МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**
Кафедра математической кибернетики и компьютерных наук

РАЗРАБОТКА ПРОГРАММЫ-ОРГАНАЙЗЕРА СРЕДСТВАМИ ФРЕЙМВОРКА REACT

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 451 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Захарова Богдана Денисовича

Научный руководитель
к. ф.-м. н., доцент _____ А. С. Иванова

Заведующий кафедрой
к. ф.-м. н., доцент _____ С. В. Миронов

Саратов 2022

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью бакалаврской работы было создание приложения, представляющее собой менеджер задач и отдыха, использующий методологию геймификации, работающую по принципу получения поощрений за выполнение заданий в виде очков и траты их на проведение досуга. Были поставлены следующие задачи, которые должна решать программа:

1. создание заданий с определенными характеристиками (важность, сложность, мотивация) с последующим вычислением награды за их выполнение в очках;
2. добавление заданий со сроком сдачи с соответствующими штрафами за опоздание или бонусами за своевременное выполнение;
3. фильтрация и сортировка заданий на основе различных характеристик, в том числе фильтрация по группам, классифицирующим задания;
4. покупка отдыха за заработанные очки;
5. возможность добавлять персонажей, привязывать их к заданиям или проводимому отдыху;
6. реализация развития отношений с персонажами в процессе выполнения и невыполнения заданий;
7. изображение графиков-статистики по выполненным заданиям и проведенному отдыху;

Актуальность создания подобного приложения обусловлена малым количеством представленных на рынке программного обеспечения приложений-таскменеджеров, использующих методы геймификации. С помощью данного приложения пользователь сможет в игровой форме организовать как работу, так и досуг, а также более эффективно поддерживать социальные связи.

Для реализации приложения был использован следующий ряд технологий:

- язык программирования Typescript;
- фреймворк React и дополнительные React-библиотеки;
- библиотека управления фронтенд-состоянием приложения MobX;
- база данных Firebase;
- инструмент для стилизации компонентов JSS.

1 Теоретические сведения о языке Typescript

1.1 Основы языка Typescript

TypeScript — это расширенная версия JavaScript, представленная в 2012 году, которая позволяет упростить разработку крупных приложений, оперирующих большим количеством сложных структур [1].

В языке определены все стандартные типы данных, а также определен тип any. При определении функции необходимо типизировать параметры и возвращаемое значение. В анонимных функциях параметры определяются автоматически на основе контекста [**?**]. Наконец, на случай многократного использования объектных типов в языке есть возможность создавать свои собственные типы (alias) [2]

1.2 Работа с фреймворком React

React - это библиотека, использующаяся для создания пользовательского интерфейса, выпущенная в 2013 году.

Чтобы избежать снижения производительности при манипулировании html-элементами с помощью JS (TS), была реализована концепция виртуального DOM, с которым и взаимодействует фреймворк.

При работе с проектом, использующим React, верстка осуществляется с помощью компонентов внутри JSX (TSX) файлов. Это расширение комбинирует код JS (TS) и XML, тем самым предоставляя простой и интуитивно понятный способ для определения кода визуального интерфейса [3].

Компоненты React изменяются на основе их состояния [4]. Состояние — это инструмент, который дает обновлять пользовательский интерфейс, основываясь на событиях.

Для реализации многостраничного приложения и навигации по нему используется библиотека react-router. Для отрисовки графиков — библиотека recharts.

1.3 Фронтенд-состояние MobX

Mobx — это библиотека, делающая управление состоянием приложения простым и масштабируемым, применяя функционально-реактивное программирование [5].

Сочетание Mobx и React дает возможность решения общих проблем разработки приложений. Если React позволяет оптимально отрисовывать интер-

фейс с помощью виртуального DOM, уменьшая количество дорогостоящих манипуляций над реальным DOM, то Mobx синхронизирует состояние между компонентами [6], используя реактивную виртуальную зависимость графического состояния, обновляемого только в случае необходимости.

1.4 База данных Firebase

Firebase — это облачная база данных, позволяющая пользователям хранить и получать сохраненную информацию. Она хранит текстовые данные в JSON-формате и предоставляет удобные методы для всех CRUD-операций [7].

Firebase очень быстро подключается в проект, серверная часть реализована самим сервисом.

После создания базы, разработчик получит ее конфигурацию, которую можно будет добавить в проект. Для взаимодействия с элементами базы необходимо обращаться к референсным ссылкам путей, по которым располагаются данные.

1.5 Использование библиотеки для стилизации JSS

JSS — это удобный инструмент для стилизации компонентов. Стили могут храниться внутри констант, формируясь с помощью метода createUseStyles [8]. Константы можно импортировать в компонент и вызвать в виде хука, после чего обращаться к классам.

1.6 Геймификация и приложения на ее основе

Геймификация — это внедрение игровых форм в неигровой контекст. Данная методология использует естественные склонности людей к конкуренции, сотрудничеству, соревнованиям и достижениям [9].

Разработанное в рамках выпускной работы приложение, в отличие от, например, аналога Habitica, дает возможность тратить заработанные очки не на внутриигровые вещи для своего аватара, а непосредственно на свой собственный отдых. Также в разработанном приложении реализованы социальные взаимодействия с персонажами. Данная механика широко распространена лишь в ролевых компьютерных играх, где игрок, выполняя задания различных фракций, улучшает с ними отношения.

2 Программный продукт

2.1 Архитектура приложения

В папке `models` находятся модели используемых в приложении сущностей, в папках `stores` и `hooks` соответственно находятся стор и хук для его использования [10], в папке `modules` располагаются вспомогательные функции, наконец, в папках `components` и `styles` соответственно можно найти рендеры компонентов и их стили.

2.2 Создание и подключение базы Firebase

При создании базы была получена ее конфигурация, впоследствии записанная в файл глобальных переменных. С данной конфигурацией база была проинициализирована в файле `firebase.ts` [11]. База импортируется в переменной `database`.

2.3 Роутинг и создание контекста

В корневом файле рендера `index.tsx` инициализируется глобальный `Store`, а также импортируется его контекст. В файле `App.tsx` располагается роутер [12], осуществляющий навигацию по приложению. В компоненте `Header` в формате меню находятся ссылки, позволяющие переходить по путям, перечисленным в компоненте `Switch`.

2.4 Управление фронтенд-состоянием приложения с помощью MobX

В сторе используемые в программе сущности объявляются вместе с декораторами [13] `@observable`, что позволяет отслеживать их изменения. ри экспорте компоненты, она оборачивается в декоратор `observer`, показывая, что она подписана на данные стора. Геттеры, используемые для получения данных, помечены декоратором `@computed`, методы, изменяющие состояние стора, помечены как `@action` [14].

2.5 Взаимодействие с базой данных Firebase

Все необходимые данные программа получает по ссылкам, указывающим на эндпоинты. Чтобы своевременно получать обновляемые в процессе работы с приложением данные, в конструкторе стора внутри функции `runInAction` объявлены обработчики `onValue` [11], которые в случае любого

изменения данных копируют их и сохраняют в соответствующие атрибуты. Получение и удаление данных осуществляется с помощью методов get и remove соответственно.

2.6 Используемые модели и их связь между собой

В программе используется две основных модели: квесты, которые дают очки за их выполнение, и покупки отдыха, снимающие эти очки. Покупки делаются на основе видов досуга, перечисленных в разделе «Прейскурант». Также есть модель персонажа, которая может использоваться в заданиях и в покупках.

2.7 Формирование и получение награды за задания

При создании задания учитываются три главные характеристики: сложность, важность и мотивация. Исходя из выбранного уровня каждой из этих характеристик, определяется коэффициент. Чем выше сложность, важность и чем меньше мотивация, тем выше соответствующие модификаторы. Эти три коэффициента умножаются на базовое значение в 100 очков. Статус задания при его успешном завершении меняется с ACTIVE на COMPLETED, а пользователь получает очки награды. В случае отмены статус снова меняется на активный, а очки отнимаются назад.

2.8 Задания со сроком выполнения

В случае добавления срока выполнения, регулярно будет вычисляться разница в днях между текущей датой и сроком выполнения. Эта разница влияет на модификатор даты, который будет дополнительным модификатором при подсчете награды за выполнение задания. Если разница отрицательная, то задание выполнено с опозданием, модификатор меньше единицы. Если разница положительная, то задание выполнено заранее, модификатор больше или равен единице.

2.9 Приобретение отдыха за очки

Награду, полученную за выполнение заданий, можно тратить на покупку отдыха. При этом берется соответствующий вид отдыха, в покупку записываются его данные. Стоимость формируется исходя из цены за минуту и количества проведенных минут. При добавлении покупки ее стоимость сразу

вычитается из общего количества очков. В случае отмены покупки ее стоимость прибавляется к количеству очков, а запись просто удаляется из базы покупок.

2.10 Отношения с персонажами

При создании персонажа выбирается начальный статус отношений и устанавливается стартовое количество очков отношений. За выполнение заданий или покупку отдыха вместе с персонажем пользователь получает очки отношений. За день бездействия (с персонажем не было никаких взаимодействий) теряется какое-то количество очков отношений. Чем выше статус отношений, тем больше очков за выполнение активностей и тем меньше штраф за бездействие. И наоборот, чем меньше статус, тем меньше очков за выполнение и тем больше штраф. При достижении пороговых значений статус меняется.

2.11 Формирование данных для статистики

Чтобы сформировать данные для отображения статистики по выполненным заданиям, сперва нужно отфильтровать только выполненные задания. Из них формируется `map`, который для каждой даты хранит количество выполненных в этот день заданий (или проведенного отдыха).`Map` затем преобразуется в массив объектов и сохраняется в стор. После этот массив передается в рендер статистики.

2.12 Верстка приложения

Для данных каждого вида: квесты, группы, отдых, покупки и персонажи — реализованы идентичные рендеры разделов. Пусть `Model` — это один из перечисленных видов данных. Таким образом, есть главный компонент `Model`, контент модального окна добавления элемента `ModelAdd`, контент модального окна редактирования элемента `ModelEdit`, виртуализованный список элементов `ModelList`, элементы этого списка `ModelElement`, текущий просматриваемый элемент `ModelSelected`.

2.13 Модальное окно

Для создания и редактирования данных используется модальное окно [15] — компонент, накладываемый поверх остального приложения, затемняющий область под собой.

Окно состоит из двух div. Внешний `div` отвечает за затемненную область вне модального окна. Внутренний же `div` отвечает за само окно. Здесь с помощью метода `stopPropagation()` отменяется эффект нажатия родительского элемента [16].

За скрытие и показ модального окна отвечает свойство `opacity`, прописанное в стилевом файле для окна. В зависимости от истинного или ложного значения `active` элемент `div` принимает класс с `opacity` равным 0 или 1 соответственно [17]. Чтобы окно отображалось поверх всех элементов и не перекрывалось ничем, используется свойство `zIndex` [18].

2.14 Добавление элементов

Модальные окна добавления элементов содержат обязательные, отмеченные звездочкой, и необязательные для заполнения поля. Нажатие на кнопку (галочку) создаст новый элемент в базе. Произойдет обращение к соответствующему методу добавления, находящемуся в сторе.

2.15 Список элементов

Все массивы данных отображаются в виде списка блочных элементов. Компонент является виртуализованным списком из библиотеки `Virtuoso` [19].

Элемент списка содержит минимум информации: имя элемента и опциональные кнопки в зависимости от раздела. При клике на блок вызывается метод `selectModel`, меняющее свойство стора `selectedModel` на значение выбранного элемента. При этом блок меняет свой класс на класс выделенного элемента, если его идентификатор совпадает с идентификатором выделенного элемента.

В списке квестов блок меняет цвет в зависимости от статуса задания: белый, если задание в процессе, зеленый, если оно выполнено и красный, если оно провалено. Также в списке квестов реализована возможность отфильтровать и отсортировать данные с помощью `select` элементов, на которые назначены обработчики событий, сохраняющие в стор информацию о том, какие данные необходимо показать и как именно.

2.16 Выбор элемента

Выбранный элемент списка отображается справа в блоке. В нем представлена подробная информация об элементе, кнопки редактирования и уда-

ления (для выполненных и проваленных квестов — только кнопки удаления).

2.17 Редактирование элемента

Модальное окно редактирования элемента схоже с окном добавления, однако в случае редактирования в поля формы по умолчанию сохраняются значения выбранного элемента. Нажатие на кнопку (галочку) редактирует элемент в базе. Произойдет обращение к соответствующему методу редактирования, находящемуся в сторе.

2.18 Отображение статистики

Отображение статистики осуществляется посредством компонента библиотеки `recharts` — линейного графика `LineChart` [20]. Раздел статистики включает в себя два компонента с графиками: один для выполненных квестов, второй для проведенного отдыха. В качестве массива значений на вход подается сформированный ранее датасет на основе всех выполненных заданий. Для отдыха график выглядит аналогично, но только используется записи всех приобретенных покупок.

ЗАКЛЮЧЕНИЕ

В настоящей работе было реализовано веб-приложение, обладающее функциональностью менеджера задач и отдыха, награждающего пользователя за выполнение заданий и позволяющего использовать полученные очки для покупки отдыха. Также разработана система отношений с персонажами, которых пользователь может привязывать к заданиям или отдыху. В зависимости от уровня отношений, пользователь получает или теряет очки отношений в процессе выполнения или невыполнения связанных активностей, тем самым улучшая или ухудшая уровень отношений. Для каждой описанной модели реализованы все CRUD-операции. Присутствует возможность создавать задания с крайним сроком выполнения, выполнение вовремя или с опозданием влияет на получаемые очки. В разделе статистики пользователь может увидеть графики выполненных заданий или проведенного отдыха по дням.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Вводный курс по TypeScript [Электронный ресурс]. — URL: https://tproger.ru/translations/course-on-typescript/ (Дата обращения 19.05.2022). Загл. с экр. Яз. Рус.

2 *Дэн, В.* Эффективный TypeScript: 62 способа улучшить код / В. Дэн. — СПб: Питер, 2022. — 288 с.

3 Основы JSX [Электронный ресурс]. — URL: https://metanit.com/web/react/1.3.php (Дата обращения 26.05.2022). Загл. с экр. Яз. Рус.

4 React документация [Электронный ресурс]. — URL: https://ru.reactjs.org/docs/getting-started.html (Дата обращения 25.05.2022). Загл. с экр. Яз. Рус.

5 *Бэнкс, А.* React и Redux. Функциональная вебразработка / А. Бэнкс. — СПб: Питер, 2018. — 336 с.

6 *Wieruch, R.* Taming the State in React: Your jorney to master Redux and MobX / R. Wieruch. — Scotts Valley: CreateSpace Independent Publishing Platform, 2022. — 258 pp.

7 Что такое Firebase [Электронный ресурс]. — URL: https://kolmogorov.pro/what-is-firebase-chto-takoe (Дата обращения 02.06.2022). Загл. с экр. Яз. Рус.

8 React JSS - альтернативный способ стилизации компонентов [Электронный ресурс]. — URL: https://highload.today/react-jss-alternativnyj-sposob-stilizatsii-komponentov/ (Дата обращения 30.05.2022). Загл. с экр. Яз. Рус.

9 Геймификация: как игровой подход помогает в обучении и на работе [Электронный ресурс]. — URL: https://trends.rbc.ru/trends/education/605c6f2f9a79473a61646994 (Дата обращения 29.05.2022). Загл. с экр. Яз. Анг.

10 *Мартин, Р.* Чистая архитекутура. Искусство разработки программного обеспечения / Р. Мартин. — СПб: Питер, 2018. — 352 с.

11 Firebase документация [Электронный ресурс]. — URL: https://firebase.google.com/docs (Дата обращения 20.05.2022). Загл. с экр. Яз. Рус.

12 React Router Documentation [Электронный ресурс]. — URL: https://reactrouter.com/docs/en/v6 (Дата обращения 22.05.2022). Загл. с экр. Яз. Анг.

13 *Padila, P.* MobX Quick Start Guide / P. Padila. — Birmingham: Packt Publishing, 2018. — 236 pp.

14 Начало работы с MobX [Электронный ресурс]. — URL: https://russianblogs.com/article/32511627402/ (Дата обращения 23.05.2022). Загл. с экр. Яз. Рус.

15 *Хавербеке, М.* Выразительный JavaScript. Современное веб-программирование / М. Хавербеке. — СПб: Питер, 2022. — 480 с.

16 Руководство по React [Электронный ресурс]. — URL: https://metanit.com/web/react/ (Дата обращения 27.05.2022). Загл. с экр. Яз. Рус.

17 JSS Introduction [Электронный ресурс]. — URL: https://cssinjs.org/ (Дата обращения 24.05.2022). Загл. с экр. Яз. Анг.

18 *Даккет, Д.* HTML и CSS. Разработка и дизайн веб-сайтов / Д. Даккет. — Москва: Эксмо, 2022. — 480 с.

19 Virtouoso docs [Электронный ресурс]. — URL: https://virtuoso.dev/ (Дата обращения 29.05.2022). Загл. с экр. Яз. Анг.

20 React Recharts Documentation [Электронный ресурс]. — URL: https://recharts.org/en-US/ (Дата обращения 27.05.2022). Загл. с экр. Яз. Анг.