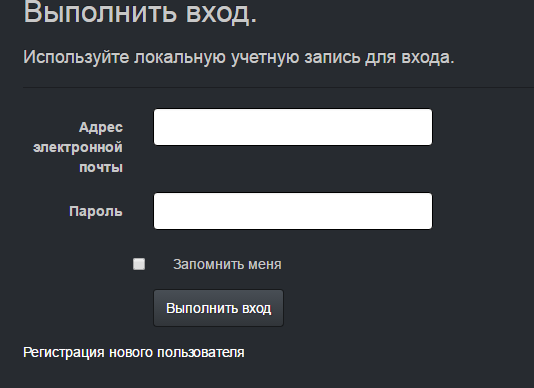


Рисунок 35. Вкладка «Авторизация»

1. Открывает вкладку «Создание работы».

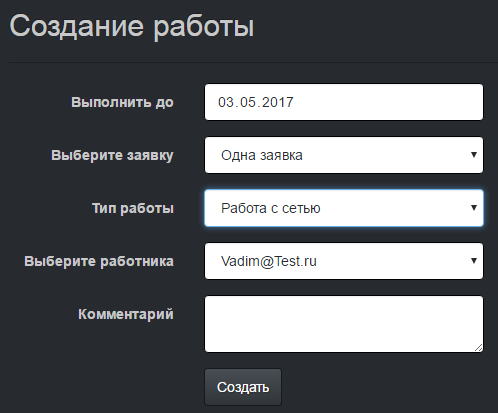


Рисунок 36. Вкладка «Создание работы»

1. На вкладке «Создание работы» необходимо заполнить поля данных заявки (срок выполнения, заявка к которой относится работа, тип работы, исполнитель). После нажатия кнопки «Создать» в таблицу базы данных jWorkList добавляется запись о созданной работе, а на вкладке «Работы» обновляется список работ.

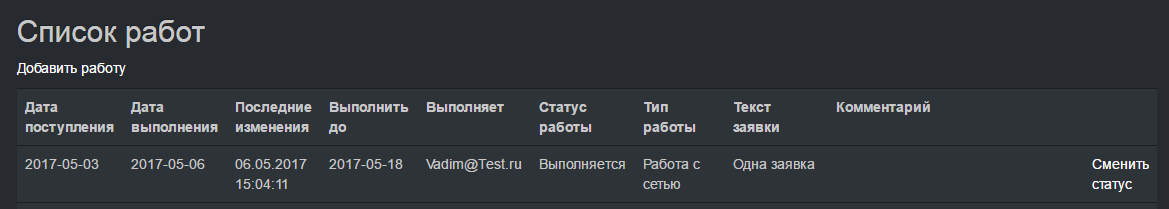


Рисунок 37. Вкладка «Работы»

Сценарий «Закрытие заявки»:

1. Пользователь авторизируется.

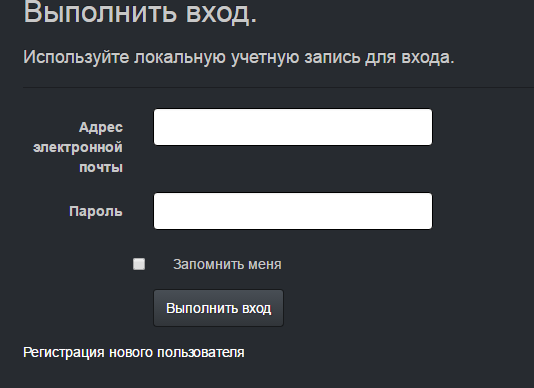


Рисунок 38. Вкладка «Авторизация»

1. Переходит на вкладку «Заявка».

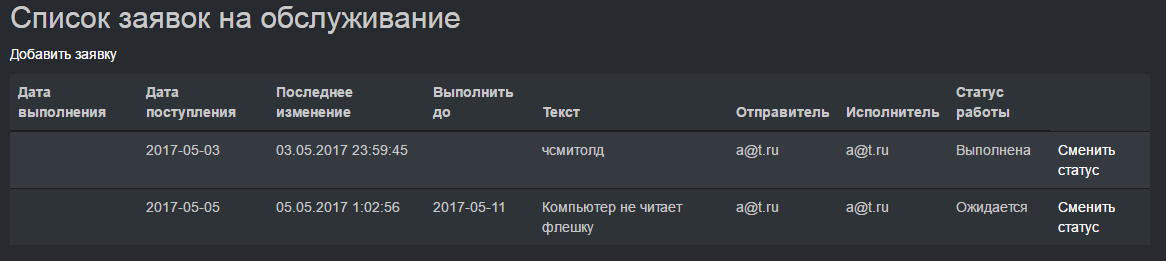


Рисунок 39. Вкладка «Заявки»

1. Выбирает требуемую заявку и переходит по ссылке «Сменить статус». На открывшейся вкладке меняют статус заявки на “выполнена” и сохраняет статус.

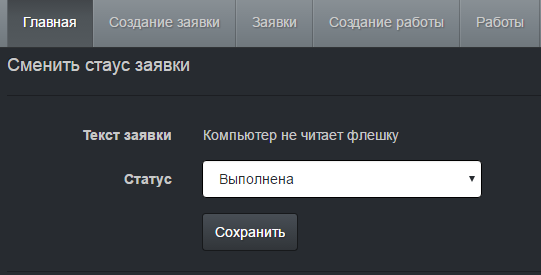


Рисунок 40. Вкладка «Смена статуса заявки»

1. На вкладке «Заявка» должен обновится статус заявки.

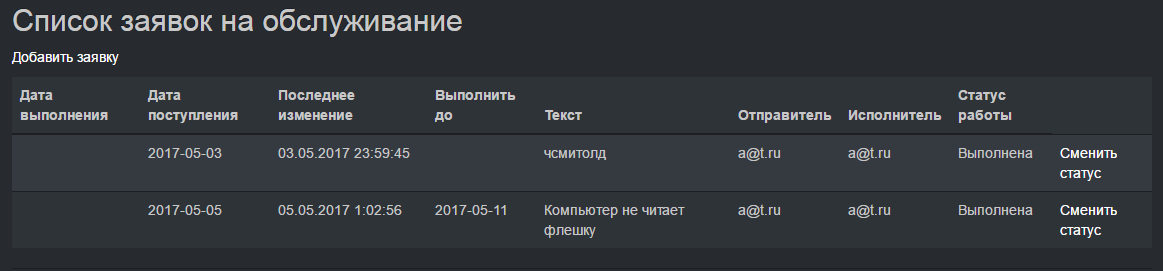


Рисунок 41. Вкладка «Заявки»

Сценарий «Смена статуса работы»:

1. Пользователь авторизируется.

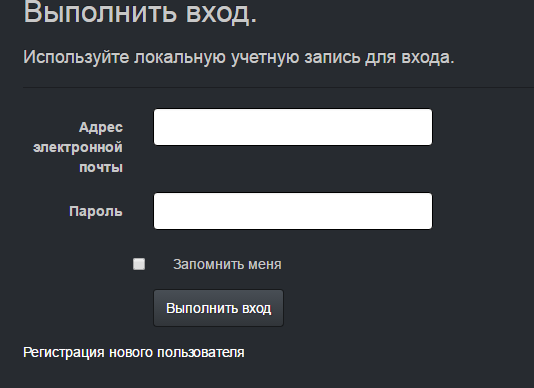


Рисунок 42. Вкладка «Авторизация»

1. Переходит на вкладку «Работы».

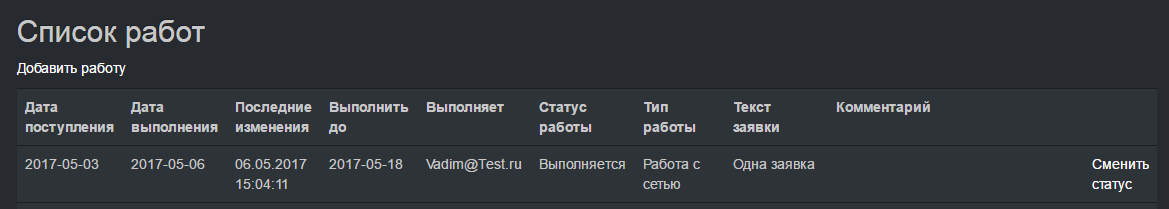


Рисунок 43. Вкладка «Работы»

1. Выбирает работу, статус которой надо сменить и переходит по ссылке «Сменить статус». На открывшейся вкладке меняют статус работы, при желании пользователь может оставить комментарий к работе. Сделанные изменения сохраняются.

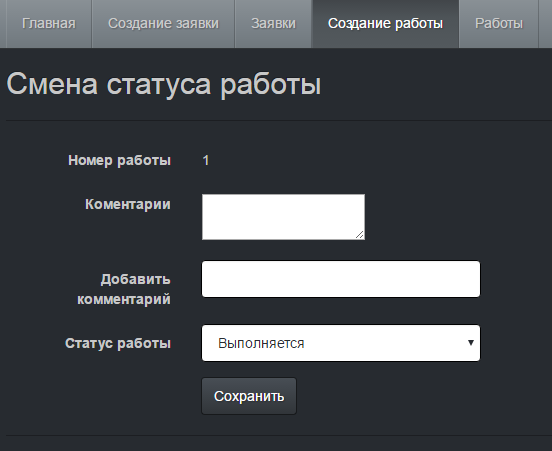


Рисунок 44. Вкладка «Смена статуса работы»

# **Список литературы**

1. Методические указания по выполнению курсового проекта по дисциплине «Методы и средства проектирования информационных систем и технологий» для бакалавров направление подготовки 09.03.02 «Информационные системы и технологии» профиль «Информационные системы и технологии». Составитель О.Н. Ванеев, Кемерово, 2017 г.;
2. <https://metanit.com/sharp/mvc.php> ;
3. <https://professorweb.ru/my/ASP_NET/base/level1/base_aspnet_index.php>