ФГАОУ ВО «Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики»

Кафедра вычислительной техники

КУРСОВАЯ РАБОТА

Программирование интернет-приложений

Третий этап. Уровень бизнес-логики

Саржевский Иван Анатольевич

Лабушев Тимофей Михайлович

Группа P3202

Санкт-Петербург

2019 г.

Введение

Ключевой особенностью разработанной игровой платформы является отсутствие необходимости в действиях со стороны пользователя для развития игры.

Под развитием игры подразумевается изменение базовых характеристик персонажа — уровеня и показателя здоровья — а также ресурсов, привязанных к персонажу.

Выделяются следующие виды ресурсов, за обновлением которых может следить пользователь:

- *посещения уроков*, необходимые для перехода на следующий уровень
- *навыки обращения с существами*, полученные в бою с ними
- *заклинания*, отражающие боевые качества персонажа и его способность выстраивать взамоотношения с другими персонажами
- *взаимоотношения* с персонажами, на создание и поддержание которых пользователи влияют напрямую
- *совы*, используемые пользователем для влияния на персонажа

Игровой процесс разделен на дискретные фазы и происходит пошагово. По окончании фазы происходит обновление характеристик и ресурсов, затем устанавливается следующая фаза и ее длительность.

На протяжении фазы пользователю демонстрируется *заметка* о текущих действиях персонажа. Некоторые заметки сохраняются в *дневник*, который также считается обновляемым ресурсом, отражающим развитие игры.

Каждая заметка ассоциируется с определенной фазой (а также с определенным уроком для фазы посещения урока, определенным клубом для фазы посещения клуба, определенным существом для фазы боя.

Заметки предлагаются пользователями. Понравившиеся заметки могут быть отмечены «сердечком», количество которых отображается публично и служит для поощрения талантливых создателей.

Процесс обновления

Для каждого персонажа доступна информация о его текущей фазе и времени перехода к следующей. Периодически база данных опрашивается о персонажах с истекшим временем фазы, и для каждого из них вызывается обработчик. Несколько персонажей могут обрабатываться одновременно. Данная логика реализуется актором `actors.GameProgressionActor`.

Обработчиком является сервис `services.GameProgressionService`, который определяет все возможные переходы между фазами и их эффекты, делегируя конкретные обновления специализированным сервисам и объектам доступа к данным *DAO*. Обработка выполняется в пределах одной транзакции, что предотвращает частичные изменения.

Синхронизация обновлений с клиентом

Для своевременного отражения обновлений между клиентом и сервером устанавливается двусторонний обмен данными через протокол WebSockets. Между идентификатором пользователя и его соединениями устанавливается связь, что позволяет отправлять сообщения нескольким активным устройствам одного пользователя. Данный функционал реализован актором `actors.SocketMessengerActor`.

Сообщение о смене фазы формируется сервисом `services.StageService` и содержит:

- новые характеристики персонажа
- заметку о новой фазе, показываемую пользователю
- длительность фазы и прошедшее время, позволяющее рассчитать прогресс хода
- идентификаторы измененных ресурсов

Сравнивая идентификаторы измененных ресурсов, клиент может запросить новые данные путем обращения к REST API платформы.

Влияние пользователя на игровой процесс

В ходе игры пользователю становятся доступны *совы*, обладающие различными эффектами. Выделяется два вида сов: применяемые немедленно (*immediate*) и участвующие в подсчете определенных фаз (*non-immediate*).

Совы, применяемые немедленно

Совы данного вида позволяют реализовать любое воздействие на текущее состояние персонажа, поскольку они реализуются как отдельные классы со следующим контрактом:

```
abstract class Owl {
  def apply(studentId: Long, payload: JsValue): Either[String, String]
}
```

`studentId` является одновременно идентификатором пользователя и его персонажа, `payload` является JSON значением, принятым от клиента. Возвращаемое значение — сообщение об успешном применении или ошибке, показываемое пользователю.

Классы `Owl` могут принимать DAO как зависимости. Они инстанциируются и вызываются сервисом `services.OwlService`.

Ограничением *immediate* сов является то, что они не могут быть учтены в процессе обновления фазы, так как принимают параметры от клиента и модифицируют глобальное состояние персонажа.

Фазовые совы

Для реализации различных бонусов во время подсчета результата фазы применяются совы, у которых отсутствует конкретная имплементация. Они передаются логике обновления как уникальные строковые идентификаторы для использования в условных выражениях.

Модерация пользовательских предложений

Предложенные пользователями заметки должны пройти модерацию, прежде чем они будут видны в системе. Этот функционал реализуется Telegram-ботом, находящемся в заранее определенном чате с администраторами платформы.

При взаимодействии с серверным приложением бот авторизуется, используя заранее определенный токен. Следующие пути реализуют основной функционал модерации предложений:

`GET /bot/creatures/unapproved`: возвращает первое в очереди модерации существо Возвращаемые значения (HTTP-статус):

- **200**, JSON-объект, содержащий имя существа и асоциированные с ним заметки
- **404**, если очередь модерации существ пуста