МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГБОУ ВО

«БРЯНСКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

**Кафедра** «Информатика и программное обеспечение»

**РАСЧЕТНО-ГРАФИЧЕСКАЯ РАБОТА**

по дисциплине

**«Конструирование программного обеспечения»**

**Тема:** АС “Отдел кадров”

Выполнил студент гр. О-18-ПРИ-рпс-Б

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Подгорняк А.А.

«\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2021

Руководитель

\_\_\_\_\_\_\_\_к.т.н., доц. Булатицкий Д.И.

«\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2021

Брянск 2021

**СОДЕРЖАНИЕ**

[Введение 4](#_Toc71878072)

[1. Пользовательские сценарии 5](#_Toc71878073)

[1.1. Добавление сотрудника в систему специалистом отдела кадров/ начальником отдела кадров 5](#_Toc71878074)

[1.2. Добавление сотрудника в систему специалистом отдела кадров/ начальником отдела кадров из “Найм” 5](#_Toc71878075)

[1.3. Редактирование напрямую специалистом отдела кадров/ начальником отдела кадров/инспектором по кадрам 6](#_Toc71878076)

[1.4. Редактирование через popup окно специалистом отдела кадров/ начальником отдела кадров/инспектором по кадрам 6](#_Toc71878077)

[1.5. Фильтрация по отдельным полям специалистом отдела кадров/ начальником отдела кадров/инспектором по кадрам 6](#_Toc71878078)

[1.6. Фильтрация по хранимым процедурам специалистом отдела кадров/ начальником отдела кадров/инспектором по кадрам 6](#_Toc71878079)

[1.7. Выбор сотрудников для повышения квалификации специалистом отдела кадров 7](#_Toc71878080)

[1.8. Назначение бонуса/штрафа сотруднику специалистом отдела кадров 7](#_Toc71878081)

[1.9. Контроль за рабочими часами сотрудника специалистом отдела кадров 7](#_Toc71878082)

[1.10. Управление расписанием сотрудника специалистом отдела кадров 7](#_Toc71878083)

[1.11. Управление отпусками сотрудника специалистом отдела кадров 8](#_Toc71878084)

[1.12. Создание справок специалистом отдела кадров 8](#_Toc71878085)

[1.13. Создание штатного расписания начальником отдела кадров 8](#_Toc71878086)

[1.14. Просмотр статистики по отделу начальником отдела кадров 8](#_Toc71878087)

[1.15. Управление кадрами в компании начальником отдела кадров 8](#_Toc71878088)

[1.16. Создание запроса на новых сотрудников начальником отдела кадров 9](#_Toc71878089)

[1.17. Добавление в список кандидатов инспектором по кадрам 9](#_Toc71878090)

[1.18. Найм новых сотрудников инспектором по кадрам 9](#_Toc71878091)

[2. Макеты интерфейса 10](#_Toc71878092)

[3. Прототипы интерфейса 12](#_Toc71878093)

[4. Разметка страницы 14](#_Toc71878094)

[5. Статическое представление 18](#_Toc71878095)

[6. Вывод из базы данных 19](#_Toc71878096)

[7. Изменение данных, полученных из базы 22](#_Toc71878097)

[8. Работа с системой управления проектами и задачами “Azure devops” 24](#_Toc71878098)

[9. Работа с дополнением “Resharper” в Microsoft Visual Studio 26](#_Toc71878099)

[10. Работа с системой контроля версий TFVC в Microsoft Visual Studio 27](#_Toc71878100)

[Заключение 29](#_Toc71878101)

[Список литературы 30](#_Toc71878102)

# Введение

Разработка качественного программного продукта не может обойтись без его полного конструирования, поскольку пренебрежение старательным выполнением этого этапа разработки может повлечь за собой ряд проблем на следующих её стадиях. Это крупная часть процесса разработки ПО, которая в зависимости от размера проекта занимает до 70% общего времени работы над ним. В процессе конструирования программного изделия осуществляются:

* функциональная декомпозиция решаемой задачи, на основе которой определяется архитектура системы, представляющей задачу;
* внешнее проектирование ПО, выражающееся в форме его внешнего взаимодействия с пользователем;
* проектирование базы данных;
* проектирование архитектуры программного обеспечения, т.е. определение множества объектов или модулей, функционально связанных с решаемой задачей, включая сопряжения между ними и требования к ним.

Курс дисциплины «Конструирование ПО» направлен на изучение приемов конструирования и знакомство с современными технологиями и инструментальными средствами сред быстрой разработки ПО. В связи с этим в рамках расчетно-графической работы была поставлена задача сконструировать веб-сайт со статьями, пройдя этапы написания пользовательских сценариев, создания макетов и приведения их к виду прототипов, разметки страниц сайта, создания веб-сайта на основе фреймворка *ASP.NET MVC* и привязки сайта к базе данных. Кроме того, было необходимо изучить дополнительные инструменты, необходимые для эффективного конструирования программного продукта.

# Пользовательские сценарии

Первым шагом разработки программного продукта является написание пользовательских сценариев. В данной расчетно-графической работе было составлено восемнадцать сценариев взаимодействия пользователей с системой. Было решено, что сайт должен поддерживать работу трёх видов пользователей: инспектор по кадрам, специалист отдела кадров и начальник отдела кадров. В начале каждого сценария была четко сформулирована цель, которую преследует пользователь. Далее были подробно описаны шаги, выполнение которых способствует её достижению. Тексты сценариев были подытожены результатами, полученными пользователями в ходе осуществления описанной последовательности действий. Ниже представлены полученные пользовательские сценарии.

## Добавление сотрудника в систему специалистом отдела кадров/ начальником отдела кадров

В разделе “Список персонала” выпадающего меню “Сотрудники” по кнопке “Добавить” открыть всплывающее окно для добавления. Далее нужно ввести данные нового работника в форму заполнения и подтвердить добавление, нажав кнопку “Подтвердить”. В результате в базу добавляется новый сотрудник, изменения в системе отражаются в таблице списка сотрудников.

## Добавление сотрудника в систему специалистом отдела кадров/ начальником отдела кадров из “Найм”

В разделе “Найм” выпадающего меню “Сотрудники” выбрать запись. По кнопке “Оформить” перейти в раздел “Список персонала” с открытым popup окном “Добавить”, и перенесенной в него имеющейся информацией. Затем ввести оставшиеся данные в форму заполнения и подтвердить добавление, нажав кнопку “Подтвердить”. Выбранная запись в разделе “Найм” переходит в статус “Принят”,для неё кнопка “Изменить/Оформить” становится неактивной. В результате в базу добавляется новый сотрудник, изменения в системе отражаются в таблице списка сотрудников.

## Редактирование напрямую специалистом отдела кадров/ начальником отдела кадров/инспектором по кадрам

Перейти в нужный раздел выпадающего меню “Сотрудники” и выбрать запись и его поле. Затем изменить данные и сохранить таблицу по кнопке “Сохранить”. В результате в базе обновляется информация о сотруднике, изменения в системе отражаются в таблице.

## Редактирование через popup окно специалистом отдела кадров/ начальником отдела кадров/инспектором по кадрам

Перейти в нужный раздел выпадающего меню “Сотрудники”. Выбрать запись, открыть всплывающее окно для редактирования по кнопке “Редактировать” и изменить данные. Подтвердить изменения, нажав кнопку “Подтвердить”. В результате в базе обновляется информация о сотруднике, изменения в системе отражаются в таблице.

## Фильтрация по отдельным полям специалистом отдела кадров/ начальником отдела кадров/инспектором по кадрам

Перейти в нужный раздел выпадающего меню “Сотрудники”. Выбрать поле для фильтрации в comobox слева от таблицы. Далее ввести искомые данные и запустить поиск по кнопке “Поиск”. А затем можно отменить поиск по кнопке “Отмена”. В результате в таблице выводятся только записи, соответствующие фильтру.

## Фильтрация по хранимым процедурам специалистом отдела кадров/ начальником отдела кадров/инспектором по кадрам

Перейти в нужный раздел выпадающего меню “Сотрудники”. Выбрать способ (хранимую процедуру) для сортировки и запустить сортировку по кнопке “Сортировка”. Затем можно отменить сортировку по кнопке “Отмена”. В результате в таблице выводятся только записи, соответствующие сортировке.

## Выбор сотрудников для повышения квалификации специалистом отдела кадров

Перейти в раздел “Квалификация” выпадающего меню “Сотрудники”. Далее нужно выбрать работников для обучения. Нажав кнопку “Обучение”, открыть popup окно и ввести назначение квалификации. Опционально можно распечатать приказ или создать файл со списком. Затем подтвердить действия по кнопке “Подтвердить”. В результате в поле “образование” добавляется новая квалификация, изменения в системе отражаются в таблице квалификации сотрудников.

## Назначение бонуса/штрафа сотруднику специалистом отдела кадров

Перейти в раздел “Бонусы/штрафы” выпадающего меню “Сотрудники”. Далее нужно выбрать сотрудника и, нажав кнопку “Добавить”, открыть popup окно, в котором можно ввести бонус/штраф и, если нужно, комментарий. Затем подтвердить действия по кнопке “Подтвердить”. Опционально можно сохранить копию таблицы в файл по кнопке “Экспорт”. В результате добавляются новые данные в поле “бонусы/штрафы” и “итог”, изменения в системе отражаются в таблице бонусов/штрафов сотрудников.

## Контроль за рабочими часами сотрудника специалистом отдела кадров

Перейти в раздел “Список персонала” выпадающего меню “Сотрудники”. Далее нужно выбрать сотрудника и открыть всплывающее окно по кнопке “Расписание”, где можно узнать отработанное кол-во часов за месяц

## Управление расписанием сотрудника специалистом отдела кадров

Перейти в раздел “Список персонала” выпадающего меню “Сотрудники”. Далее нужно выбрать сотрудника и открыть всплывающее окно по кнопке “Расписание”, где можно указать расписание работы (в виде недели). Затем подтвердить действия по кнопке “Подтвердить”. В результате расписание работника обновилось, изменения в системе отражены в таблице (в виде недели) в окне “Расписание”

## Управление отпусками сотрудника специалистом отдела кадров

Перейти в раздел “Список персонала” выпадающего меню “Сотрудники”. Далее нужно выбрать сотрудника и открыть всплывающее окно по кнопке “Расписание”, где можно указать отпуск (с… до…) работника. Опционально добавить комментарий. А затем подтвердить действия по кнопке “Подтвердить”. В результате обновилась информация об отпуске работника, изменения в системе кратко отражены в таблице список сотрудников и полностью в окне “Расписание”

## Создание справок специалистом отдела кадров

Перейти в раздел “Справки” выпадающего меню “Документация”. Далее необходимо выбрать один из шаблонов: копии или выписки из приказов о приеме, отпуске, переводе на другую должность, увольнении; справка с места работы с указанием информации о зарплате и других сведений; копия или выписку из трудовой книжки. Затем выбрать работника для автоматического заполнения и запустить печать по кнопке “Печать”.

## Создание штатного расписания начальником отдела кадров

Перейти в раздел “Штатное расписание” выпадающего меню “Документация”. Далее нужно заполнить документ согласно шаблону и сохранить в системе по кнопке “Сохранить”. Опционально можно сохранить в файл по кнопке “Экспорт” или распечатать по кнопке “Печать”. В результате штатное расписание обновилось/было создано, изменения в системе отражены в “Штатном расписании”.

## Просмотр статистики по отделу начальником отдела кадров

Перейти в раздел “Статистика” выпадающего меню “Отчёты” и выбрать отдел. В результате будет сформирован отчёт, объединяющий всю информацию по отделу

## Управление кадрами в компании начальником отдела кадров

Перейти в раздел “Управление персоналом” выпадающего меню “Сотрудники”. Выбрать отдел и затем сотрудника. Выбрать одно из решений (сократить/ перевести-отдел+должность/ повысить-новая должность) и подтвердить действия по кнопке “Подтвердить”. Автоматически будет распечатан приказ, заполненный по шаблону в соответствии с выбранными данными. В результате у выбранного сотрудника изменяется отдел/должность/статус, эти изменения в системе отражены в “Списке” сотрудников.

## Создание запроса на новых сотрудников начальником отдела кадров

Перейти в раздел “Найм” выпадающего меню “Сотрудники”. Далее по кнопке “Создать” открыть popup окно. В нем выбрать отдел и должность, а затем опубликовать заявку по кнопке “Опубликовать”. Состояние автоматически изменится на “в поиске”. В результате создается новое предложение для найма, информация о котором отражается в разделе “Найм”.

## Добавление в список кандидатов инспектором по кадрам

Перейти в раздел “Собеседование” выпадающего меню “Сотрудники”. Далее по кнопке “Добавить” открыть всплывающее окно для добавления. В нём ввести данные нового работника в форму заполнения, а затем подтвердить добавление по кнопке “Добавить”. В результате в список потенциальных сотрудников добавлена новая запись, изменения в системе отражены в таблице раздела “Собеседование”

## Найм новых сотрудников инспектором по кадрам

Перейти в раздел “Найм” выпадающего меню “Сотрудники”. Далее нужно выбрать запись в таблице и по кнопке “Изменить” открыть всплывающее окно для выбора сотрудника на должность. Изменить состояние запроса на “сотрудник прошел собеседование”. Выбрать человека из списка для “собеседования” (выводятся подходящие под должность). А затем подтвердить добавление, нажав кнопку “Сохранить”. Опционально можно по кнопке “Экспорт” сохранить копию таблицы в файл. В результате выбран человек на требуемую должность, изменения в системе отражены в таблице раздела “Найм”.

# Макеты интерфейса

Следующим шагом является визуализация описанных пользовательских сценариев. В ходе её выполнения формируется внешний вид страниц, необходимых для осуществления пользователями перечисленных действий. Так, словесное описание приобретает визуальную реализацию, а абстрактные элементы управления становятся конкретными. В ходе выполнения данного этапа расчетно-графической работы были получены основные страницы, необходимые для корректного функционирования сайта и покрытия пользовательских сценариев. Разработка проводилась в сервисе для построения макетов пользовательских интерфейсов *“Balsamiq Mockups”*.

Рассмотрим макеты, разработанные для покрытия сценария добавление сотрудника в систему начальником отдела кадров. В первую очередь была разработана начальная страница сайта, содержащая панели для быстрого доступа к разделам, а также выпадающие списки для навигации по сайту, которые повторяются на каждой странице.

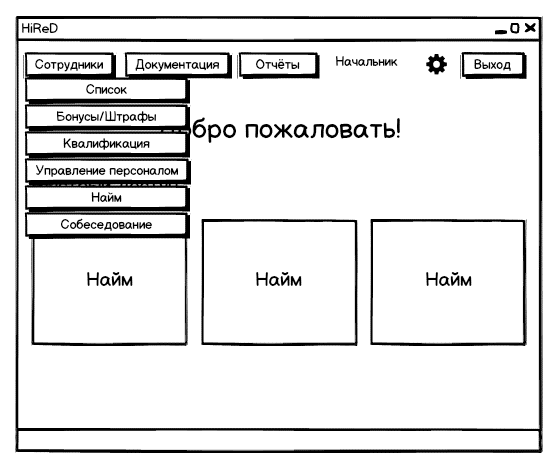


Рис. 1. Макет начальной страницы сайта

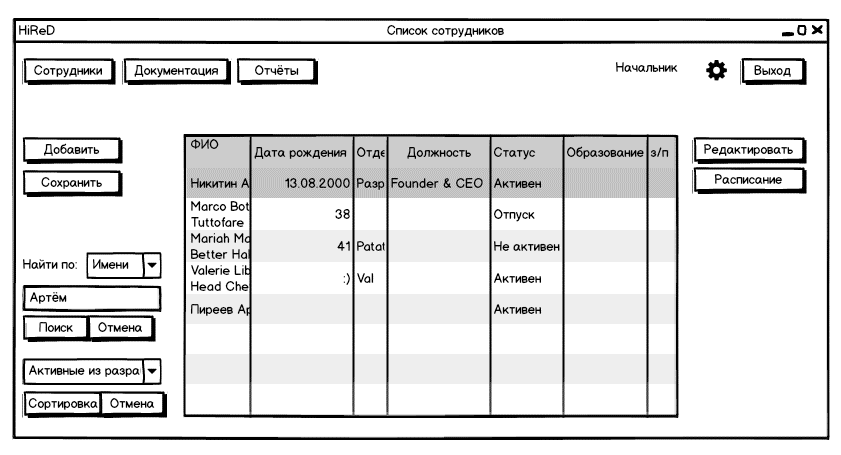
Перейдя на страницу со списком сотрудников, начальник отдела кадров имеет возможность добавить нового работника.

Рис. 2. Макет страницы со списком сотрудников

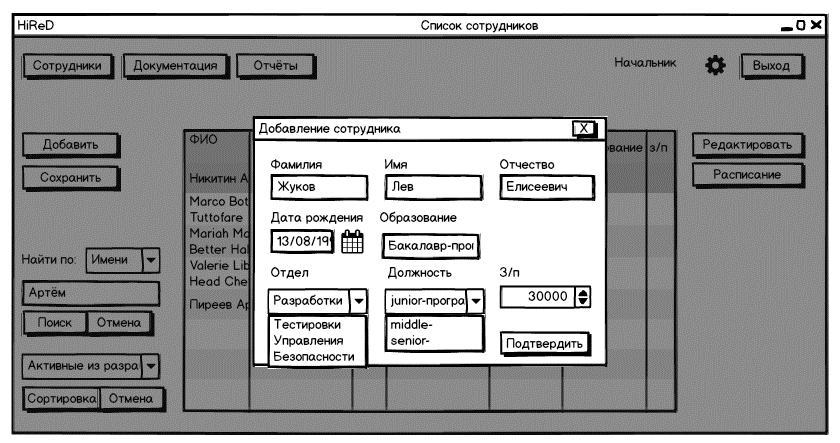
Целевая часть сценария – модальное окно добавления сотрудника – располагает полями для ввода персональных данных человека, а также данных о должности в компании. Отдел и должность выбираются из имеющихся через элементы combobox.

Рис. 3. Макет модального окна добавления сотрудника

Созданные макеты позволяют материализовать задумку разработчиков, но сделать это статически, то есть, не предусматривая имитацию действий, которые могут осуществляться пользователями.

# Прототипы интерфейса

На этапе разработки прототипа пользовательского интерфейса созданные ранее макеты улучшаются до прототипов, имитирующих какие-либо действия пользователя, а также устанавливающих связи между элементами управления и страницами. Это придает проекту динамики и дает возможность изобразить действия пользователя на примере предусмотренных разработчиком статических данных.

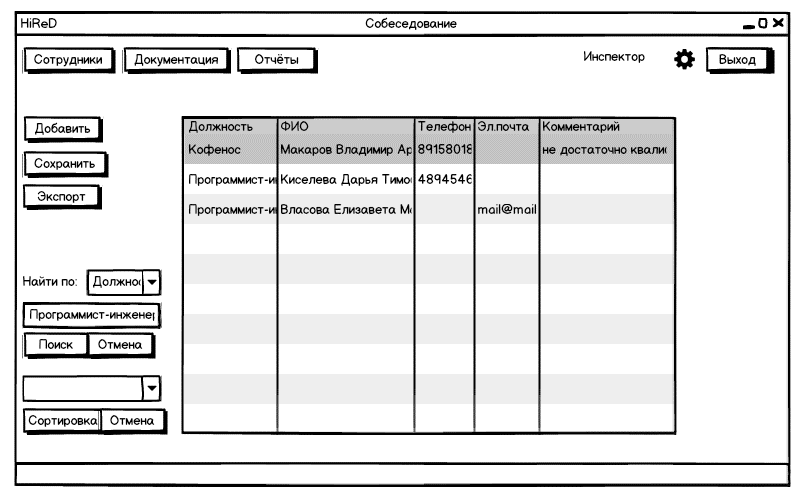
В качестве примера рассмотрим прототип, имитирующий добавление нового кандидата на работу инспектором по кадрам. Чтобы добавить человека, инспектору нужно перейти на страницу “Собеседование”. При нажатии на кнопку “Добавить” открывается модальное окно добавления нового потенциального сотрудника.

Рис. 4. Макет списка кандидатов

Когда основные поля будут заполнены, нужно нажать на кнопку «Добавить», после чего в списке появится новая запись.

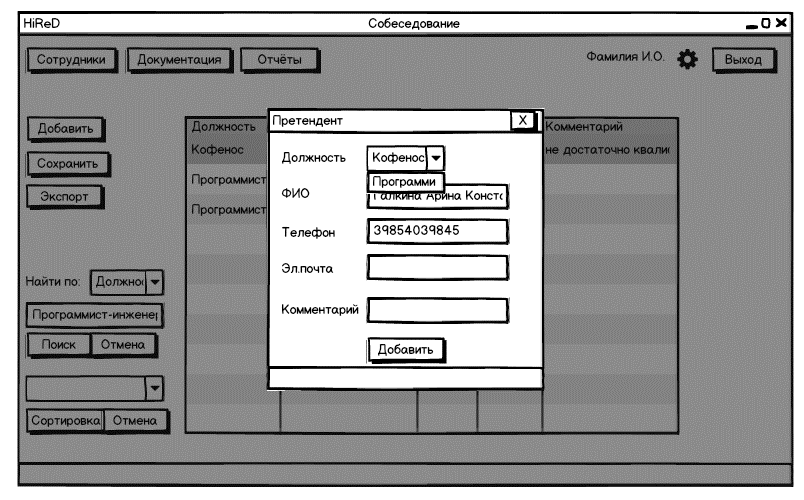
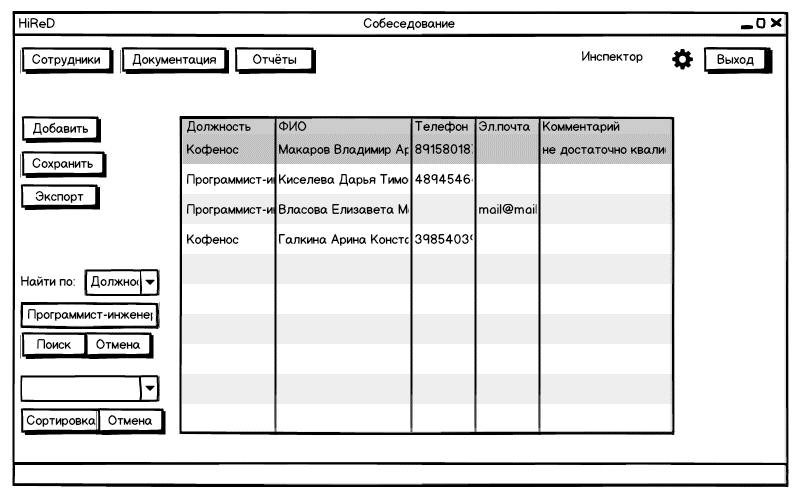


Рис. 5. Макет модального окна добавления кандидата

Рис. 6. Список кандидатов с новой записью

# Разметка страницы

После того, как были разработаны макеты и прототипы проекта сайта, следует перейти к разметке реальных страниц. Для упрощения этого процесса была использована библиотека *“Bootstrap”*. Возможности этой библиотеки позволяют генерировать html-разметку простым написанием html-кода практически без использования файлов стилей *css*.

Рассмотрим процесс создания разметки на примере страницы сайта для поиска новых сотрудников. Для того чтобы наметить её структуру, необходимо воспользоваться различными контейнерами. Для создания сетки сайта были использованы классы с готовыми отступами и размерами колонок. Для стилизации кнопок использовались специальные для этого bootstrap классы.

Внешний вид страницы, открытой в браузере, изображен на рис. 8.

Рис. 7. Страница “Найм”, размеченная с помощью “Bootstrap”

Также можно просмотреть созданную разметку в текстовом файле, с которым придется работать в последующие этапы разработки (**Ошибка! Источник ссылки не найден.**).

Листинг 1. Разметка страницы “Найм”

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <link href="~/Content/bootstrap.min.css" rel="stylesheet">  <meta name="viewport" content="width=device-width, initial-scale=1">  <link href="~/Content/solid.css" rel="stylesheet">  <link href="~/Content/solid.min.css" rel="stylesheet">  <link rel="stylesheet" href="~/Content/Style.css">  <link rel="stylesheet" href="~/Content/jquery.dataTables.min.css">  <script src="~/Scripts/jquery-3.6.0.min.js"></script>  <script src="~/Scripts/bootstrap.js"></script>  <script src="~/Scripts/bootstrap.min.js"></script>  <script src="~/Scripts/jquery.unobtrusive-ajax.js"></script>  <script src="~/Scripts/jquery.dataTables.min.js"></script>  <meta charset="UTF-8">  <title>HiReD</title>  </head>  <body>  <div class="All bg-dark " style="height: 100vh">  <div class="Header d-flex bg-success text-white ">  <div class="Logo mx-2 my-3 ">  <h2>HiReD</h2>  </div>  <div class="Navigation offset-md-5 my-3 ">  <h3>Найм</h3>  </div>  </div>  <div class="Sections ms-3 mt-2 ">  <button type="button" class="btn-sm btn-success ">Сотрудники</button>  <button type="button" class="btn-sm btn-success ">Документы</button>  <button type="button" class="btn-sm btn-success ">Отчёты</button>  <button type="button" class="btn-sm btn-success float-end mx-3">Выход</button>  <div class="Gear float-end mx-3 ">  <img src="~/Content/cog.png" alt="">  </div>  <a href="@Url.Action("InspectorView", "Inspector")" class="float-end mx-3 text-decoration-underline text-light">Начальник отдела кадров</a>  </div>  <div class="Content mt-5 mx-3 ">  <div class="Left">  <div class="BuTable bg-light text-center rounded">  <button type="button" class="btn-sm btn-success my-2 " id="Input" data-bs-toggle="modal" data-bs-target="#AllNew" onclick="fillAddModal()">Создать</button>  <br>  <button type="button" class="btn-sm btn-success mb-2 ">Экспорт</button>  </div>  <div class="Space">  </div>  <div class="Sort bg-white text-center rounded">  <label for="Find">Найти по:</label>  <select id="Find">  <option selected>Отдел</option>  </select>  <input class="mt-2" type="text" size="15">  <button type="button" class="btn-sm btn-success my-2 me-1">Поиск</button>  <button type="button" class="btn-sm btn-success my-2 ">Отмена</button>  <select id="Sort" class="mt-4">  <option selected></option>  </select>  <button type="button" class="btn-sm btn-success my-2 ">Сортировка</button>  <button type="button" class="btn-sm btn-success my-2 ">Отмена</button>  </div>  </div>  <div class="MainTable bg-white">  <table class="table table-hover cell-border table-bordered " id="MainTable">  <thead class="table-light">  <tr>  <th scope="col">№</th>  <th scope="col">Отдел</th>  <th scope="col">Должность</th>  <th scope="col">Статус</th>  <th scope="col">Информация</th>  </tr>  </thead>  <tbody id="MainTable-body">  @Html.Action("GetBD", "Recruitment")  </tbody>  </table>  </div>  <div class="Right">  <div class="BuNote bg-light col-1 text-center rounded">  <button type="button" class="btn-sm btn-success my-2 " id="Input" data-bs-toggle="modal" data-bs-target="#staticBackdrop" onclick="fillChangeModal()">Изменить</button>  <br>  <button type="button" class="btn-sm btn-success mb-2 ">Оформить</button>  </div>  </div>  </div>  <!-- Модальное окно изменения -->  <div class="modal fade" id="staticBackdrop" data-bs-backdrop="static" data-bs-keyboard="false" tabindex="-1" aria-labelledby="staticBackdropLabel" aria-hidden="true">  <div class="modal-dialog modal-dialog-centered">  <div class="modal-content">  <div class="modal-header bg-success">  <h5 class="modal-title text-white" id="staticBackdropLabel">Заявка</h5>  <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Закрыть"></button>  </div>  <div class="modal-body">  <table class="table recruit">  <tbody>  <tr>  <td><input class="mt-2" id="id" type="text" hidden></td>  </tr>  <tr>  <td><label for="Dep">Отдел:</label></td>  <td><input class="mt-2" id="Dep" type="text" disabled size="15"></td>  </tr>  <tr>  <td><label for="Post">Должность:</label></td>  <td><input class="mt-2" id="Post" type="text" disabled></td>  </tr>  <tr>  <td><label for="Status">Статус:</label></td>  <td>  <select id="Status">  <option value="В поиске">В поиске</option>  <option value="Прошел собеседование">Прошел собеседование</option>  <option value="Принят">Принят</option>  </select>  </td>  </tr>  <tr>  <td><label for="FIO">Выбор:</label></td>  <td>  <select id="FIO">  </select>  </td>  </tr>  </tbody>  </table>  </div>  <div class="modal-footer justify-content-center">    <button type="button" class="Save btn btn-success" onclick='Change()'>Сохранить</button>  </div>  </div>  </div>  </div>  <!-- Модальное окно добавления -->  <div id=”AllNew” class=”modal fade”>…</div>  </div>  <script src="~/Scripts/script.js"></script>  </body>  </html> |

# Статическое представление

Для того чтобы разработанные страницы были размещены на локальный сервер, необходимо перенести их в рамки проекта, построенного на фреймворке для создания веб-приложений *ASP.NET MVC*, который реализует шаблон *Model* (Модель) – *View* (Представление) – *Controller* (Контроллер).

На данном этапе ещё не требовалось создание модели. Достаточным было перенести текст разметки, полученной после предыдущих шагов, в файл, отвечающий за представление страницы. Контроллер, связанный с рассматриваемой страницей, на данный момент возвращает только её представление без каких-либо действий в соответствующей программе.

Такие ресурсы как скрипты, картинки и таблицы стилей были добавлены в соответствующие папки, которые находятся внутри проекта, и, соответственно, были изменены пути, ведущие к этим файлам, например *“ ~/Scripts/jquery.min.js”*. Также требовалось изменить ссылки, предназначенные для перехода с одной страницы на другую, на строки вида *“@Url.Action("Controller", "View")”*, где *“Controller”* – название контроллера, а *“View”* – название *View*.

# Вывод из базы данных

Перед тем, как осуществлять вывод из базы данных, необходимо её создать и перенести туда все те данные, которые отображались на странице в статическом представлении. Для самого вывода, прежде всего, необходимо подключить к проекту модель *“ADO.NET”*, установив подключение с базой, в результате чего для каждой таблицы подключенной базы автоматически будет создан класс, содержащий столбцы таблицы в качестве параметров.

Для того чтобы осуществить вывод списка заявок для поиска сотрудников, необходимо в контроллере получить контекст модели. Это необходимо для того чтобы в дальнейшем передать данные из целой таблицы в возвращаемое контроллером значение, обращаясь к полю контекста. После чего осуществляется цикл, в котором записываются необходимые данные в элемент модели, который производит последовательный вывод информации о статье в необходимый блок (листинг 2).

Листинг 2. Получение данных из БД

|  |
| --- |
| private HiReDEntities db = new HiReDEntities();  // GET: Recruitment  public ActionResult Index()  {  GetBD();  return View();  }  [HttpGet]  public ActionResult GetBD()  {  List<Complete> InterCruits = new List<Complete>();  var recruits = db.Recruitments.ToList();  foreach (var post in recruits)  {  Complete Entry = new Complete(post);  if (post.Interview != null)  if (post.Interview.PhoneNumber != null)  Entry.Info = post.Interview.FIO + " - " + post.Interview.PhoneNumber;  else  Entry.Info = post.Interview.FIO + " - " + post.Interview.Email;  else  Entry.Info = "";  InterCruits.Add(Entry);  }  ViewBag.Recruits = InterCruits;  return PartialView(InterCruits);  }  [HttpPost]  public ActionResult GetInterVIEW(string post,string name)  {  var men = PopulateView(post, name);  return PartialView(men);  }  private static List<Interview> PopulateView(string post, string name)  {      string serverName = "DESKTOP-S13P8DB";  string databaseName = "HiReD";    SqlConnectionStringBuilder sqlBuilder = new SqlConnectionStringBuilder();    sqlBuilder.DataSource = serverName;  sqlBuilder.InitialCatalog = databaseName;  sqlBuilder.IntegratedSecurity = true;    string providerString = sqlBuilder.ToString();  List<Interview> items = new List<Interview>();  using (SqlConnection con = new SqlConnection(providerString))  {  string query = " SELECT distinct Interview.FIO FROM Interview,Recruitment WHERE Interview.Post='"+post +  "' and Interview.id NOT IN ( SELECT Recruitment.Info FROM Recruitment WHERE Recruitment.Info IS NOT NULL )";  using (SqlCommand cmd = new SqlCommand(query))  {  cmd.Connection = con;  con.Open();  using (SqlDataReader sdr = cmd.ExecuteReader())  {  while (sdr.Read())  {  Interview tmp = new Interview();  tmp.FIO = sdr["FIO"].ToString();  items.Add(tmp);  }  }  con.Close();  }  if(name!=null)  {  Interview temp = new Interview();  temp.FIO = name;  items.Add(temp);  }  }  return items;  } |

Листинг 3. Заполнение модального окна изменения

|  |
| --- |
| function fillChangeModal()  {  var row = document.getElementsByClassName('table-active');  if (row[0].cells[3].innerText != "Принят") {  var id = document.getElementById('id');  id.value = row[0].cells[0].innerText;  var Department = document.getElementById('Dep');  Department.value = row[0].cells[1].innerText;  var Post = document.getElementById('Post');  Post.value = row[0].cells[2].innerText;  var Status = document.getElementById('Status');  Status.value = row[0].cells[3].innerText;  var SendPost = Post.value;  let FIO ;  if (row[0].cells[4].innerText != "") {  var arr = row[0].cells[4].innerText.split(' - ');  FIO = arr[0];  }  $('#FIO').load(url + "Recruitment/GetInterVIEW/", { post: SendPost, name: FIO });  }  else {  var Department = document.getElementById('Dep');  Department.value = row[0].cells[1].innerText;  var Post = document.getElementById('Post');  Post.value = row[0].cells[2].innerText;  var Status = document.getElementById('Status');  Status.value = row[0].cells[3].innerText;  let FIO = document.getElementById('FIO');  while (FIO.options.length > 0) {  FIO.options.remove(0);  }  if (row[0].cells[4].innerText != "") {  var arr = row[0].cells[4].innerText.split(' - ');  let newOption = new Option(arr[0], arr[0]);  FIO.append(newOption);  }  var btn = document.getElementsByClassName('Save');  btn[0].hidden = true;  }  } |

Листинг 4. Цикл вывода данных

|  |
| --- |
| @model IEnumerable<Complete>  @foreach (var recruit in Model)  {  <tr>  <td><p>@recruit.id</p></td>  <td><p>@recruit.Department</p></td>  <td><p>@recruit.Post</p></td>  <td><p>@recruit.Status</p></td>  <td><p>@recruit.Info</p></td>  </tr>  } |

# Изменение данных, полученных из базы

Для того чтобы можно было назвать разработанное приложение полноценным, необходимо реализовать возможность редактирования выгруженных из базы данных на страницах сайта.

Рассмотрим пример добавления новой заявки для поиска работника и изменение старой. После заполнения/добавления данных в модальное окно вызываются JS функции сбора и асинхронной отправки информации на сервер. Далее в контроллерах происходит обработка полученных данных и сохранение в БД.

Листинг 5. Отправка данных

|  |
| --- |
| function Change() {  let EntryArray = [];  var table = document.getElementsByClassName('recruit');  EntryArray.push(table[0].children[0].children[0].cells[0].children[0].value) //id  EntryArray.push(table[0].children[0].children[3].cells[1].children[0].value) //status  EntryArray.push(table[0].children[0].children[4].cells[1].children[0].value) //man choice  var request = new XMLHttpRequest();  request.open('POST', 'https://localhost:44387/Recruitment/EditRecuest');  request.setRequestHeader("Content-type", "application/x-www-form-urlencoded");  request.send('data=' + EntryArray);  $("#MainTable-body tr").remove();  $('#MainTable-body').load(url + "Recruitment/GetBD/");  document.location.reload(true);  }  function Add()  {  let EntryArray = [];  var table = document.getElementsByClassName('new-request');  EntryArray.push(table[0].children[0].children[0].cells[1].children[0].value) //status  EntryArray.push(table[0].children[0].children[1].cells[1].children[0].value) //man choice  var request = new XMLHttpRequest();  request.open('POST', 'https://localhost:44387/Recruitment/AddRecuest');  request.setRequestHeader("Content-type", "application/x-www-form-urlencoded");  request.send('data=' + EntryArray);  document.location.reload(true);  } |

Листинг 6. Обработка и сохранение данных

|  |
| --- |
| [HttpPost]  public ActionResult EditRecuest(string data)  {  string[] res = data.Split(new char[] { ',' });  string temp = res[2];  Interview interviewee = (db.Interviews.Where(a => a.FIO == temp).ToList())[0];  Recruitment request = db.Recruitments.Find(Int32.Parse(res[0]));  request.Status = res[1];  request.Info = interviewee.id;  request.Interview = interviewee;  db.Entry(request).State = EntityState.Modified;  db.SaveChanges();  return RedirectToAction("Index","Recruitment");  }  [HttpPost]  public ActionResult AddRecuest(string data)  {  string[] res = data.Split(new char[] { ',' });  Recruitment request = new Recruitment();  request.Department = res[0];  request.Post = res[1];  request.Status ="В поиске";    try  {  db.Entry(request).State = EntityState.Added;  db.SaveChanges();  }  catch (DbEntityValidationException ex)  {  foreach (DbEntityValidationResult validationError in ex.EntityValidationErrors)  {  Response.Write("Object: " + validationError.Entry.Entity.ToString());  Response.Write("");  foreach (DbValidationError err in validationError.ValidationErrors)  {  Response.Write(err.ErrorMessage + "");  }  }  }  return RedirectToAction("Index");  } |

# Работа с системой управления проектами и задачами “Azure devops”

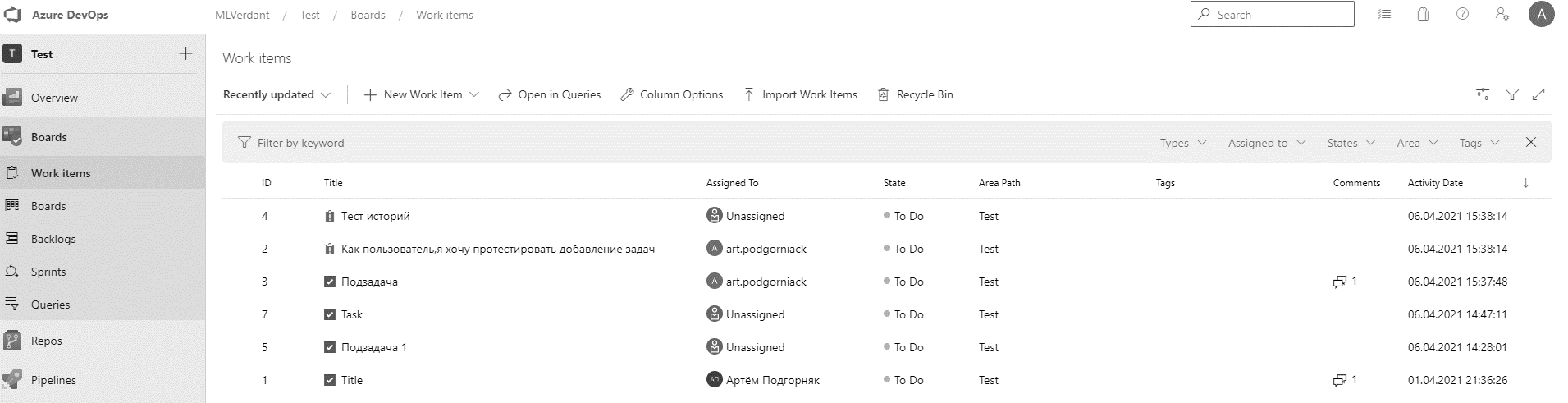
В рамках расчетно-графической работы также было необходимо ознакомиться с системой управления проектами и задачами на примере веб-интерфейса *“Azure DevOps”*. Работа с данным сервисом началась с создания проекта одного из предложенных типов и выбора методологии разработки (в данном случае предпочтение было отдано *Scrum* – методологии гибкой разработки ПО). После создания проекта предлагается создать новый спринт и добавить в него задачи, которые подлежат дальнейшему выполнению.

Рис. 8. Список задач

На вкладке *“Sprints”* можно запустить спринт, в результате чего будет необходимо обозначить сроки его завершения. После этого открывается панель проекта, содержащая следующие доски: *«To Do»*, *«Doing»*, *«Done»*. Здесь можно осуществлять изменения состояния выполнения проекта перетягиванием задачи из одной доски в другую. Каждой задаче присваивается приоритет её выполнения, а также исполнитель. Автор задачи записывается системой автоматически при её создании.

После того, как все поставленные задачи получают статус «Готова», спринт можно завершить преждевременно.

Помимо движения задачи вперед по этапам разработки, также можно отправить её на доработку, перетянув её из доски *«Done»* на доску *«Doing»*. Таким образом, с помощью *Azure DevOps* можно осуществлять эффективный процесс разработки проекта, отслеживая готовность его подзадач.

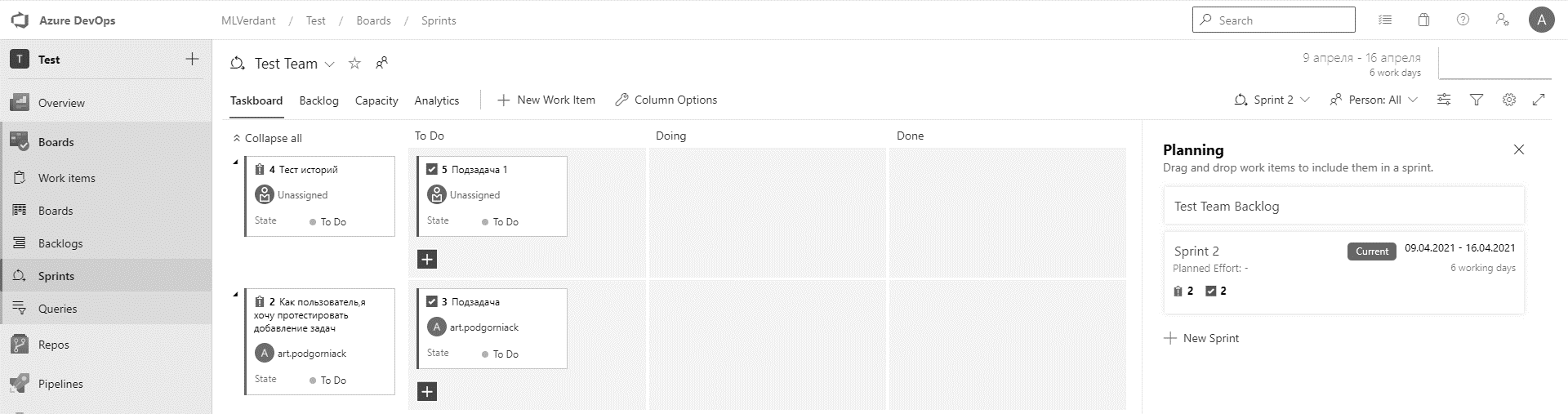


Рис. 9. Активный спринт

# Работа с дополнением “Resharper” в Microsoft Visual Studio

При разработке программного обеспечения программисты зачастую допускают ошибки оформления кода, и эти ошибки кажутся им несущественными, поскольку они не являются синтаксическими. Тем не менее, код необходимо писать в едином стиле, чтобы он выглядел чистым и был понятен другому разработчику. С этой целью при выполнении расчетно-графической работы в среду *“Microsoft Visual Studio”* было установлено дополнение *“ReSharper”*.

Данная система предлагает возможности статического анализа качества кода и исправления выявленных ошибок, причем изменение элементов доступно на уровне всего проекта. Кроме того, существуют средства рефакторинга, такие как, например, выделение из повторяющегося участка кода отдельного метода. Таким образом, над разрабатываемым проектом были произведены процедуры создания методов, переименования переменных с целью удовлетворения правилам написания кода и другие средства преобразования кода.

# Работа с системой контроля версий TFVC в Microsoft Visual Studio

С целью изучения процесса работы с системой контроля версий в среде разработки *“Microsoft Visual Studio”* было использовано расширение *Azure DevOps*, поддерживающее работу с системой *Team Foundation Version Control.* Для работы с ресурсом необходима регистрация, далее – вход в систему из среды *Microsoft VS*. На сервере *Azure* был создан репозиторий, с которым будет происходить взаимодействие. Это можно делать как из веб-интерфейса, так и из самой *VS*. Для добавления текущего проекта в систему контроля версий в обозревателе *“Team Explorer”* в контекстном меню был выбран пункт добавления в *TFVC*. Главная панель *“Team Explorer”* содержит различные элементы, например, панель «Ожидающие изменения», позволяющая фиксировать изменения локально, так и отправлять их на сервер с комментариями.

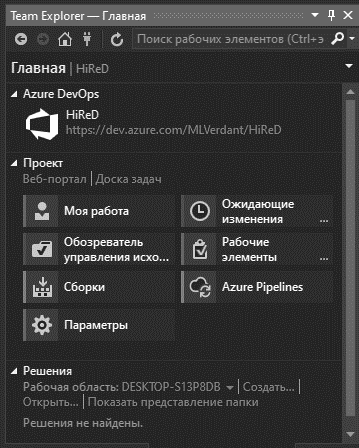


Рис. 10. Главная страница панели Team Explorer

Также рассмотренное окно содержит панель, позволяющую управлять ветвями проекта, а именно: создавать ветви, назначать ветвь главной, вносить изменения для определенной ветви, сливать несколько ветвей. В данной панели было выполнены перечисленные действия, а также вызваны конфликты слияния, которые решаются либо слиянием изменений обеих ветвей, либо предпочтением изменений одной ветви другой.

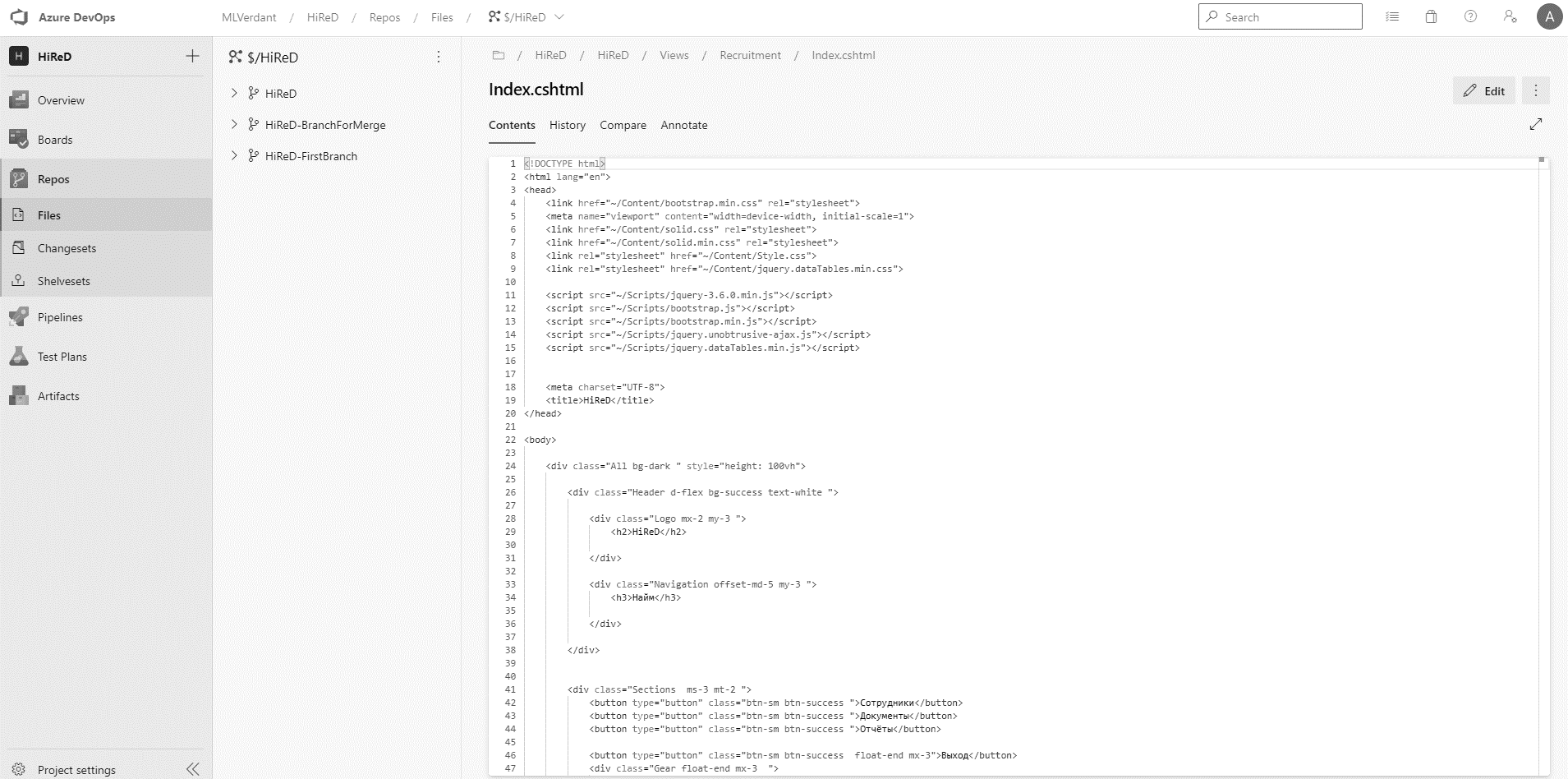
Веб-интерфейс *Azure DevOps* выглядит следующим образом. Здесь доступны как файлы проекта, так и информация о ветвях и закрепленных изменениях.

Рис. 11. Веб-интерфейс Azure DevOps

# Заключение

Результатом выполненной расчетно-графической работы является веб-сайт, поддерживающий три типа пользователей: инспектор по кадрам, специалист отдела кадров и начальник отдела кадров. Последовательность разработки была следующей: было разработаны восемнадцать пользовательских сценариев, на их основании были реализованы макеты и прототипы системы в программе “*Balsamiq* *Mockups* *3*”, на основании их размечены страницы с помощью фреймворка “*Bootstrap*”. Полученные страницы были перенесены в проект ASP.NET MVC, где можно реализовать связь сайта с базой данных с возможностью вывода из неё данных и их редактирования. Последующие выполненные действия связаны с освоением процесса конструирования. А именно: проведен рефакторинг кода с использованием инструмента “*ReSharper*”, организована совместная работа над проектом с помощью *“Azure DevOps”* и системы контроля версий *“TFVC”*.

Выполнение данной работы позволило ознакомиться с последовательным выполнением действий в процессе конструирования ПО.

# Список литературы

1. Официальный сайт Azure Devops. URL: https://dev.azure.com/.
2. Официальный сайт Balsamiq Mockups. URL: https://balsamiq.com/.
3. Официальный сайт ReSharper. URL: https://www.jetbrains.com/resharper/.
4. Официальный сайт Bootstrap. URL: http://getbootstrap.com/.
5. Сайт о программировании METANIT.COM. URL: https://metanit.com/.
6. Фримен, А. ASP.NET MVC 3 Framework с примерами на C# для профессионалов / А. Фримен, С. Сандерсон. – М.: Вильямс, 2012. – 672 с.