**3 РАЗРАБОТКА ПРИЛОЖЕНИЯ**

**3.1 Разработка диаграммы вариантов использования**

Мобильное приложение должно предоставлять следующие возможности:

* Регистрации и входу в учетную запись;
* Просмотру всех упражнений;
* Настройки профессии;
* Настройки предпочтительного времени;
* Оповещения о необходимости провести профилактическую тренировку;
* Просмотра и отслеживания результатов выполнения тренировки.

В Приложении A отображена диаграмма вариантов использования приложения.

**3.2 Разработка диаграмм деятельности**

Основной функцией данного дипломного проекта является возможность выполнять упражнения для профилактики профессиональных заболеваний в соответствии с определенными настройками. Соответственно, необходимо разработать диаграммы деятельности для следующих действий:

* Выполнение тренировки;
* Выполнение настроек;
* Оповещение о необходимости выполнить упражнение.

Диаграмма деятельности для функции «Выполнение тренировки» изображена в Приложении Б лист 1.

На диаграмме деятельности для функции «Выполнение тренировки» видно, что пользователь сначала просматривает список всех упражнений. Далее пользователь решает, выполнять ли тренировку сейчас или позже. Если позже – осуществляется выход из деятельности. Если пользователь решает начать тренировку сейчас, то пользователь должен будет прочитать описание упражнения и посмотреть видео, а затем выполнить упражнение. Если это упражнение последнее в списке – осуществляется выход из деятельности, если это одно из упражнений, но не последнее, повторяется цикл действий по просмотру и выполнению упражнений.

Диаграмма деятельности для функции «Выполнение настроек» изображена в Приложении Б лист 2.

Для выполнения настроек пользователь переходит на вкладку настроек. Если ему необходимо настроить удобное время тренировок, он переходит к настройке удобного времени, если необходимо осуществить настройку профессии – переходит к выбору профессии. Как только настройки закончены – осуществляется выход из деятельности.

Диаграмма деятельности для функции «Оповещение о необходимости выполнить упражнение» изображена в Приложении Б лист 3.

Как только наступает время выполнить профилактическую тренировку, приложение посылает уведомление пользователю. При клике на уведомление открывается тренировка с возможностью начать ее или отложить.

**3.3 Разработка структуры компонентов**

**3.3.1** Внедрение компонентного подхода. В целом – это то, как обычно решается задача структуризации приложения. Любая компонентная система должна уменьшать общую сложность через предоставление изоляции, или естественных барьеров, скрывающих сложность одних систем от других. Хорошая изоляция также облегчает повторное использование и внедрение сервисных парадигм.

Современные веб-приложения столь же сложны, как и любые другие программные приложения, и зачастую создаются несколькими людьми, объединяющими усилия для создания финального продукта. В таких условиях, чтобы повысить эффективность, естественно искать правильные способы разделения работы на участки с минимальными пересечениями между людьми и подсистемами. Внедрение компонентного подхода (в целом) – это то, как обычно решается такая задача. Любая компонентная система должна уменьшать общую сложность через предоставление изоляции, или естественных барьеров, скрывающих сложность одних систем от других. Хорошая изоляция также облегчает повторное использование и внедрение сервисных парадигм.

Изначально сложность веб-приложений в основном регулировалась со стороны сервера за счет разделения приложения на отдельные страницы, что требовало от пользователя соответствующим образом переходить в браузере с одной страницы на другую. С внедрением AJAX и связанных технологий разработчики смогли отказаться от потребности делать «переходы» между разными страницами веб-приложения. Для типичных сценариев вроде чтения почты или новостей ожидания пользователей изменились. К примеру, после логина в почту, вы можете «пользоваться почтовым приложением» с одного и того же адреса (URL) и находится на этой странице целый день (т.н. [Single-Page Applications](https://en.wikipedia.org/wiki/Single-page_application), SPA). Логика клиентских веб-приложений в таких ситуациях существенно усложняется, иногда она даже становится сложнее, чем на серверной стороне. Возможным разрешением данной сложности может являться дальнейшее разделение на компоненты и изоляция логики внутри одной страницы или документа.

Цель веб-компонентов в уменьшении сложности за счет изоляции связанных групп кода на HTML, CSS и JavaScript для выполнения общей функциональности в пределах контекста одной страницы.

Так как веб-компоненты должны связать воедино HTML, CSS и JavaScript, необходимо учитывать модели изоляции присущие каждой из технологий:

* Изоляция стилей в CSS;
* JavaScript и области видимости (замыкания);
* Изоляция глобального объекта.

Все эти механизмы покрывает система однофайловых компонентов VueJs.

[Компоненты](https://ru.vuejs.org/v2/guide/components.html) развивают идею плагинов. Каждый из них реализует какую-то свою возможность (а если нет существующих, то можно написать и свой). Если понадобится реализовать подобное в другом месте – легко переиспользовать плагин снова. Взаимодействие можно описать простым интерфейсом: отправляем в плагин входные параметры, а для обратной связи можем отслеживать события.

Всё это справедливо и для компонентов. С тем лишь отличием, что компонент может представлять собой не только одну вещь (например, красивый и функциональный select), но и какую-то часть приложения, которая должна работать и выглядеть везде единообразно (например, форма комментирования, с аватаркой, редактором и красивым select’ом).

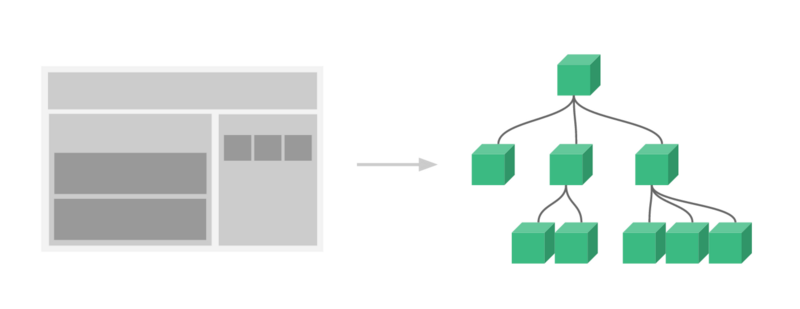


Рисунок 3.1 – Иллюстрация разделения страницы на компоненты

Компонентный подход позволяет избежать мешанины кода и чётко выстраивать архитектуру приложения. Любую сложную страницу всегда можно разбить на меньшие составляющие. Каждую из таких частей при выделении в компонент проще поддерживать, а при необходимости повторять разбиение внутри компонента на ещё меньшие части.

Первым заметным плюсом подобного разбиения на компоненты будет удобство в поддержке – больше не нужно держать в голове логику всего приложения, можно сосредоточиться на конкретной его части. Вносить изменения или доработки нужно будет только в одном месте. Изолированность же компонентов избавит от появления конфликтов с другими частями приложения.

Поэтому применение компонентного подхода теперь широко используется во многих фреймворках. Vue не остался в стороне и предоставляет прекрасные возможности по работе с компонентами.

Компонент и его структура. В разметке компоненты выглядят как нестандартные теги (например, <comment-form></comment-form>). Но компонент, в отличии от обычного тега, может “скрывать” под собой не только любую разметку, но и какую-то логику своей работы и использование других компонентов.

Описывать составляющие компонентов можно просто в коде страниц. В этом случае есть [некоторые ограничения](https://ru.vuejs.org/v2/guide/components.html#%D0%9E%D1%81%D0%BE%D0%B1%D0%B5%D0%BD%D0%BD%D0%BE%D1%81%D1%82%D0%B8-%D0%BF%D0%B0%D1%80%D1%81%D0%B8%D0%BD%D0%B3%D0%B0-DOM-%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD%D0%B0) по использованию и именованию (для браузера всё должно быть валидно). Vue также [имеет поддержку JSX](https://ru.vuejs.org/v2/guide/render-function.html#JSX).

Для удобной организации компонентов и их составляющих предлагается использовать [vue-файлы](https://ru.vuejs.org/v2/guide/single-file-components.html). Такой файл состоит из трёх частей:

* <template> для шаблона с разметкой компонента;
* <script> с логикой компонента;
* <style> со стилями компонента.

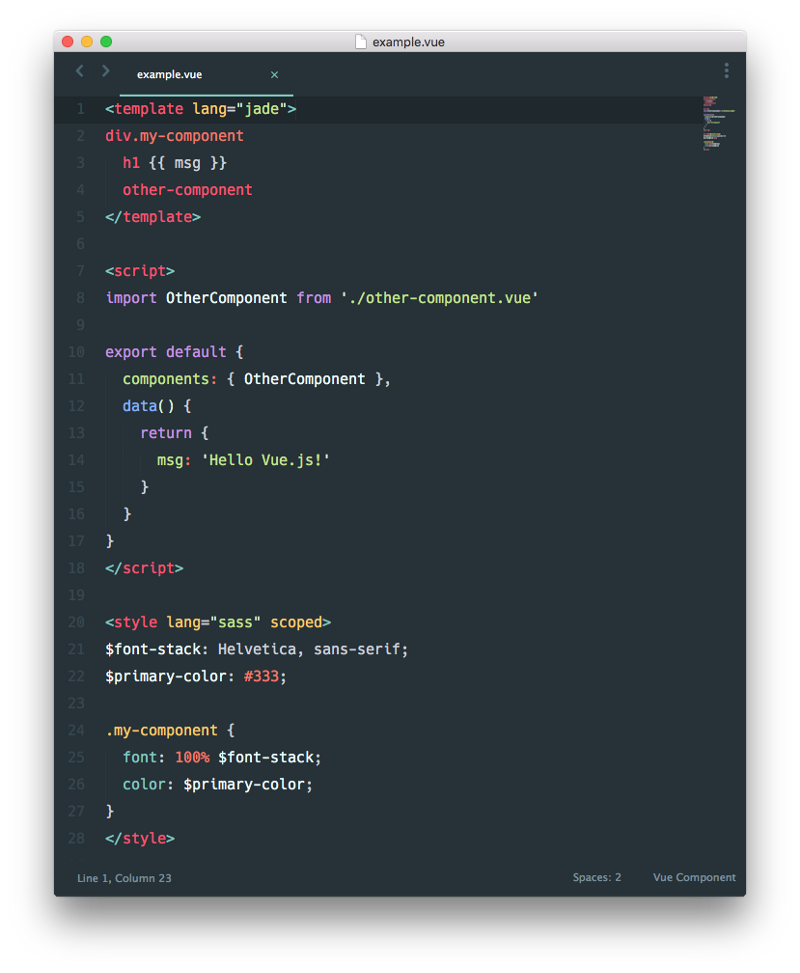


Рисунок 3.2 – Пример файла компонента.

**3.3.2** Разработка архитектуры приложения. В данной дипломной работе используется компонентный подход. Соответственно, необходимо разработать компонентную структуру и диаграмму компонентов для приложения.

Мобильное приложение состоит из следующих компонентов:

* Аpplication;
* Router;
* SignIn;
* SignUp;
* Exercise;
* Exercises;
* SearchSettings;
* Settings;
* PrefferedTime;
* Profession;
* TraininList;
* Training;
* TrainingResult.

Рассмотрим функции каждого компонента в отдельности.

Компонент Application представляет собой корневой компонент приложения. Он связывает все компоненты воедино в целостное приложение.

Компонент Router является библиотечным компонентом и осуществляет маршрутизацию в приложении.

Компонент SignIn представляет собой компонент, который отвечает за вход в приложение.

Компонент SignUp позволяет осуществлять регистрацию в приложении.

Компонент Exercise позволяет просматривать видео и описание одного конкретного упражнения.

Компонент Exercises позволяет просматривать все доступные упражнение, а так же осуществлять их фильтрацию. Включает в себя компонент SearchSettings, который отвечает за фильтрацию и поиск упражнений по критериям, профессиям и названию упражнения.

Компонент Settings отвечает за настройку приложения. Включает в себя компоненты:

* PrefferedTime;
* Profession.

Компонент PrefferedTime позволяет добавлять, изменять и в целом настраивать удобное для пользователя время профилактической тренировки.

Компонент Profession позволяет настраивать текущую профессию.

Компонент TrainingList представляет собой список с упражнениями данной тренировки. В нем отображены и отсортированы в порядке сложности все упражнения данной тренировки. Включает в себя компонент Training. Компонент Training отвечает за отображение конкретного упражнения в тренировке. Здесь пользователь может просмотреть видео с примером выполнения упражнения, посмотреть описание и к-во повторение конкретно для него в данном упражнении.

Компонент TrainingResult отвечает за отображение результата тренировки.

Диаграмма компонентов представлена в Приложении В.