Практическая работа № 3

РАЗРАБОТКА ГРАФИЧЕСКОГО ИНТЕРФЕЙСА ПРОГРАММНОГО СРЕДСТВА

Цель работы: проектирование графического интерфейса компьютерного тренажера.

Методические указания

Дизайн пользовательского интерфейса является фактором, оказывающим влияние на три основных показателя качества программного продукта: его функциональность, эстетику и производительность.

Функциональность является фактором, на который разработчики приложений зачастую обращают основное внимание. Они пытаются создавать программы так, чтобы пользователи могли выполнять свои задачи и им было удобно это делать. Функциональность важна, но, тем не менее, это не единственный показатель, который должен учитываться в ходе разработки.

Эстетичный внешний вид самого приложения и способа его представления (вплоть до упаковки) позволяет сформировать у потребителя положительное мнение о программе. Однако эстетические характеристики весьма субъективны и описать их количественно горздо труднее, чем функциональные требования или показатели производительности. Вся эстетика приложения зачастую сводится к простому выбору: соотносятся ли между собой используемые цвета, передают ли элементы интерфейса их назначение и смысл представляемых операций, что ощущает человек при использовании тех или иных элементов управления и насколько успешно он их использует.

Производительность, а равно и надежность, также влияют на перспективу применения программы. Если приложение хорошо выглядит, имеет простое и удобное управление, но, к примеру, медленно прорисовывает экраны, регулярно «подвисает» на десяток-другой секунд или, того хуже, падает с критической ошибкой при некорректных действиях

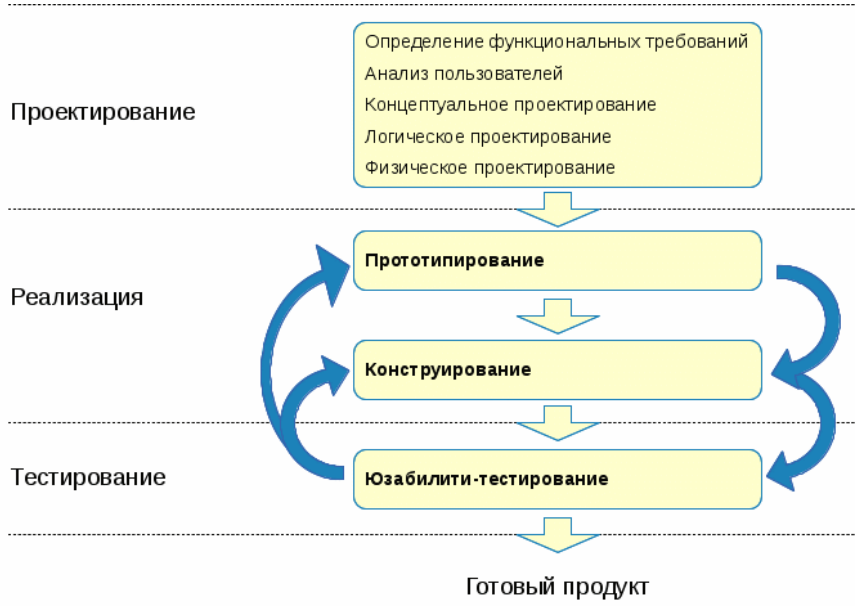
пользователя, у него, вероятно, будет мало шансов на длительную эксплуатацию. В свою очередь, быстрая и стабильная работа приложения могут отчасти компенсировать его не самый стильный дизайн или отсутствие каких-то вторичных функций.

Разработка интерфейса как процесс

Для обеспечения успешной работы пользователя от дизайнера интерфейса требуется соблюдать баланс между вышеперечисленными факторами на протяжении всего жизненного цикла разработки приложения. Это достигается последовательной и тщательной проработкой деталей интерактивного взаимодействия на каждом из *этапов разработки пользовательского интерфейса* (рис. 1), включающих:

1. **Проектирование**
    - Функциональные требования: определение цели разработки и исходных требований.
    - Анализ пользователей: определение потребностей пользователей, разработка сценариев, оценка соответствия сценариев ожиданиям пользователей.
    - Концептуальное проектирование: моделирование процесса, для которого разрабатывается приложение.
    - Логическое проектирование: определение информационных потоков в приложении.
    - Физическое проектирование: выбор платформы, на которой будет реализован проект и средств разработки.
2. **Реализация**
    - Прототипирование: разработка бумажных и/или интерактивных макетов экранных форм.
    - Конструирование: создание приложения с учетом возможности изменения его дизайна.
3. **Тестирование**

- Юзабилити-тестирование: тестирование приложения различными пользователями, в т.ч. и пользователями с ограниченными возможностями (Accessibility testing).

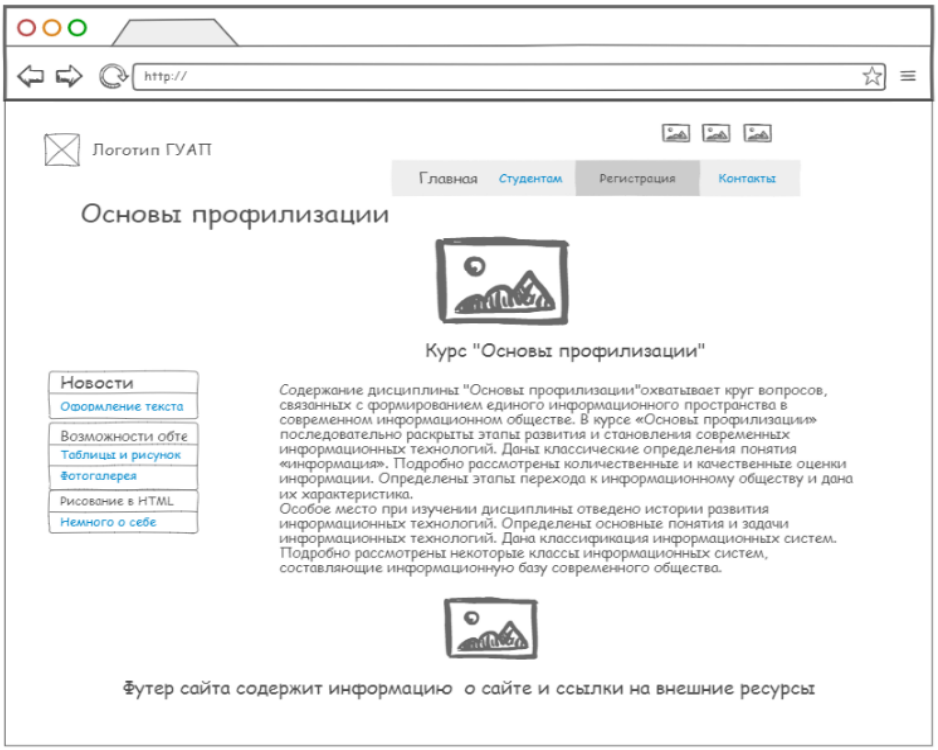


Как и разработка приложения в целом, создание пользовательского интерфейса для него — процесс итеративный. Маловероятно, что такие этапы, как прототипирование, конструирование и тестирование интерфейса могут быть завершены за один проход. Поэтому, если в результате юзабилити-тестирования выявлены недоработки, то они, если это возможно, устраняются путем повторного конструирования, либо разрабатывается новый прототип интерфейса.

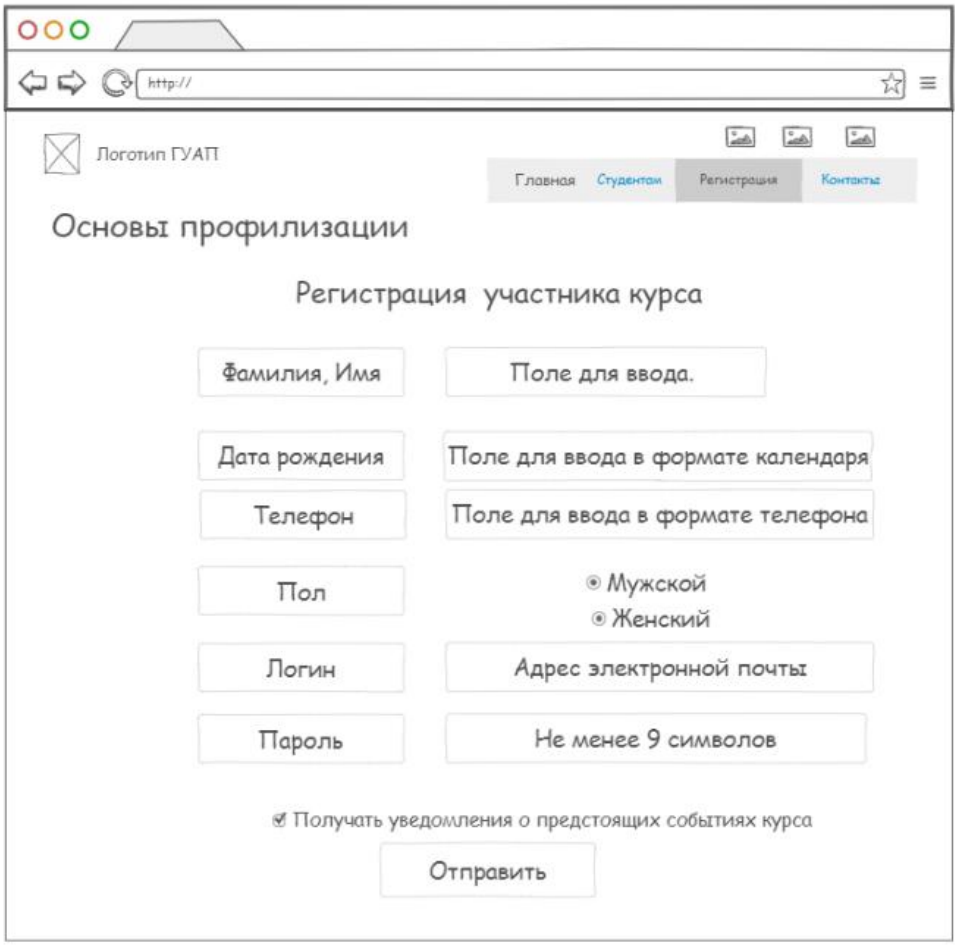
Внешняя структура веб-приложения показывает расположение видимых пользователем блоков интернет-проекта (рис. 5.15).

Для создания прототипа сайта можно использовать специальные программы, например, специализированные онлайн-сервисы Draw.io (https://www.draw.io) или Ninjamock (https://ninjamock.com). На рис.5.19-5.20 представлены прототипы *Главной страницы* и страницы *Регистрация*.

Следует заметить, что прототипирование страниц веб-ресурса можно выполнять в программах для обработки текстовой информации, в частности, Microsoft Word, в графических редакторах, например, Paint или в специализированных программах, таких как Microsoft Visio.



Прототип Главной страницы



Прототип страницы Регистрация

Порядок выполнения работы

1. Выбрать предметную область, в которой будет разработано программное средство – компьютерный тренажер. Программные технологии, которые будут использованы для реализации тренажера, выбираете самостоятельно в зависимости от вида программного продукта: стационарного приложения или веб-приложения. Использование программных сред Power Point, Excel и других приложений, инструментами которых можно реализовать тренажер запрещено. При реализации используем языки программирования, фреймворки и др.

Студенты заочной формы обучения могут согласовать тему и технологии реализации (если не умеют программировать) с преподавателем по электронной почте. Уровень исполнения и сложности программы должен соответствовать уровню магистратуры по направлению «Прикладная информатика».

2. После обдумывания замысла тренажера следует реализовать его интерфейс. Первый этап – это проектирование (прототипирование) интерфейса в любой известной вам программе. Разработайте Главную страницу сайта, общий вид страниц второго и последующих уровней вложения.

Второй этап – его программная реализация. В отчете по этой практической работе выкладываем отчет по разработке только интерфейса: скриншот (ы) графического интерфейса и фрагменты программного код, который его создает.

Содержание отчета.

1. Цель работы.
2. Описание выбранного сервиса для прототипирования сайта (любого, Figma, Draw.io, Word, Paint, Photoshop и др.).
3. Прототипы (макеты, фреймы) не менее пяти страниц будущего сайта.
4. Выводы о проделанной работе с описанием проблем, которые возникли в процессе выполнения практической работы, и путей их решения.
5. Список актуальных использованных источников.

Приложение

Пример выполнения № 1

Ход работы:

Разработку графического интерфейса мы разделили на 3 части, в соответствии с методическими указаниями:

1) Прототипирование
2) Разработка
3) Тестирование

Первый этап, прототипирование, выполняла Дарья Шаталова.

Перед проектирование любого продукта должно проводиться исследование, с помощью которого сформируется общее представлении о продукте. Проведя анализ сайтов, таких как https://findplace.club/ и https://online.seterra.com/ близких по функционалу к нашему приложению и выделила следующие пункты:

1. Пользователь хочет видеть сайт с простым и минималистичным дизайном;
2. Цвета не должны быть яркими и броскими, приветствуются приглушенные тона фиолетового, зеленного цвета;
3. Функциональность системы должна быть осуществлена строго в рамках разработанного плана;

Изучив на просторе интернета средства для создания графического интерфейса, мое внимание привлек сервис proto.io. Это достаточно простой и понятный инструмент для проектирования каркасов веб-продуктов любой сложности.

С результатами можно ознакомиться на рисунках 1 -5.

Запуская приложение пользователю предоставляется на выбор три теста разной сложности.

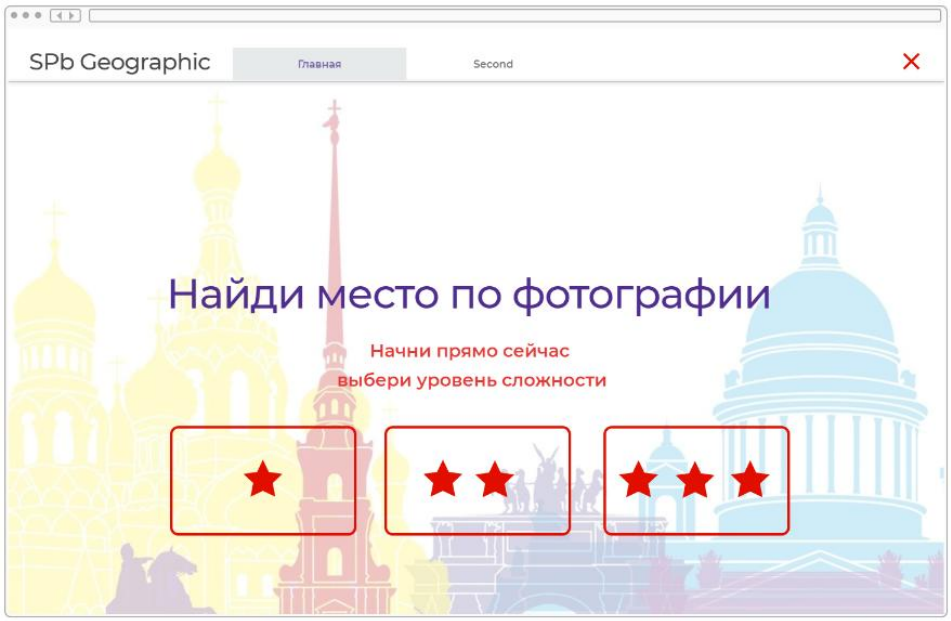


Рисунок 1 – Главная страница

Первый тест – уровень сложности Easy, состоит из 12 вопросов, где пользователь должен угадать по картинке достопримечательность, ему необходимо выбрать правильный ответ из 4 предоставленных вариантов. После ответа, он может перейти к следующему вопросу.

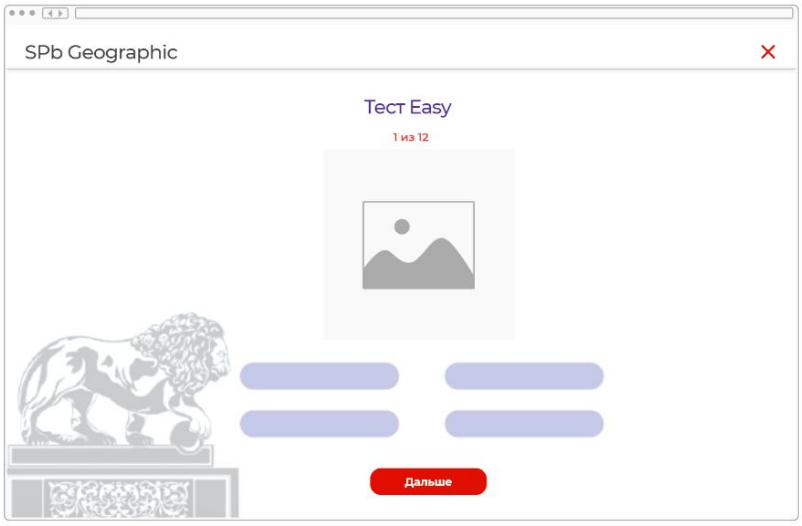


Рисунок 2 - Тест Easy

Второй тест – уровень сложности Medium, состоит из 12 вопросов. Теперь пользователю нужно вписать правильный ответ и после приступить к следующему вопросу.

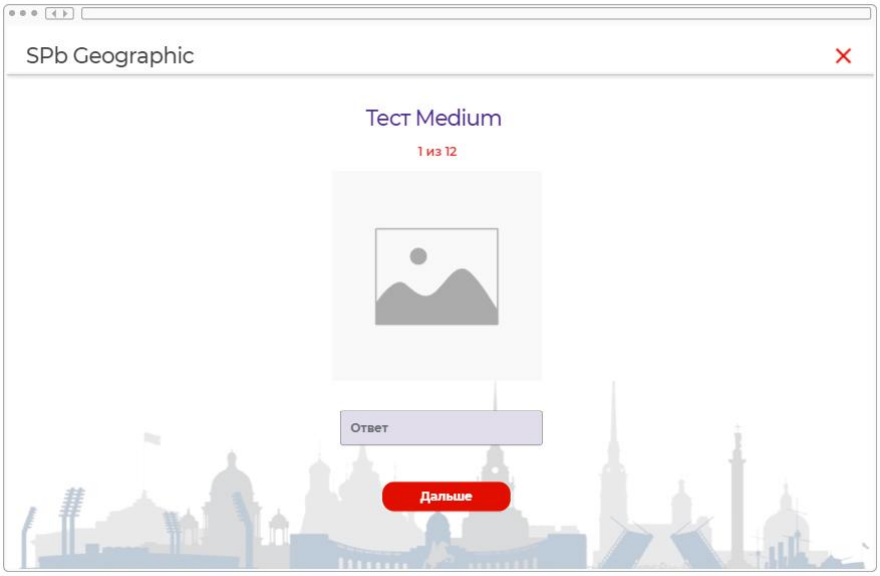


Рисунок 3 - Тест Medium

Третий тест – уровень сложности Hard, состоит из 12 вопросов. Показано изображение достопримечательности и 4 варианта его вида со спутника. Пользователь, основываясь на своих визуальных и территориальных знаниях должен выбрать правильный ответ и прейти к следующему вопросу.

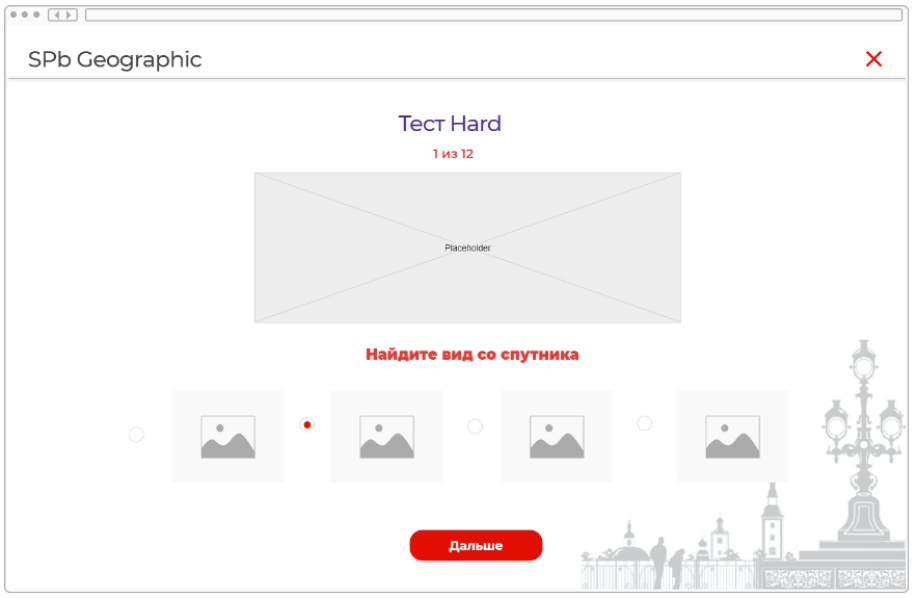


Рисунок 4 - Тест Hard

В конце теста пользователю показ его результат в процентном соотношении. Помимо этого, ему предоставляется возможность пройти тест заново или вернуться на главную страницу и выбрать другой тест.

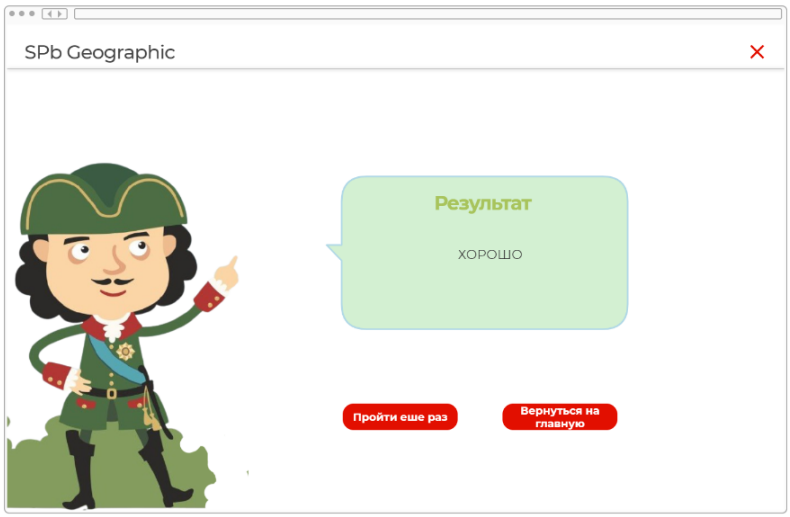


Рисунок 5 - Результат

Второй этап, реализацию графического интерфейса, на себя взяли Никита Калимов и Александр Гловацкий. Наш тренажер – это веб-приложение. В качестве фреймворка используется ASP.NET Core как лучший фреймворк в экосистеме .NET для реализации веб—приложений.

В своей верстке и реализации интерфейса мы использовали:

- Razor-страницы: представления, которые сочетают в себе классический фронтенд-стек (HTML, CSS, JavaScript) и C#. Позволяют динамически отрисовывать страницы, используя конструкции C#;
- Чистый HTML, CSS, JavaScript для верстки, стилизации и программирования поведения страниц соответственно;
- Фронтенд-фреймворк Bootstrap для ускорения разработки интерфейса – он содержит в себе стилизацию и утилиты, с помощью которых можно реализовать типовые компоненты например карточки.

Мы реализовали три основных страницы-компонента:

1) Главная
2) Страница теста
3) Страница с результатами

Примеры страниц:

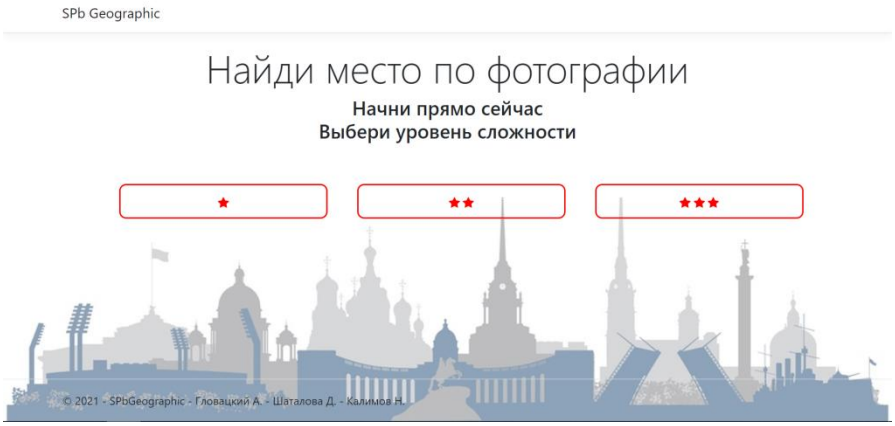


Рисунок 6 – главная страница

Программный код разбит на две части: Layout(макет) – то, что общее для всех страниц в приложении; например, навигационная панель и футер; и само тело страницы.

Программный код макета:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>SPb Geographic</title>
    <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
    <link rel="stylesheet" href="~/css/site.css" />
</head>
<body>
    <header>
        <nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light bg-white
border-bottom box-shadow mb-3">
            <div class="container">
                <a class="text-left navbar-brand" asp-area="" asp-controller="Home" asp-
action="Index">SPb Geographic</a>
            </div>
        </nav>
```

```
    </header>
    <div class="container">
        <main role="main" class="pb-3">
            @RenderBody()
        </main>
    </div>

    <footer class="border-top footer">
        <div class="container">
            &copy; 2021 - SPbGeographic - Гловацкий А. - Шаталова Д. - Калимов Н.
        </div>
    </footer>
    <script src="~/lib/jquery/dist/jquery.min.js"></script>
    <script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
    <script src="~/js/site.js" asp-append-version="true"></script>
    @await RenderSectionAsync("Scripts", required: false)
</body>
</html>
```

Атрибуты HTML-тегов с приставкой “asp” – это вспомогательные атрибуты от выбранного фреймворка веб-приложения ASP.NET Core. Они помогают генерировать HTML-код, основываясь на метаданных.

Программный код главной страницы:

```
<style>
    body {
        background: 100% 100% url(/images/home-background-2.jpg) no-repeat;
        background-size: 100%;
    }
</style>

<div class="text-center">
    <h1 class="display-4">Найди место по фотографии</h1>
    <h3>Начни прямо сейчас <br />
    Выбери уровень сложности</h3>
    <ul class="tests-list">
        <li class="nav-item">
            <a class="nav-link text-dark" asp-area="" asp-controller="Test" asp-
action="Easy">
                <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16"
fill="#FF0000" class="bi bi-star-fill" viewBox="0 0 16 16">
                    <path d="M3.612 15.443c-.386.198-.824-.149-.746-.592l.83-4.73L.173
6.765c-.329-.314-.158-.888.283-.95l4.898-.696L7.538.792c.197-.39.73-.39.927 0l2.184 4.327
4.898.696c.441.062.612.636.282.95l-3.522 3.356.83 4.73c.078.443-.36.79-.746.592L8
13.187l-4.389 2.256z" />
                </svg>
            </a>
        </li>
```

И т.д. (ограничение наложено преподавателем, в отчете представляем код полностью)

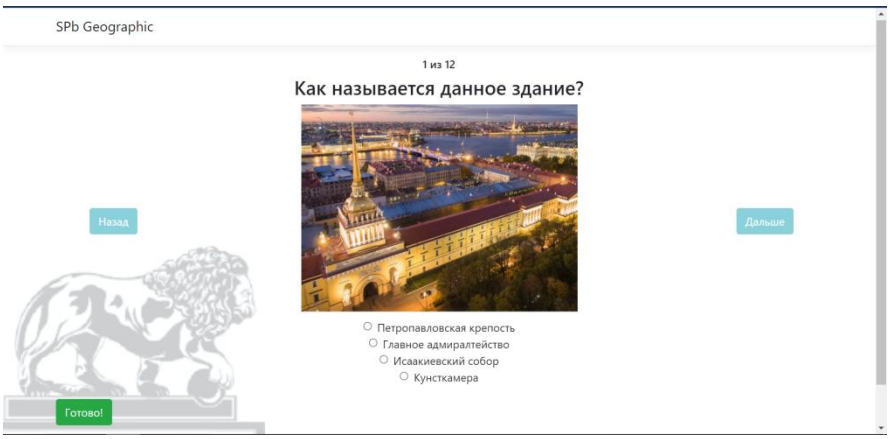


Рисунок 7 – страница теста «Легкий»

Программный код страницы теста «Легкий»:

```
@model IList<WebApplication.Models.Easy.EasyQuestionViewModel>
<style>
    body {
        background: 0 100% url(/images/lion.png) no-repeat;
        background-size: 30%;
    }
</style>
<div id="test-carousel" class="carousel slide text-center" data-ride="false">
    <form enctype="multipart/form-data" asp-controller="Result" asp-action="Easy"
method="post" class="carousel-inner">
        @for (int i = 0; i < Model.Count; i++)
        {
            string carousel_item_class = i == 0 ? "carousel-item active" : "carousel-
item";
        <div class="@carousel_item_class">
            <div class="question">
                <h6>@(i + 1) из @(Model.Count)</h6>
                <h3 name="[@i].Question">@Model[i].Question </h3>
                <div class="question-image">
                    <img height="300" src="@Model[i].ImageBase64" alt="Картинка"
class="scale" />
                </div>
                <div class="select-answer">
                    @foreach (var answer in Model[i].Options)
                    {
                        <div class="form-check">
                            <label class="form-check-label">
                                <input class="form-check-input"
name="[@i].SelectedAnswer" type="radio" value="@answer" />
                                @answer
                            </label> <br />
                        </div>
                    }
```

 и т.д. (ограничение наложено преподавателем)

Пример № 2

Ход выполнения работы:

1. Тренажёр выполнен в форме подготавливающих тестов, которые в случае неправильного ответа выводят подсказку.
2. Разработка интерфейса:

На главной странице можно выбрать тест из списка возможных (рисунок 1), так как список тестов загружается динамически, пользователь будет проинформирован с помощью loader (рисунок 2). Loader использован во всех местах, где требуется загрузка информации с сервера.

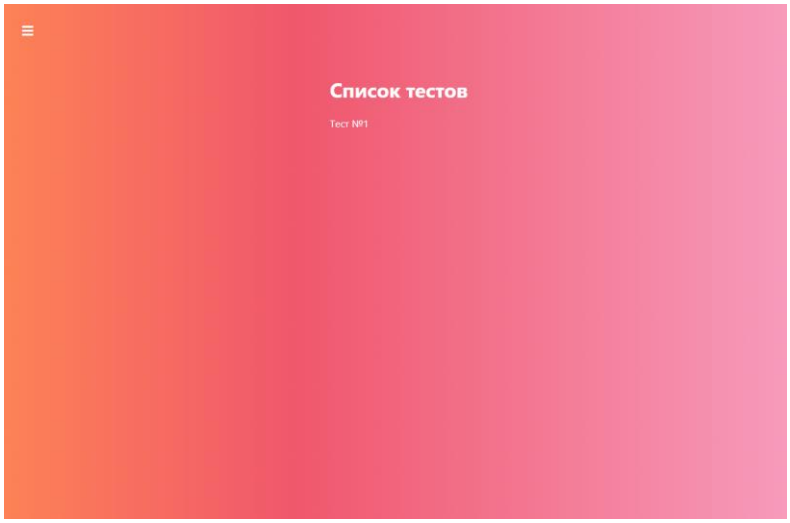


Рисунок 1 – Главная страница

Рисунок 2 – loader

После выбора теста пользоваетеля перебрасывает на страницу с вопросами (рисунок 3)

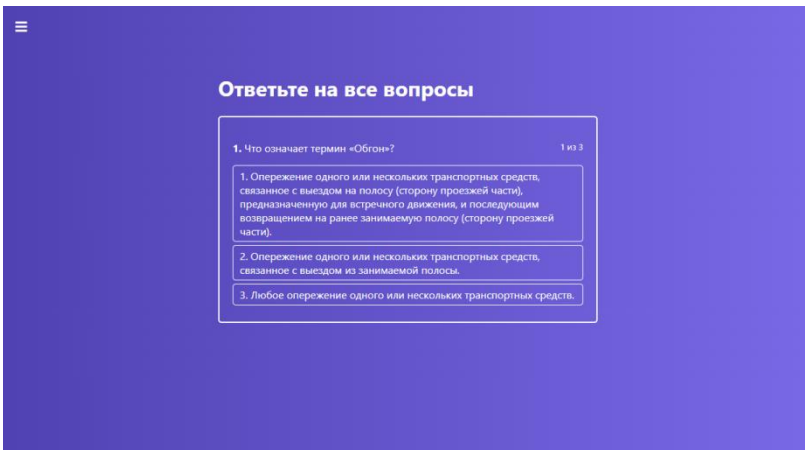


Рисунок 3 – страница вопросов

В слуачае неправильного ответа тренажёр проинформирует пользователя о неудаче при помощи красной подсветки выбранного варианта ответа, также пользователь получает посказку под списком выриантов ответов (рисунок 4)

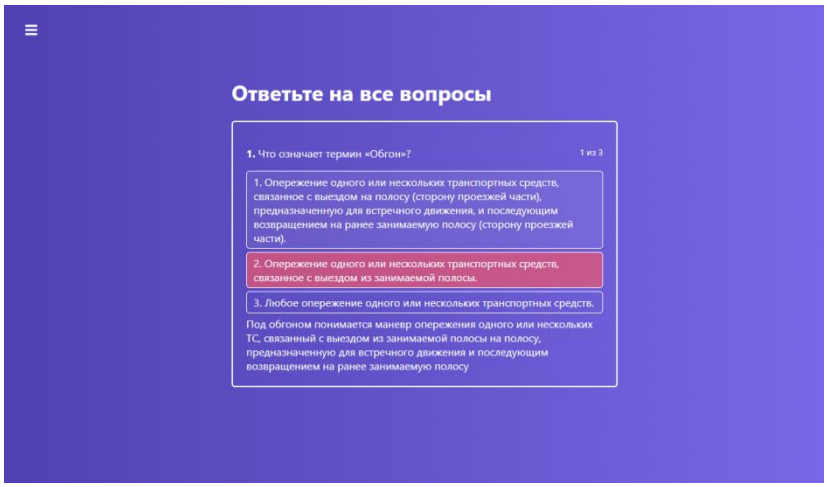


Рисунок 4 – поведение тренажёра при неправильном ответе

Если пользователь правильно ответил на вопрос, то выбранный вариант ответа подсвечиается зелёным цветом 2 секунды, после чего переходит к следующему вопросу (рисунок 5).

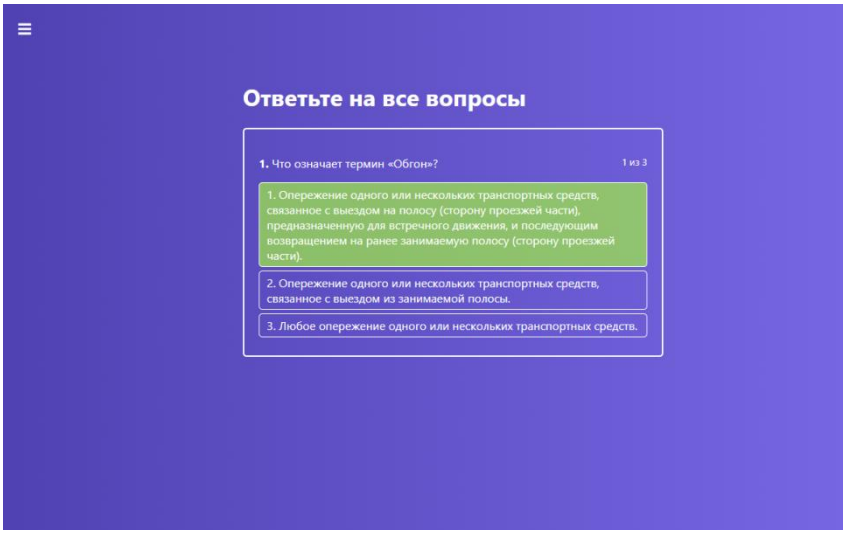


Рисунок 5 - поведение тренажёра в случае правильного ответа

В тренажёре возможно выводить картинки, если вопрос того требует (рисунок6).

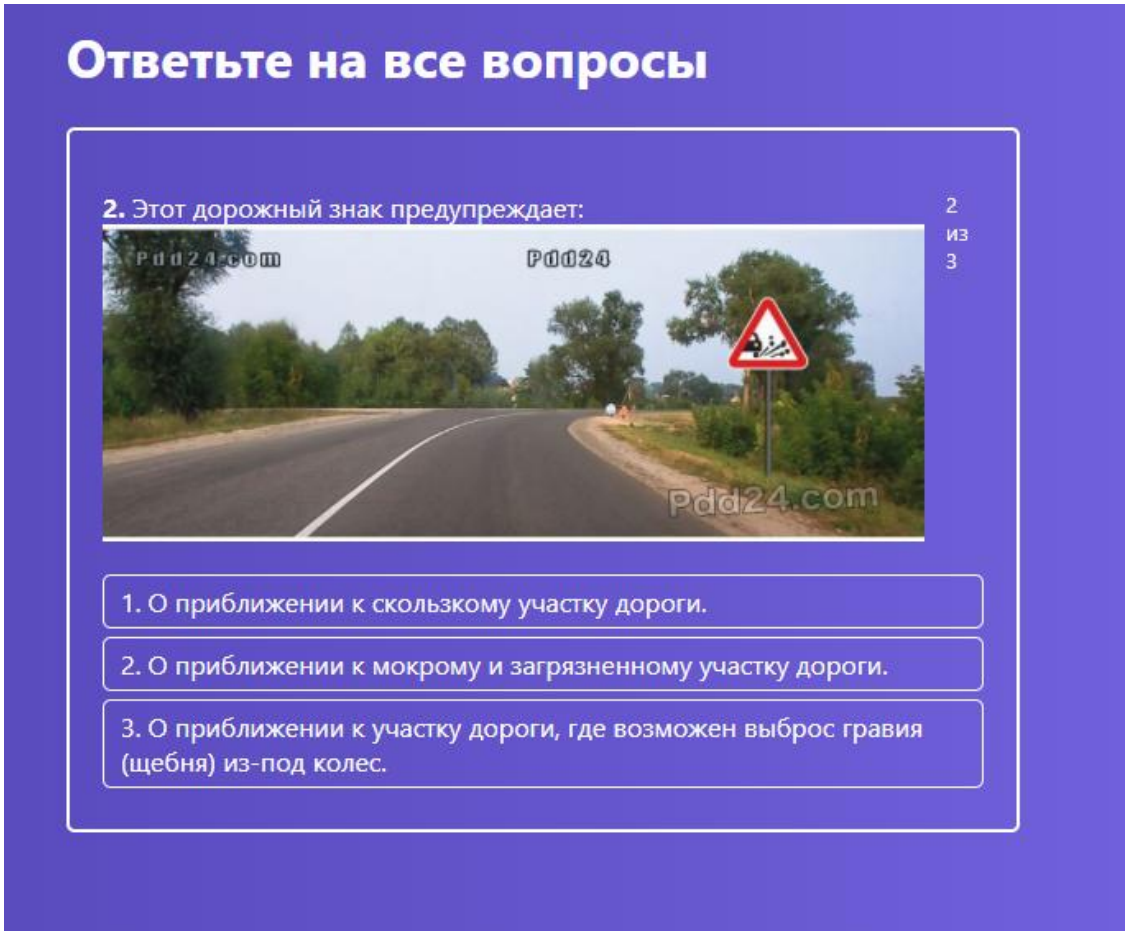


Рисунок 6 – вопрос с картинкой

После проохождения теста, тренажёр выводит список вопросов с правильными и ошибочными ответами, таким образом пользоваетль сможет понять, какие темы требуют дополнительного внимания (рисунок 7).

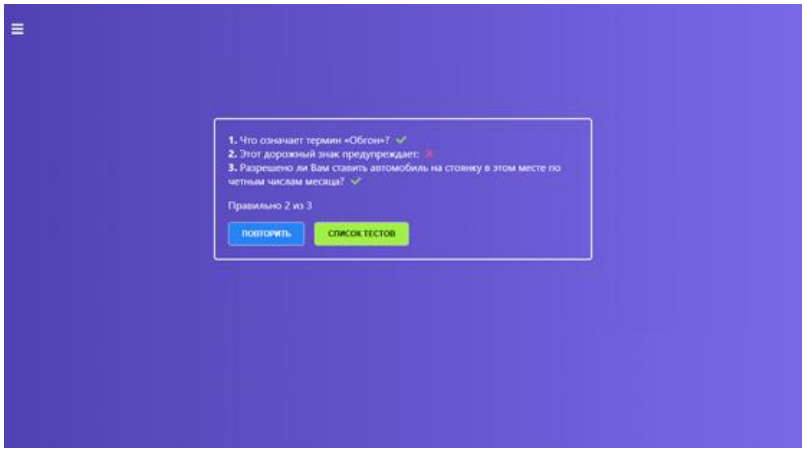


Рисунок 7 – результат прохождения теста

В тренажёре возможна авторизация. Авторизованные пользователи могут создавать новые тесты. Для авторизации необходимо выбрать в меню пункт «Авторизация» (рисунок 8) и на странице авторизации ввести свой email и пароль (рисунок 9). Если

пользователь ввёл невалидные данные, приложение уведомит пользоваетеля об этом (рисунок 10).

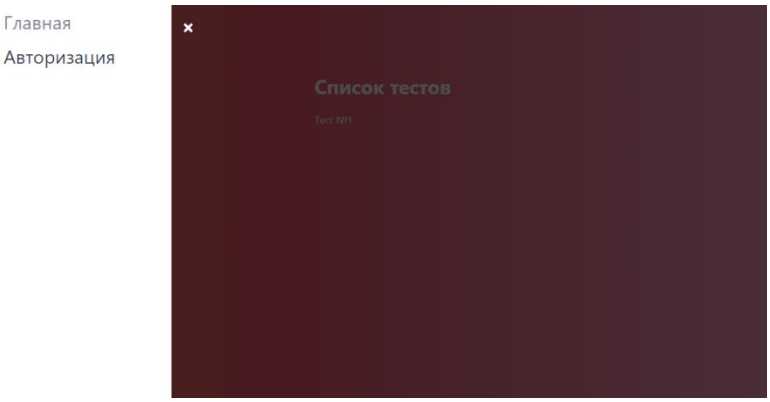


Рисунок 8 – меню тренажёра

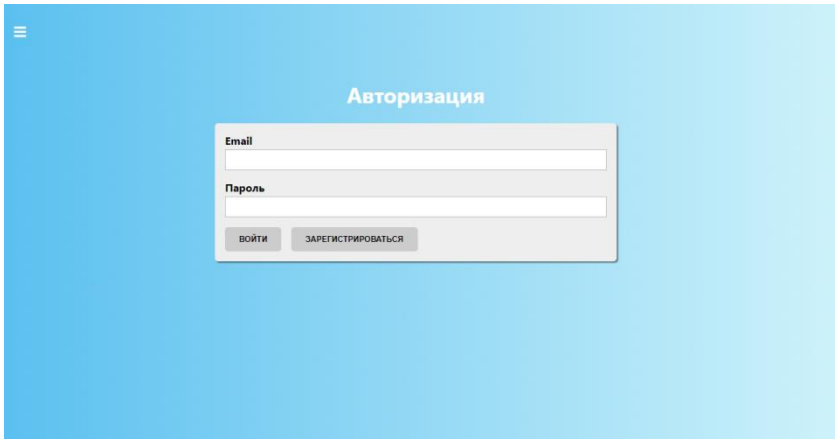
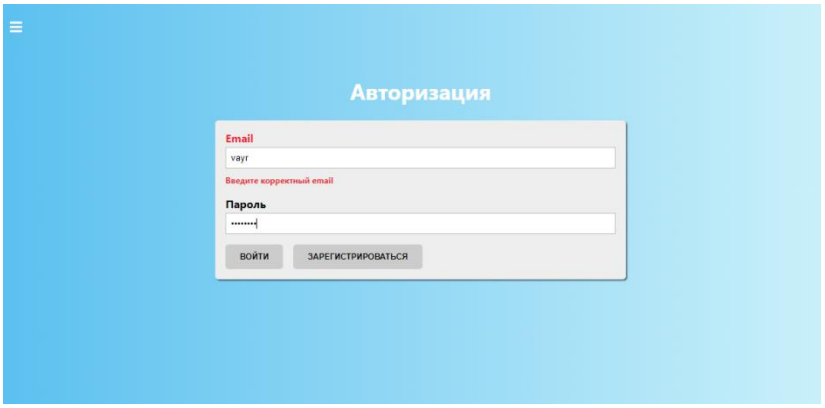


Рисунок 9 – страница авторизации



Риснок 10 – сообщение о некорректно введённом email

После прохождения авторизации, в приложении повяляются новые пункты меню «Создать тест» и «Выйти» (Рисунок 11). Теперь пользователь может попасть на страницу создания нового теста (Рисунок 12)

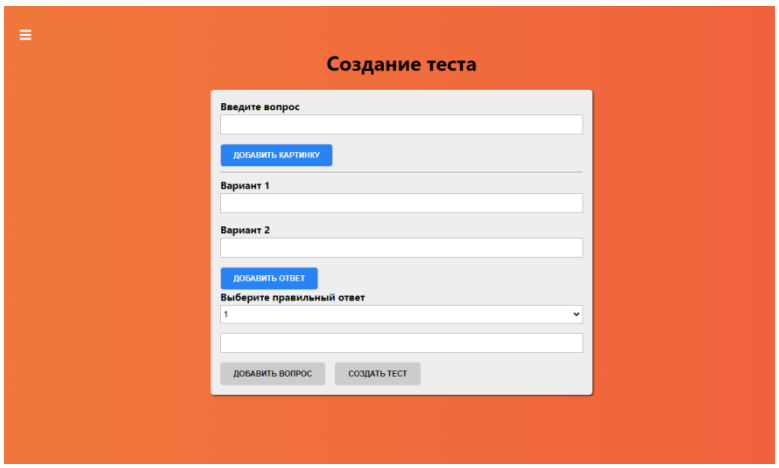
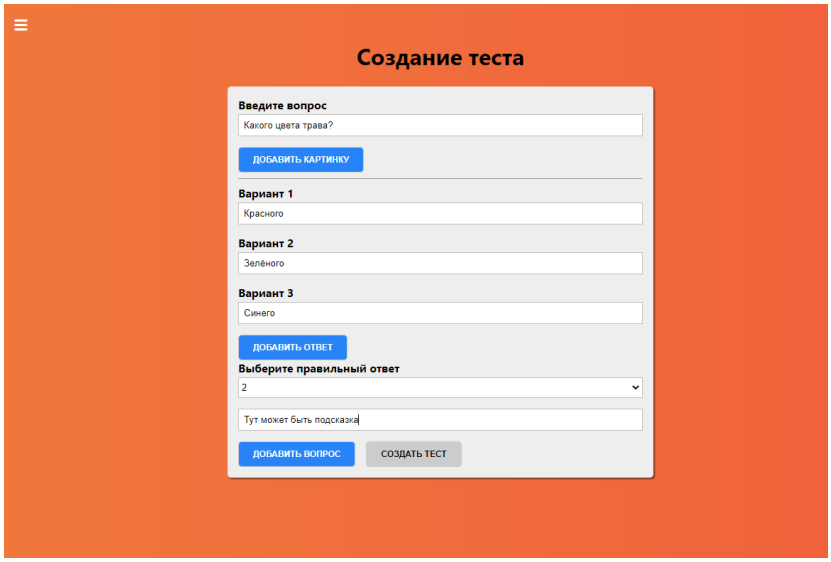


Рисунок 12 – страница создания теста

Здесь пользователь может добавить до 8 вариантов ответов, картинку и обязательно указать правильный вариант ответа и подсказку.



Вывод: в ходе выполнения лабораторной работы был реализован интерфейс тренажёра.

Пример № 3

1. Проектирование

- Функциональные требования: интерфейс должен представлять из себя пошаговый виджет обучения пользователя методу сьемки кругового монокулярного видео человека с последующим запуском камеры (фронтальной либо основной).
- Анализ пользователей: добавить уровень прогресса прохождения обучения, сценарий использования – пошаговый полноформатный проход.
- Концептуальное проектирование: в основе концепта заложена возможность пользователя в домашних условиях получить свою 3д модель с параметрическими характеристиками тела.
- Логическое проектирование: см. рис. 1

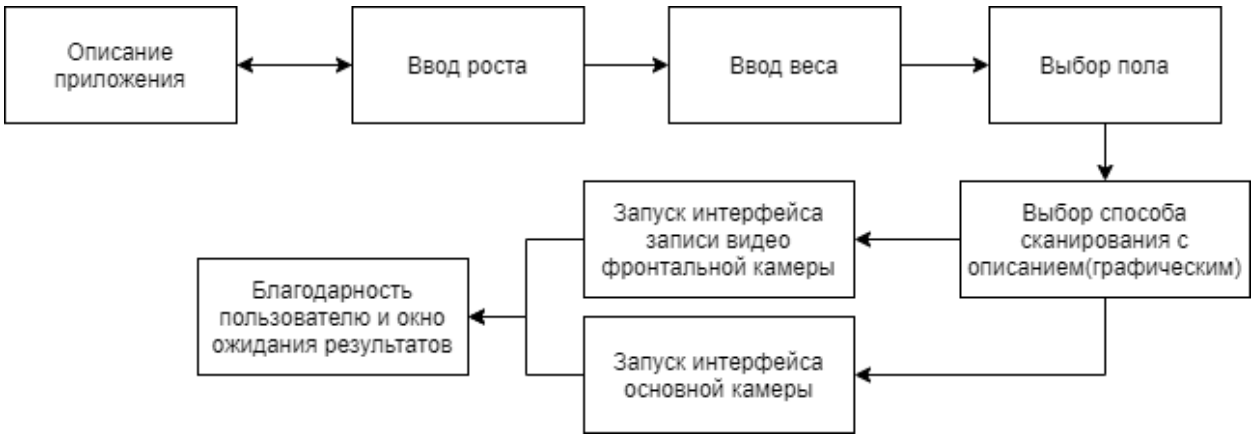
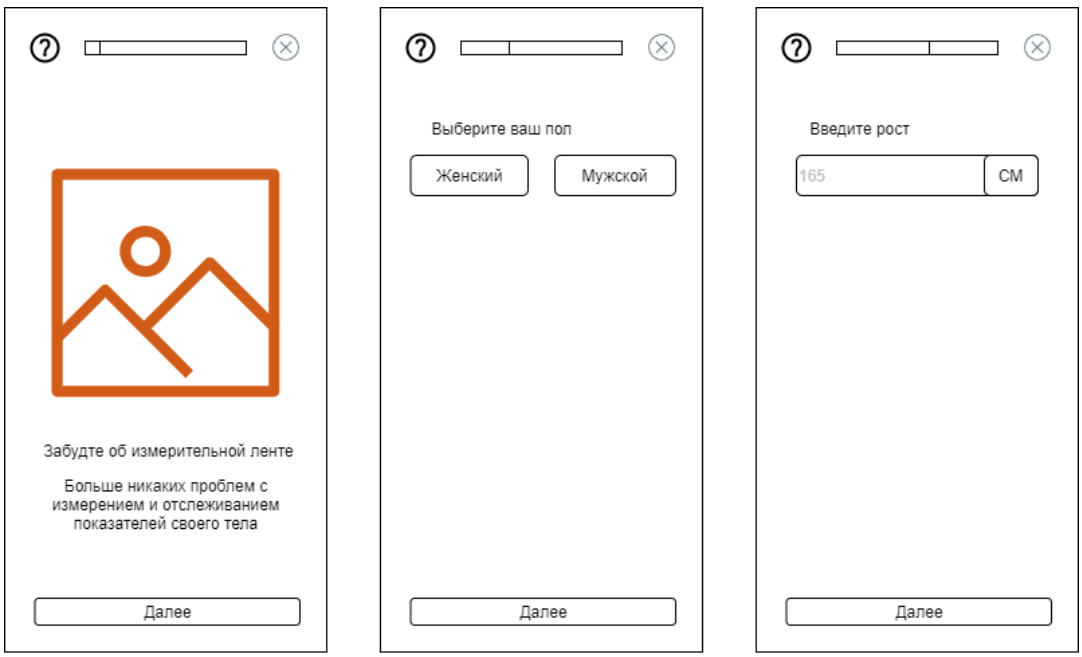


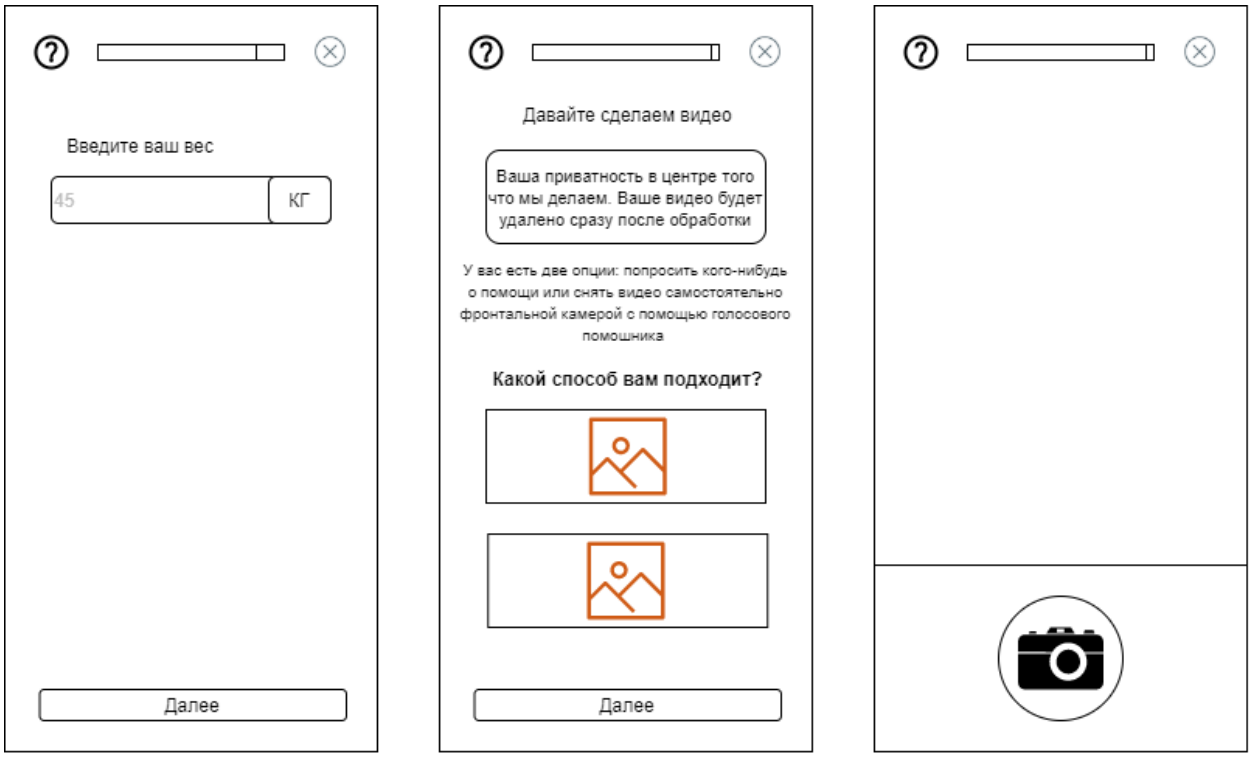
Рисунок 1 – Логическая схема интерфейса

- Физическое проектирование: Выбран язык HTML и JavaScript Framework VueJS

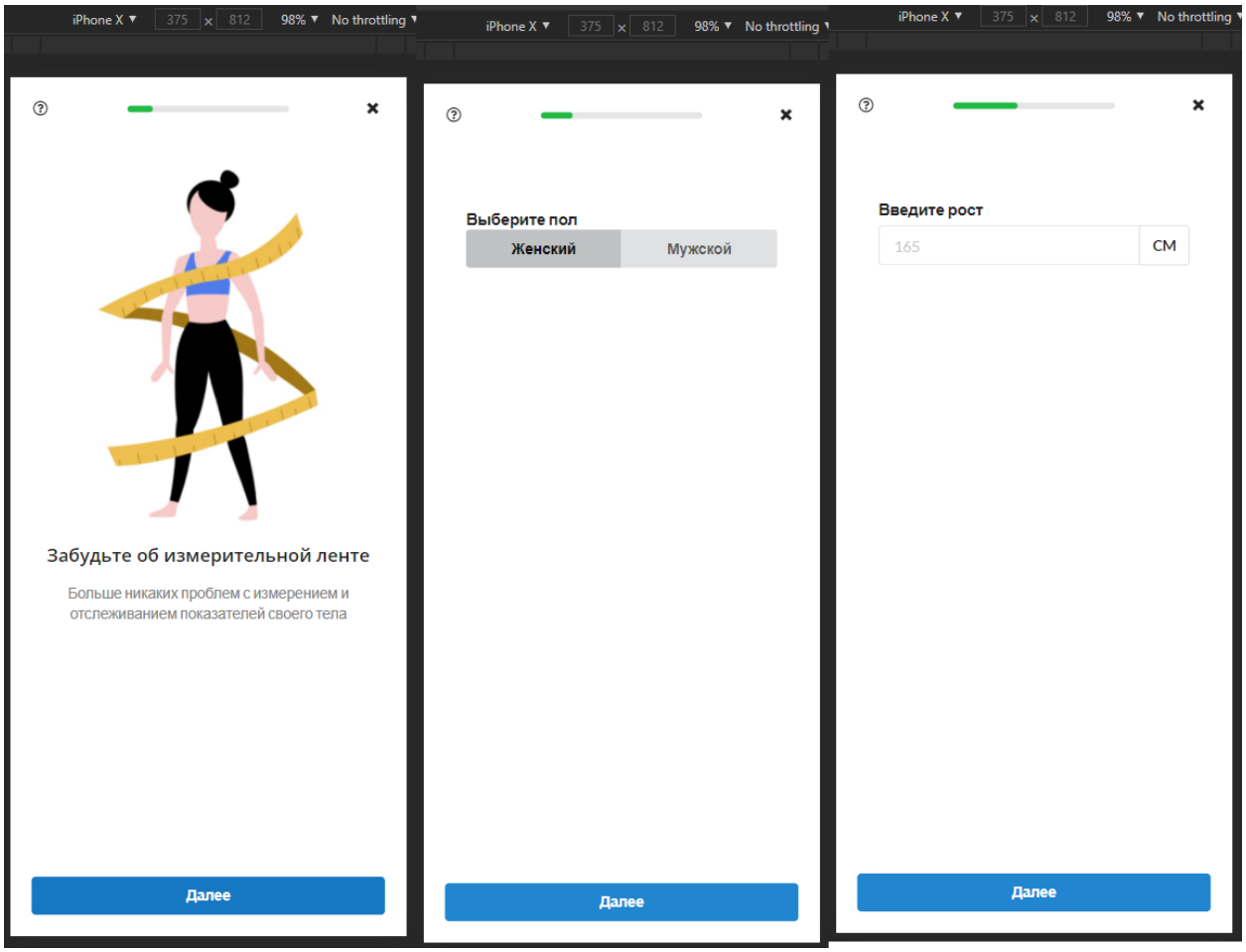
2. Реализация:

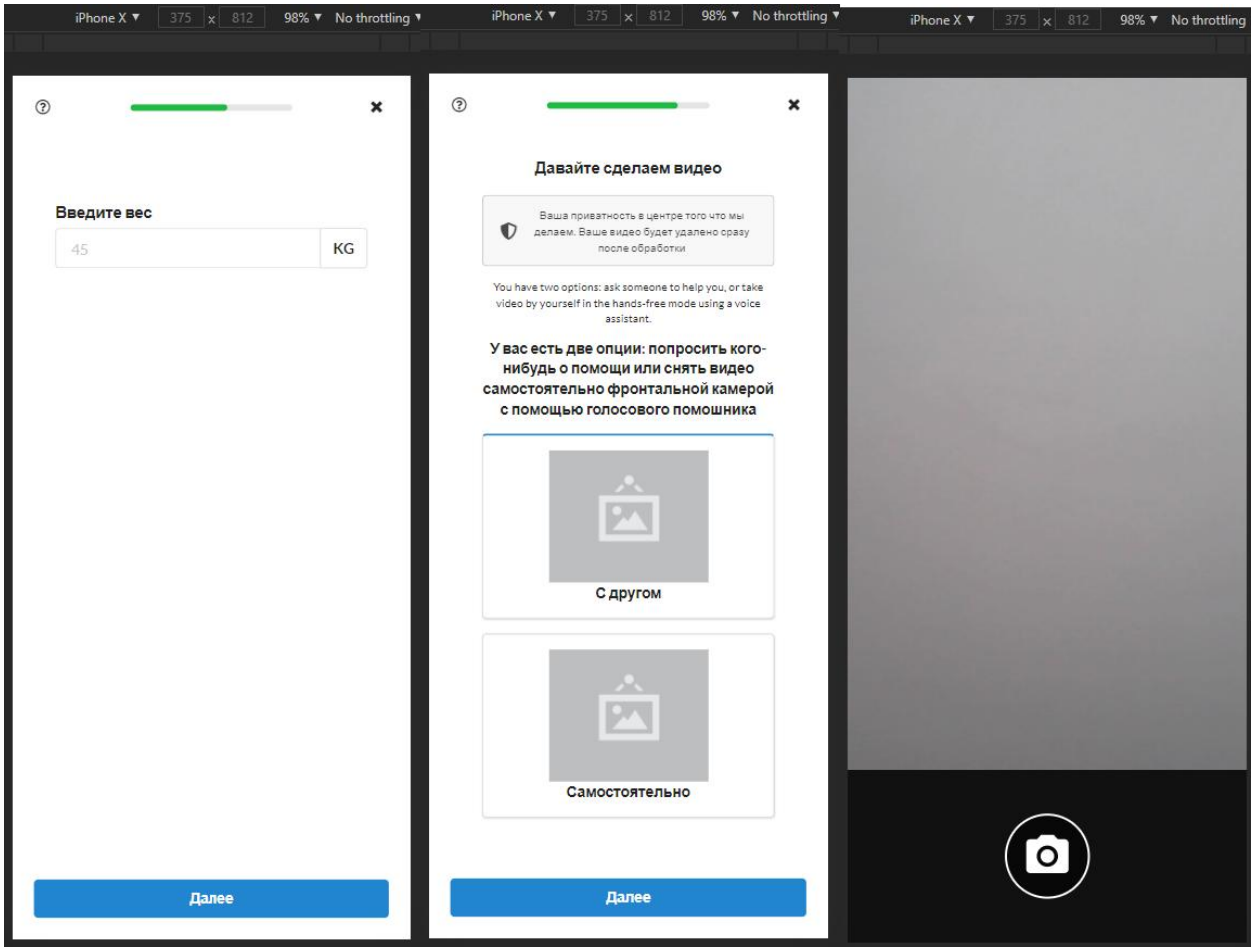
- Прототипирование:





- Конструирование:





Код реализации форм представлен в Приложении 1.

Вывод: в данной практической работе успешной спроектировал и реализовал интерфейс мобильного виджета