# Лабораторная работа №1. Основы HTML и CSS. Верстка и оформление сайта.

Пререквизиты: Visual Studio 2017 Community Edition версии 15.3 (скачать можно отсюда <https://www.visualstudio.com/downloads/>, при установке нужно отметить модули «ASP.NET and web development», «Azure development» и «.NET Core cross-platform development»)

1. Создать проект ASP.NET Core (с аутентификацией или без).
2. Оформить выбранный доклад [https://1drv.ms/x/s!Ag5pMwumdpilhMoDii4Vxn9Ze\_8sMQ](https://vk.com/away.php?to=https%3A%2F%2F1drv.ms%2Fx%2Fs%21Ag5pMwumdpilhMoDii4Vxn9Ze_8sMQ&cc_key=) в HTML. Разделы доклада можно оформить как в виде блоков одной страницы, так и на нескольких страницах.
3. Каждый доклад предполагает наличие примеров. Примеры можно представить на самом сайте, либо реализовать в одном из сервисов <https://codepen.io/>, <https://jsbin.com/>, <https://jsfiddle.net/>.
4. Если примеры предполагают использование javascript, то javascript-код требуется вынести в отдельные файлы, а в HTML загружать javascript-код с использованием инструкции <script src=’{url-javascript-файла}’></script>.
5. Оформление элементов страницы должно строиться на CSS-классах, чтобы избежать дублирование css-правил на однотипных элементах страницы. Весь css-код требуется вынести в отдельные файлы, а в HTML загружать css-код с использованием инструкции <link rel=’stylesheet’ href=’{url-css-файла}’>.
6. Сайт одновременно должен эстетично выглядеть и быть информативным.

Примеры, как не стоит оформлять сайты:

* <http://www.dokimos.org/ajff/>
* <http://www.spaceistheplace.ca/>
* <http://www.ronoslund.com/>
* <http://www.theworldsworstwebsiteever.com/>

Советы по популярному в настоящее время плоскому-дизайну: <http://tilda.education/articles-flat-design>.

1. В случае использования сторонних примеров на них обязательно нужно ссылаться.

# Лабораторная работа №2. Отправка данных форм на сервер. Контактная форма.

В этой ЛР требуется добавить на сайт из ЛР№1 страницу с контактной формой и реализовать отправку сообщений на email.

1. Создать контроллер ContactController.
2. Добавить в контроллер действие Index. Действие не принимает параметров и должно возвращать представление с контактной формой.
3. Контактная форма должна содержать кнопку отправки сообщения на сервер и три текстовых поля:
   1. email для обратной связи;
   2. тема сообщения;
   3. текст сообщения.
4. По нажатию кнопки отправки данные формы должны отправляться на сервер методом POST в действие Send контроллера ContactController.
5. Форма должна поддерживать валидацию полей. Валидацию требуется реализовать c использованием атрибутов модели <https://docs.microsoft.com/en-us/aspnet/core/mvc/models/validation> с проверкой состояния модели в действии контроллера. Правила валидации:

* email должен быть валидным;
* тема сообщения не должна превышать 50 символов;
* текст сообщения не должен превышать 2000 символов;
* поле email обязательно к заполнению;
* поле ‘текст сообщения’ обязательно к заполнению.

В случае, если модель не валидна, сообщение не должно отправляться, а на странице должны быть отображены ошибки валидации.

1. Если модель валидна, то сообщение должно отправляться на email администратора сайта. Настройки отправки требуется вынести в файл конфигурации, например, в appsettings.json. Типичный набор настроек для отправки email-сообщения:

* Адрес smtp-сервера (например, smtp.gmail.com)
* Порт smtp-сервера (например, 587)
* Логин
* Пароль
* Email получателя

Для отправки email можно использовать любую библиотеку, например: <https://github.com/jstedfast/MailKit>, <https://github.com/lukencode/FluentEmail> или любую иную.

# Лабораторная работа №3. Игра “Жизнь”

В этой лабораторной работе требуется написать симулятор игры ‘Жизнь” (<http://kvant.mccme.ru/1974/09/igra_zhizn.htm>, <https://ru.wikipedia.org/wiki/%D0%96%D0%B8%D0%B7%D0%BD%D1%8C_(%D0%B8%D0%B3%D1%80%D0%B0)>) на языке javascript с визуализацией в Canvas <https://www.w3.org/TR/2dcontext/>.

Краткое описание правил игры:

* Для симуляции игры используется клеточное поле
* У каждой клетки есть 8 соседних клеток
* Каждая клетка может находится в одном из двух состояний – либо пустая, либо в ней находится фишка.
* Фишка выживает (клетка остается в том же состоянии) в следующем поколении если у неё есть две или три соседние фишки.
* Фишка погибает (клетка опустошается) в следующем поколении, если она имеет больше чем три (от перенаселенности) или меньше, чем два соседа (от одиночества).
* Фишка рождается в пустой клетке, если в соседних с ней клетках стоит ровно три фишки.

# Лабораторная работа №4. Web-api сервисы. JSON.

В текущей лабораторной необходимо написать программу (консольное приложение), которая бы выводила на экран следующую информацию:

1. Названия профессий в вакансиях, объявленная зарплата которых привышает либо равна 120000 руб.
2. Названия ключивых навыков в вакансиях, объявленная зарплата которых привышает либо равна 120000 руб.
3. Названия профессий в вакансиях, объявленная зарплата которых менее 15000 руб.
4. Названия ключивых навыков в вакансиях, объявленная зарплата которых менее 15000 руб.

Для получения данных о вакансиях предлагается использовать Web-api сервис предоставляемый проектом HeadHunter <https://api.hh.ru>.

В случае, если указан диапазон зарплаты (от... до...), нужно брать среднее значение диапазона. Если в диапазоне указано только одно значение (от... или до...) нужно брать его. Если зарплата не указана, вакансию не нужно учитывать.

Обратите внимание:

О том, как можно получать список вакансий используя HeadHunter api (описание тут <https://github.com/hhru/api/blob/master/docs/vacancies.md>)

Необходимо получить вакансии со всех страниц, возвращаемых сервисом.

Для работы с форматом JSON можно использовать библиотеку JSON.NET (<http://www.newtonsoft.com/json>, можно найти в NuGet).

Данные о запрлате (поле salary) и ключевых навыках (поле key\_skills) имеются не во всех вакансиях.

Зарплата в вакансиях может быть указана не в рублях, эти значения тоже нужно учитывать.

Авторизация в веб-сервисе не требуется.

Фильтрация списка на стороне веб-сервиса не требуется.

Получить данные в формате json из сервиса можно при помощи класса HttpClient (<https://msdn.microsoft.com/en-us/library/system.net.http.httpclient(v=vs.118).aspx>)

В параметрах запроса и в получаемом от сервиса ответе есть информация о пейджинге (можно посмотреть в описании api). Её необходимо использовать чтобы загрузить все страницы, предоставляемые сервисом.

Максимальное количество вакансий возвращаемых сервисом - 2000 (на нескольких страницах).

Максимальное количество вакансий на одной странице возвращаемой сервисом - 500.

Данные о ключевых навыках (поле key\_skills) возвращается в детализированной информации о вакансии.

По желанию можно выбрать другой сервис, отличный от HeadHunter, представляющий данные о вакансиях в JSON или XML форматах.