

ФГАОУ ВО «САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ»

КАФЕДРА ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

КУРСОВАЯ РАБОТА

Программирование интернет-приложений

Саржевский Иван Анатольевич

Лабушев Тимофей Михайлович

Группа Р3202

Санкт-Петербург

2018 г.

Sumaju nikki

БРАУЗЕРНАЯ МНОГОПОЛЬЗОВАТЕЛЬСКАЯ ТЕКСТОВАЯ ИГРА
ЖАНРА ZERO-PLAYER GAME

Содержание

1	Введение	2
1.1	Цель создания	2
1.2	Целевая аудитория	2
1.3	Концепция игры	2
2	Прецеденты использования	3
2.1	Описание прецедентов	3
2.2	UML-диаграмма прецедентов	6
3	Требования к аппаратно-программному обеспечению	7
3.1	Требования к серверному обеспечению	7
3.2	Требования к клиентскому обеспечению	7
4	Требования к архитектуре системы	7
4.1	Глоссарий	7
4.2	Уровень back-end	7
4.3	Уровень front-end	8
4.4	Telegram-бот	8
5	Архитектура системы	9
6	Прототипы интерфейсов системы	10

1 Введение

Предметом разработки является платформа для многопользовательской текстовой игры жанра *Zero-Player Game (ZPG)*, использующая браузерный интерфейс и сохраняющая прогресс игрока на игровом сервере.

В основе жанра лежит концепция протекания игрового процесса без вмешательства со стороны пользователя. При этом игроку предоставляются возможности ускорения своего игрового прогресса и взаимодействия с другими пользователями.

Предполагается, что взаимодействие с платформой будет представлять собой не длинные сессии, свойственные традиционным играм, а короткие, но частые посещения, характерные для социальных сетей.

1.1 Цель создания

Платформа разрабатывается с целью предоставления творческим молодым людям возможности самовыражения в игровой форме. Платформа предоставляет набор базовых игровых механик, в то время как содержательная часть создается самими пользователями, путём предоставления возможности формировать предложения для контента

1.2 Целевая аудитория

Творческие молодые люди, в возрасте 18-25 лет, знакомые с современными медиа, активные пользователи интернета. Не имеют времени/желания проводить много времени за игрой в компьютерные игры, ищут более подходящего под свой стиль жизни формата.

1.3 Концепция игры

События игры разворачиваются в здании средней школы. Огромное количество кабинетов соединены длинными темными переходами, в которых прячутся опасные создания. Бои с ними — такая же повседневность для учеников школы, как монотонные уроки, веселые клубы и неуклюжие отношения.

Игрок принимает роль призрака-хранителя, сверхъестественного существа, которое охраняет ученика. Ученики о своих помощниках даже не задумываются, поскольку призраки лишены физического контакта со смертными. Помогают хранителям совы, когда-то охранявшие холлы школы.

Грозные создания, пришедшие на смену мудрым и храбрым совам, заточили птиц в тесных клетках, раскиданных по всему зданию. Встречая сов, ученики выпускают их на свободу, не ожидая ничего взамен (*возможно, они лучше всех понимают, насколько тускла жизнь в заточении*).

Будучи птицами чести, совы возвращаются к призраку-хранителю освободившего их ученика, чтобы оказать ответную услугу.

2 Прецеденты использования

2.1 Описание прецедентов

Регистрация

Цель: Завести аккаунт в системе, создать персонажа

Сценарий:

1. Незарегистрированный пользователь открывает любую страницу платформы.
2. Система перенаправляет пользователя на главную страницу с формой регистрации.
3. Пользователь вводит адрес электронной почты и пароль, отправляет форму.
4. Система создает аккаунт и перенаправляет пользователя на страницу создания персонажа.
5. Пользователь указывает имя и пол персонажа, опционально загружает картинку, отправляет форму.
6. Система авторизует пользователя.

Расширения:

- На шаге 3 пользователю предоставляется возможность регистрации используя аккаунт социальной сети.
- Ошибка на шаге 4 (адрес электронной почты уже используется, пароль содержит менее восьми символов) предотвращает переход к следующему шагу.

Авторизация

Цель: Войти в систему

Сценарий:

1. Незарегистрированный пользователь открывает любую страницу платформы.
2. Система перенаправляет пользователя на главную страницу с формой входа.
3. Пользователь вводит адрес электронной почты и пароль, отправляет форму.
4. Система авторизует пользователя и перенаправляет его на страницу персонажа.

Расширения:

- На шаге 3 пользователю предоставляется возможность авторизоваться, используя аккаунт социальной сети.
- На шаге 4 возникновение ошибки (адрес электронной почты уже используется, пароль содержит менее восьми символов) предотвращает переход к следующему шагу.

Наблюдение за игровым процессом

Цель: Отследить развитие персонажа, прочесть интересные записи в дневнике

Сценарий:

1. Авторизованный пользователь открывает страницу персонажа.
2. Система предоставляет обзор характеристик персонажа, его текущее занятие и цели для повышения уровня.
3. За один клик пользователь имеет возможность перейти к дневнику персонажа, энциклопедии известных персонажу существ, карте мира.

Расширения:

- Пользователь может увидеть краткую информацию о состоянии персонажа (его здоровье и текущее занятие) в шапке любой страницы игрового портала.

Участие в игровом процессе

Цель: Оказать влияние на развитие персонажа

Сценарий:

1. Авторизованный пользователь открывает меню *сов* с помощью кнопки, доступной в шапке любой страницы игрового портала.
2. Пользователь выбирает из списка доступных сов одну.
3. Система добавляет сову во внутреннюю очередь эффектов хода, уменьшает количество доступных сов выбранного типа на одну, показывает пользователю уведомление об успешном начале полета птицы.
4. По завершению хода в дневнике персонажа появляется заметка о действии совы.

Расширения:

- Число возможных сов одного типа ограничено, что должно быть понятно пользователю из интерфейса системы.

Взаимодействие с другими игроками

Цель: Повлиять на отношения персонажа с другими учениками, усилив таким образом влияние на персонажа сов

Сценарий:

1. Авторизованный пользователь выбирает *чернильную сову* из списка доступных.
2. Система предоставляет пользователю форму с поиском персонажа-получателя по имени, в которой доступно автодополнение и список последних адресатов.
3. Пользователь выбирает получателя.
4. Система случайно выбирает *клуб*, для которого создается запись о письме. При посещении персонажем-получателем клуба, его отношения с персонажем пользователя изменяются.
5. В дневники обеих персонажей добавляется запись об изменении отношений.

Расширения:

- Действие невозможно, если у пользователя отсутствует необходимый тип совы.
- Улучшение отношений ведёт к более частым упоминаниям игроков в дневниках и усилению влияния сов на персонажа

Предложение пользовательского контента

Цель: Добавить в систему свой контент, получить за это вознаграждение (новых сов)

Сценарий:

1. Авторизованный пользователь открывает страницу предложений.
2. Система предоставляет пользователю информацию о состоянии существующих предложений (*приняты, отклонены*) и прогрессу до следующей награды.
3. Пользователь выбирает класс добавляемого контента (*существо, клуб, урок*) и переходит к форме редактирования.
4. Пользователь добавляет записи для дневника и отправляет запрос на добавление контента.

Расширения:

- На шаге 4 пользователь указывает один из существующих клубов (уроков) или создает новое существо с произвольным именем.
- Пользовательские предложения проходят модерацию перед появлением в дневниках персонажей. Интерфейс модерирования контента реализуется как Telegram-бот.

Оценка пользовательского контента

Цель: Поощрить создателей интересных записей

Сценарий:

1. Авторизованный пользователь открывает страницу дневника персонажа.
2. Пользователь отмечает понравившуюся запись, нажимая на кнопку напротив нее.
3. Система сохраняет информацию о количестве одобрений у записи.

Расширения:

- Пользователь, отправивший администрации запись, может отследить количество ее одобрений на странице предложений. По достижению определенного числа одобрений пользователь получает сову, вид которой ему заранее не известен.

2.2 UML-диаграмма прецедентов



3 Требования к аппаратно-программному обеспечению

3.1 Требования к серверному обеспечению

Система, которая обеспечивает выполнение программных продуктов сервера приложений и хранение данных платформы, должна отвечать следующим требованиям:

- Поддержка операционной системой бинарного интерфейса приложений (ABI) Linux
- Наличие сервера баз данных PostgreSQL версии 9.6 и выше

3.2 Требования к клиентскому обеспечению

Браузерный интерфейс разрабатывается с учетом следующих требований к программному обеспечению на стороне пользователя:

- веб-браузер Google Chrome версии 67 и выше или Mozilla Firefox версии 61 и выше с включенным интерпретатором сценариев JavaScript,
- отсутствие запрета веб-страницам платформы доступа к внешним ресурсам, а именно изображениям, шрифтам, таблицам стилей CSS и сценариям JavaScript, в том числе блокировщиками рекламы.

4 Требования к архитектуре системы

4.1 Глоссарий

Уровень back-end — серверное приложение, с которым взаимодействует пользовательский интерфейс игры. Уровень back-end обеспечивает хранение данных, расчет игровых процессов, взаимодействие между пользователями.

Уровень front-end — браузерное приложение, которое реализует пользовательский интерфейс игры. Уровень front-end включает в себя HTML-страницы, сценарии JavaScript, таблицы стилей CSS.

4.2 Уровень back-end

1. Серверное приложение должно разрабатываться на платформе *JVM* с использованием фреймворков *Akka* и *Play*.
2. Взаимодействие между уровнями front-end и back-end должно осуществляться посредством REST API; возможно использование протокола WebSockets для реализации обновлений в режиме реального времени.
3. Серверное приложение должно реализовывать аутентификацию пользователей с поддержкой входа через социальные сети (OAuth), используя библиотеку *play-silhouette*. Пароли пользователей должны храниться как криптографический хэш.

4. Серверное приложение должно публиковать push-уведомления об игровых событиях, используя сервис *Firebase Cloud Messaging*.
5. Серверное приложение должно отправлять пользователям еженедельное новостное сообщение электронной почты, используя *JavaMail API*.
6. Серверное приложение должно предоставлять API для доступа к очереди пользовательских предложений, а также их принятию, отклонению и редактированию (см. раздел 4.4).

4.3 Уровень front-end

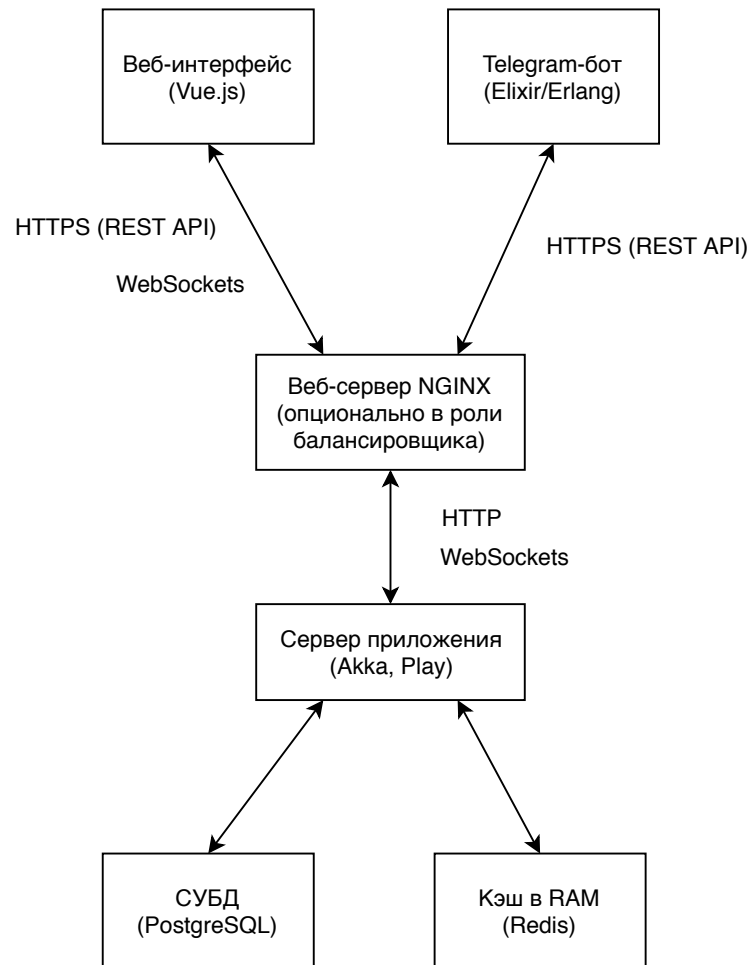
1. Клиентское приложение должно разрабатываться с использованием фреймворка Vue.js.
2. Веб-интерфейс должен быть адаптирован для отображения в трех режимах: *desktopном* (ширина экрана больше 1107px), *планшетном* (больше 889px) и *мобильном* (меньше 889px).
3. Веб-интерфейс должен предоставлять пользователю возможность подписаться на push-уведомления.
4. Веб-интерфейс должен распознаваться в мобильной ОС Android как приложение, используя *Web App Manifest*.

4.4 Telegram-бот

Модерация добавляемого пользователями контента должна осуществляться при помощи Telegram-бота.

1. Бот должен работать в заранее определенном чате, в котором находятся администраторы платформы.
2. Предложения должны выноситься ботом по одному на обсуждение в чат. Администраторы должны иметь возможность принятия, отклонения, редактирования предложения.

5 Архитектура системы



6 Прототипы интерфейсов системы

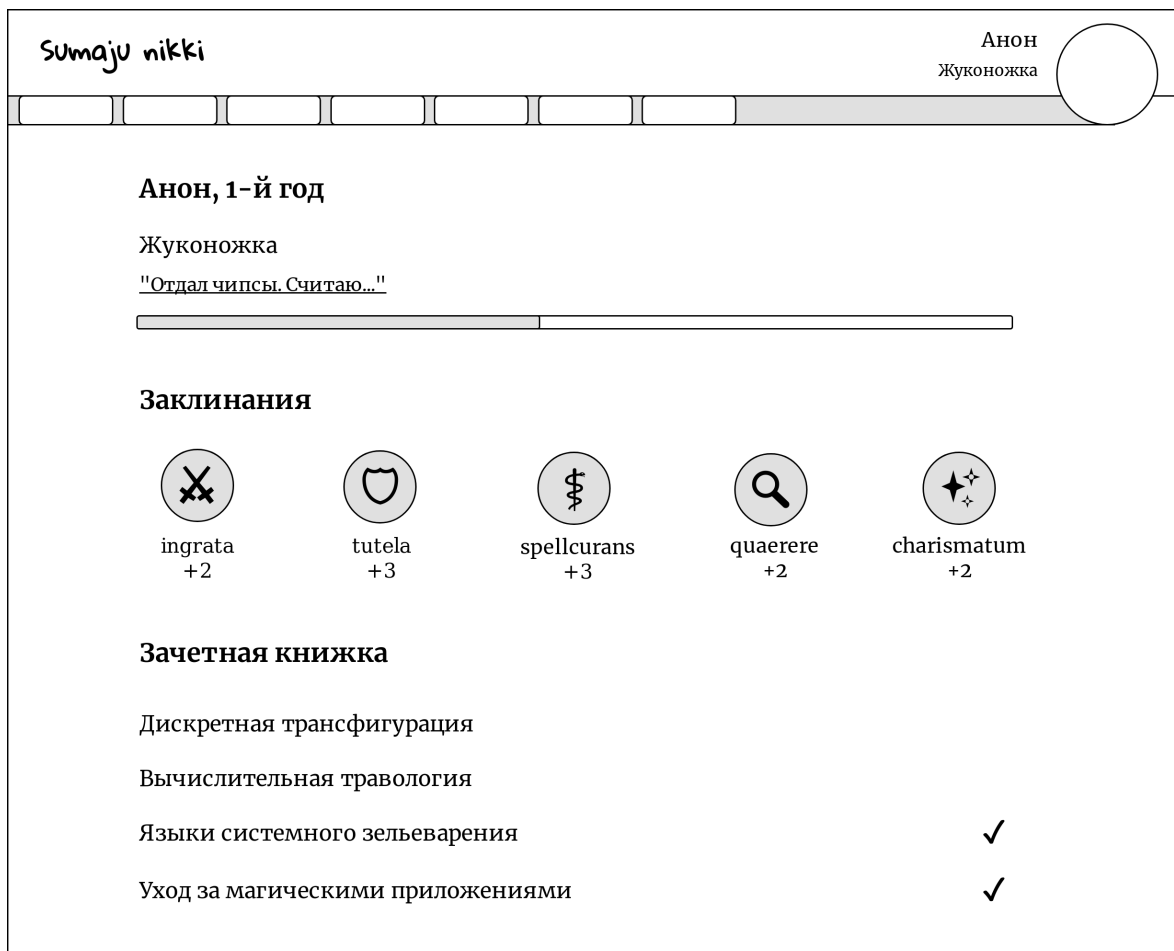


Рис. 1: Обзор персонажа

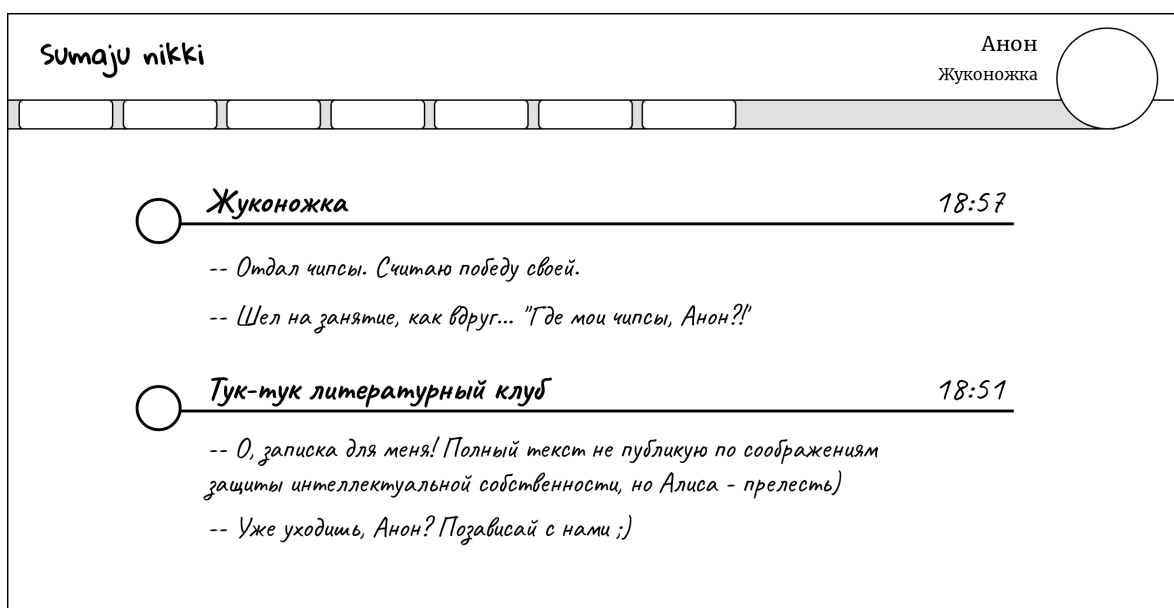


Рис. 2: Дневник персонажа

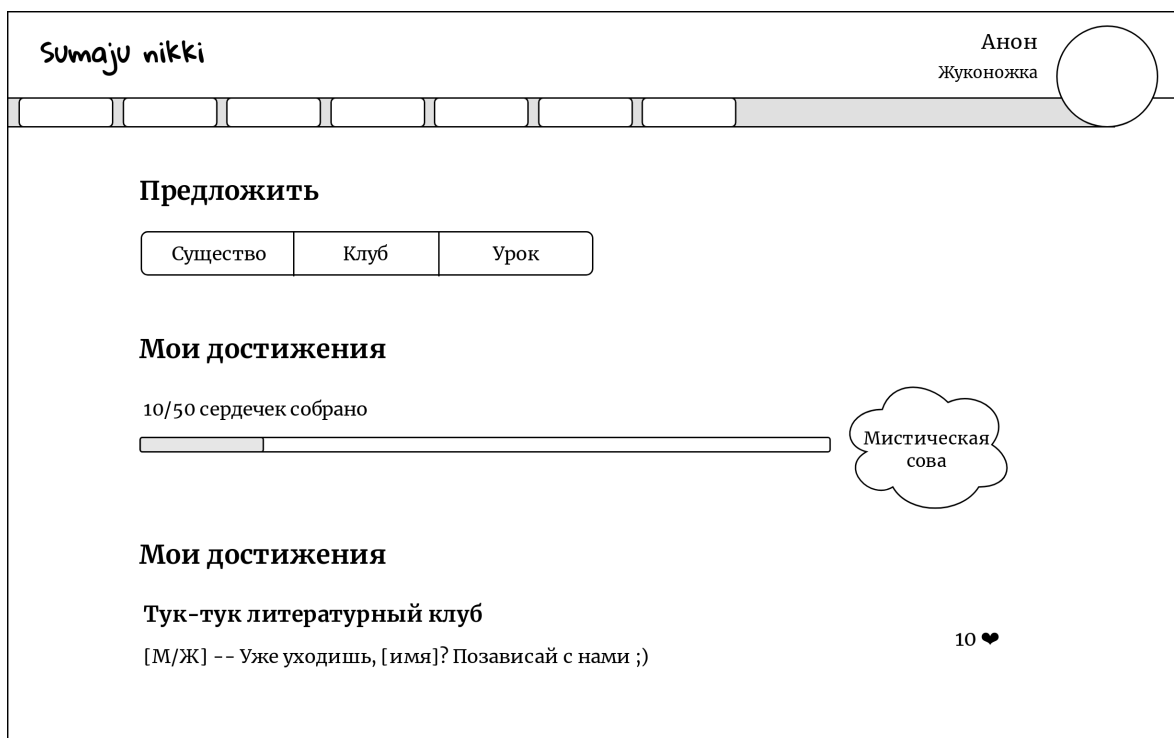


Рис. 3: Пользовательские предложения