# «Сказка» Documentation

*Выпуск 0.3.26*

Разработчики «Сказки»

июн. 18, 2022

# Содержание

«Сказка» — браузерная многопользовательская текстовая ролевая песочница с действующими самостоятельно героями.

**Сайт игры** https://the-tale.org

**Огранизациия** https://github.com/the-tale

**Репозиторий** https://github.com/the-tale/the-tale

**Форум** https://the-tale.org/forum/subcategories/28

**Разработчики** *контакты*

Глава 1

# Введение

## 1.1 Что такое «Сказка»

На текущий момент «Сказка» — это два тесно переплетённых и дополняющих друг друга проекта:

- браузерная многопользовательская текстовая ролевая песочница с оригинальным геймплеем.
- magic punk вселенная, которая разрабатывается параллельно с игрой.

Оба проекта разрабатываются в тесном сотрудничестве с игроками, и вы можете легко примкнуть к любому из них.

«Сказка» развивается с 2011 года.

### 1.1.1 Игра

Начинаясь как чистая Zero Player Game игра постепенно обрастала элементами песочницы и на текущий момент кратко определить её жанр сложно. Неизменным остаётся нетребовательность геймплея ко времени игроков.

Можно выделить несколько ключевых особенностей игры:

- непрямое управление героями — они действуют самостоятельно, в большинстве случаев игроки могут влиять на их поступки только косвенно;
- состояние мира игры полностью зависит от действий игроков и их героев;
- всё что происходит в игре описываются **художественным** текстом, вид которого зависит от множества параметров;

В «Сказку» можно играть как в тамагочи (выращивая своего героя) или как в «политическую песочницу» (кооперируясь с другими игроками для получения нужного эффекта на мир).

### 1.1.2 Вселенная

Вселенная появилась в процессе разработки игры, но уже давно вышла за её пределы. Далеко не всё из лора реализовано в игре.

Есть несколько ключевых отличий сказочной вселенной от множества других:

- инженерный подход к использованию магии;
- Пандора — мир, в котором происходит действие, — является гигантским магическим механизмом;
- бессмертие (включая воскрешение и вечную молодость) является «рядовым» явлением, хотя и редким.

Наработано большое количество хужожественных описаний существ и артефактов существующих в Пандоре.

Кроме этого мы работаем над каноничным описанием мира.

**Мы заинтересованы в выходе новых игр (или других произведений) по нашей вселенной** и готовы к сотрудничеству.

## 1.2 Цели проекта

На протяжении жизни игры цели её разработки меняли свой приоритет, но их список остаётся неизменным. На текущий момент по приоритету он выглядит так:

1. Создать игру, мир которой будет восприниматься игроками как живой.
2. Дать каждому игроку его уникальное бесконечное приключение — собственную сказку.
3. Создать оригинальную вселенную, которая станет отдельным культурным явлением.
4. Получить площадку для экспериментов в области разработки ПО и геймдизайна.
5. Создать площадку для тематического творчества игроков.
6. Заработать денег, если игра станет достаточно популярной.

## 1.3 Технологии

На текущий момент игра представляет собой монолитный проект, который постепенно переписывается на микросервисы (в целях уменьшения связанности). Большая часть функциональности написана самостоятельно.

**Основные ЯП** Python3 (backend) и JavaScript (frontend)

**Фреймворки** Django для сайта, aiohttp для (новых) фоновых сервисов

**База данных** PostgreSQL

**Кеш** Redis

**Очереди** RabbitMQ

Разработка ведётся с помощью виртуальной машины, поднимаемой с помощью Vagrant.

## 1.4 Краткая история

- 2011 год — Елецкий Алексей (Tiendil) начал разработку игры (по вечерам).
- 2012 год — Tiendil уволился и начал работать над игрой полный рабочий день.
- 2012 год — Первые игровые концепции сформированы в сотрудничестве с Александром Титовым (Gizoom).
- 2012 год — К разработке примкнули геймдизайнеры Саша и Лена Дедковы (спасибо Gizom-у).
- 2015 год — у Tiendil-а кончились деньги :-D он устроился на работу и продолжает заниматься игрой по вечерам.
- ... — разработка продолжается.

## 1.5 Команда

### 1.5.1 Елецкий Алексей (Tiendil)

**Занятие** программирование, но занимается всем.

**Блог** https://tiendil.org

**Skype** Tiendil

**E-mail** a.eletsky@gmail.com

### 1.5.2 Саша и Лена Дедковы

**Занятие** геймдизайн и разработка вселенной.

Глава 2

# Планы

На этой странице представлены примерные **стратегические** планы развития игры. Планы примерные и порядок реализации «фич» периодически существенно пересматривается. Здесь представлены ключевые, наиболее существенные моменты игры — небольшие изменения и дополнения могут происходить спонтанно и в любой момент.

## 2.1 Будет реализовано

### 2.1.1 версии 0.4.*

- Доработка и развитие гильдейских эмиссаров.
- Гильдии NPC с особыми механиками (некроманты, демонопоклонники, Серый Плащи, etc.).
- Чат (Discord).
- Вики вместо путеводителя.
- Вики вместо технической документации.
- Вики вместо фольклора (как расширение функциональности текущего фольклора).
- Замена форума на сторонний движок.
- Галерея (как расширение функциональности фольклора).
- Общение с героем (ответы «молитвы»), модифицирующее его характер.
- Переработка характера героя (предпочтений и черт) в пользу общения с героем.
- Отказ от кнопки «помочь» в пользу использования карт судьбы и общения с героем.
- Переработка системы прокачки и способностей героев (в сторону EVE online).
- Временные эффекты на героях (проклятья, благословения, травмы).

### 2.1.2 далёкое будущее

- Встречи героев.
- Подробное описание внешности героя (зависящее от его одежды, характера, достижений);
- Появление воды на карте (реки, озёра и прочее);
- Подземелья.
- Развитие системы заданий.
- Группы связанных фраз в лингвистике.
- Деревни: постройка, развитие, жители, задания связанные с ними — отдельные поселения, которые могут развивать как отдельные игроки, так и гильдии или группы друзей.
- События на карте: логова опасных монстров, ярмарки, нашествия некромантов, погодные катаклизмы, подземелья.
- Создание гильдиями легендарных артефактов.
- Групповые задания.
- Характер героя — особенности — уникальные черты героев, делающие их «особенными» (параметры не влияющие на силу героев: специальные задания, особые фразочки; примеры: кладоискатель, филателист, романтик, дипломат);

## 2.2 Уже реализовано

### 2.2.1 версии 0.3.*

- **[сделано]** гильдии - базовый механизм для объединения игроков, закрытый гильдейский форум;
- **[сделано]** достижения и рассказы-награды за них;
- **[сделано]** усложнение системы артефактов - поломка/починка, покупка с учётом полезности, градация качества, особые эффекты;
- **[сделано]** карты судьбы - универсальный способ влиять на мир игры и героев;
- **[сделано]** спутники героев;
- **[сделано]** рынок - внутриигровая торговля картами судьбы;
- **[сделано]** характер для персонажей в городах, влияющий на мастерство, получение влияния городами и персонажами, выдаваемые ими задания и на все остальные моменты игры. Каждый персонаж станет действительно уникальным, получит друзей и врагов среди таких же персонажей;
- **[сделано]** улучшение интерфейса рынка;
- **[сделано]** специальные правила объединения карт;
- **[сделано]** автоматическое вознаграждение авторов лингвистики;
- **[сделано]** редактирование дорог игроками, улучшенный поиск пути героями;
- **[сделано]** совершенствование управления гильдиями;
- **[сделано]** эмиссары гильдий;

Глава 3

# Разработка

Все обсуждения, связанные с разработкой, можно вести на форуме игры.

Важные новости для разработчиков публикуются в отдельной теме на форуме игры.

## 3.1 Установка и запуск

Окружение игры организуется с помощью Docker.

В рамках этого документа предполагается, что вы знакомы с базовыми концепциями Docker.

Инструкция проверялась под Ubuntu, должна работать для любого популярного linux дистрибутива.

Инструкция должна работать для Windows под WSL — Windows Subsystem for Linux, но не проверялась.

**Мы с радостью примем pull requests с уточнениями как документации, так и кода.**

### 3.1.1 Термины и соглашения

Далее будут встречаться некоторые понятия, которые лучше пояснить отдельно.

- **старые сервисы** — сервисы игры, разработанные на оригинальном движке. Используют Django, общаются через очереди, реализованы в коде сайта. Отвечают за логику игры.
- **новые сервисы** — часть игры, переписанная на микросервисную архитектуру. Использует aiohttp, общается по http, реализованы отдельными модулями.

### 3.1.2 Установка (для разработки)

Установите Docker и Docker Compose.

Clone with HTTPS

Clone with SSH

```
git clone https://github.com/Tiendil/the-tale.git
```

```
git clone git@github.com:the-tale/the-tale.git
```

```
cd ./the-tale

# Все следующие команды необходимо выполнять из корня проекта.

# При необходимости переключаем репозитории в ветке develop.
# git checkout develop

# При необходимости определяем переменную окружения, отвечающую за тип создаваемого окружения.
# Про типы окружений будет рассказано подробнее.
# Значение по умолчанию: stage
# export TT_ENV=stage

# Обратите внимание на использование обёртки над docker-compose.
# Все операции необходимо выполнять через них.

# Собираем все контейнеры.
./bin/docker_compose_build_all.sh

######################################################
# Производим первоначальную настройку состояния игры
######################################################

# Создаём файл с кастомными настройками игры.
# Запомните его, он будет полезен в процессе разработки.
# В данном случае мы явно включаем в игре режим отладки.
# Файл settings_local.py добавлен в .gitignore и не сохраняется в репозиторий.
echo "DEBUG = True" > ./docker/the_tale/settings_local.py

# подготовливаем игру к запуску
./bin/before_first_start.sh

# В том числе команда создаём суперпользователя с параметрами по-умолчанию.
# ПОМЕНЯЙТЕ ИХ, если планируете давать доступ к игре другим людям.
# - ник: superuser
# - почта: superuser@example.com
# - пароль: 111111
```

### 3.1.3 Запуск игры

```
./bin/tt_game_start

# Теперь игра должна быть доступна по адресу "localhost".

# Остановить игру можно командой
# ./bin/tt_game_stop

# После остановки игры, можно остановить инфраструктуру
# ./bin/docker_compose.sh down
```

### 3.1.4 Типы окружений

Игра может запускаться в нескольких режимах, управляемых переменной окружения `TT_ENV`:

- `prod` — окружение для запуска проекта в боевом режиме.
- `stage` — окружение для запуска на тестовых серверах или на машине разработчика.
- `tests` — окружение, оптимизированное для прогона тестов.

В большинстве случаев вам будет хватать `stage`.

Окружение `test` использует оптимизированную конфигурацию контейнеров для ускорения прогонки тестов:

- PostgreSQL запускается на tmpfs то есть держит абсолютно все данные в памяти. Убедитесь, что у вас достаточно RAM.

### 3.1.5 Docker Compose

Вся конфигурация контейнеров находится в директории `./docker`.

Базовую конфигурацию можно найти в файлах :

- `./docker/docker-compose.build.yml` — параметры сборки контейнеров.
- `./docker/docker-compose.temlates.yml` — общие параметры сервисов. В этот конфиги используется шаблонизация yaml.
- `./docker/docker-compose.base.yml` — персонализированные параметры сервисов, общие для всех окружений.
- `./docker/docker-compose.$TT_ENV.yml` — конфиги конкретных окружений.

Итоговое окружение получается с помощью переопределения нескольких конфигов. Это делается в обёртках над docker compose (см. `./bin/docker_compose.sh` и прочие скрипты).

Сервисы разбиты на несколько профилей:

- `core` — ключевые сервисы инфраструктуры: база, кэш, веб-сервер, etc.
- `services` — все новые сервисы. От их доступности зависит работоспособность сайта.
- `workers` — все старые сервисы. От их доступности зависят некоторая функциональность сайта. Например, регистрация.
- `site` — сервис сайта.
- `utils` — вспомогательные контейнеры для запуска утилит.

- `tasks-managers` — сервис менеджера периодических задач (а-ля cron).
- `tasks` — сервисы периодических задач, по сервису на задачу, управляются tasks manager-ом.

Сервисы без указанных профилей — сервисы инфраструктуры. В большинстве случаев все они должны быть запущены.

### 3.1.6 Опциональные репозитории

Часть проектов, родившихся в рамках разработки, доросли до стабильной версии и хостятся на pypi.org.

Если необходимо делать правки в них (например, добавить новую функциональность), их следует клонировать по аналогии с обязательными репозиториями и вручную поставить из исходников в нужные контейнеры.

Репозитории:

- генератор имён персонажей: https://github.com/Tiendil/pynames
- продвинутые перечисления: https://github.com/Tiendil/rels
- генератор текста: https://github.com/Tiendil/utg
- умные импорты для Python: https://github.com/Tiendil/smart-imports
- генератор карты: https://github.com/the-tale/deworld
- генератор заданий: https://github.com/the-tale/questgen

### 3.1.7 Нюансы

Настройка форума проводится через админку Django.

Права пользователей также настраиваются через админку Django.

Админка Django доступна по адресу `https://localhost/admin`

После настройки, в базе игры не будет фраз для лингвистики, вместо них будут отображаться заглушки, описывающие тип фразы и её параметры. Фразы необходимо добавлять руками. Вы можете написать нам и мы вышлем дамп таблиц лингвистики для **личного пользования**.

В окружении разработчика используется самоподписанный сертификат, поэтому браузеры будут сообщать о «небезопасном соединении». Это нормально (для окружения разработчика). Если вы хотите избавиться от этого предупреждения, импортируйте сертификат к себе в систему или поправьте конфиги nginx.

### 3.1.8 Разработка

Процесс разработки с помощью Docker ещё не устоялся и может поменяться. На текущий момент:

- Код из репозитория монтируется в соответствующие контейнеры.
- Изменения в коде будут появляться в контейнерах, но запущенные сервисы не будут перезапускаться.
- Если вы ведёте активную разработку одного из сервисов, рекомендуем запустить `bash` в соответствующем контейнере и запускать тесты и сервис вручную оттуда.

Во всех контейнерах, где необходимо, есть ряд утилит с именами `tt_*`. Они закрывают большинство нужд разработки.

Пример:

```
# обратите внимание на параметры
# --name - должен быть установлен в имя сервиса в docker-compose.base.yml, иначе другие сервисы не␣
↪найдут его в сети.
# --entrypoint - указываем контейнеру запустить bash вместо команды по-умолчанию.
./bin/docker_compose.sh run utils-site bash

# запускаем какие-то команды

# стартуем сайт в обычном режиме
# tt_site -b 0.0.0.0:80 -w 4

# стартуем сайт в режиме разработчика
# tt_django runserver 0.0.0.0:80

# запускаем тесты
# tt_django test the_tale.portal
```

### 3.1.9 Запуск тестов

Тесты сервисов:

```
./bin/docker_compose.sh run tt-diary tt_run_tests
```

Главные тесты игры:

```
# Выключаем всё
./bin/tt_infrastructure_stop

# Запускаем только необходимые для тестов сервисы.
./bin/tt_infrastructure_start
./bin/docker_compose.sh --profile services up -d

./bin/docker_compose.sh run utils-site tt_django utils_run_tests
```

**Предупреждение: Тесты игры идут очень долго.** На моей машине около часа.

Небольшая часть тестов может сообщить об ошибках (обычно до 5) — это «нормально» — следствие большой вариативности логики игры. Стабилизация таких тестов — хорошая задача для нового разработчика.

### 3.1.10 Бэкапы

Контейнер `utils-postgresql` предоставляет экспериментальную функциональность по созданию бэкапов, загрузке их на `amazon s3`, выгрузке и восстановлению.

## 3.2 Как присоединиться?

**Спасибо что интересуетесь нашим проектом!** Мы (в особенности Tiendil) обещаем всеми силами помогать вам.

Присоединиться в разработке можно разными способами, все они привычны для open source проектов:

- Помогите улучшить нашу докумнетацию — её не бывает много.
- Напишите новый юнит-тест — это поможет познакомиться с интересующей вас частью игры и сделает её немного надёжнее.
- Подумайте над небольшим улучшением интерфейса — мы не можем уделять ему достаточно внимания, поэтому там всегда есть место для улучшения.
- Выберите одну из задач для новичков.
- Из игры постепенно выделяются *дочерние проекты*, которые могут пригодиться другим разработчикам. Вы можете помочь в их разработке.
- Если вы опытный разработчик, можете взяться за одну из *больших задач*.

Среди меток каждой задачи вы обязательно найдёте указание на её сложность, тип и компонент игры, к которому она относится.

Кроме этого вы можете помочь игре как политик, автор лингвистики или фольклора.

Прежде чем браться за какую-либо задачу:

- Научитесь запускать *локальную версию игры*.
- Напишите Tiendil-у (сделать это можно на сайте игры или по любому из контактов на tiendil.org). Он подробно расскажет про все нюансы выбранной задачи.
- Прочтите *про организацию процесса разработки*.

## 3.3 Процесс разработки

### 3.3.1 Общий подход

Разработка игры идёт в духе «инкрементального прототипирования», что предполагает прохождение каждой «фичи» через последовательность прототипов, совершенствующих её реализацию. Не делается попыток с первого раза написать самый правильный и самый крутой код. Как следствие:

- новая «фича» реализуется самым простым и понятным образом, без переусложнений и оптимизаций;
- по мере развития игры (и анализа её работы), логика «фичи» уточняется и совершенствуется;
- если «фича» оказывается достаточно независимой, она выделяется в отдельную библиотеку.

Следствием такого подхода является постоянный рефакторинг проекта.

В случае крупных инфраструктурных изменений их допустимо не применять сразу ко всему проекту, а только к тем частям, над которыми идёт работа в настоящий момент. Старый код вычищается тогда, когда появляется необходимость изменения компонента, в котором он находится.

### 3.3.2 Тесты

Весь важный код покрывается тестами.

Весь не важный код покрывается тестами по возможности.

### 3.3.3 Миграции

Миграции схемы базы создаются на основе Django ORM (даже если сам Django больше нигде в сервисе не используется).

### 3.3.4 Работа с версиями

Разработу ведём по git-flow:

- http://nvie.com/posts/a-successful-git-branching-model/
- https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow

Master ветка всегда содержит в себе текущую рабочую версию проекта (ту, которая сейчас запущена на сайте).

Работа над новой версией ведётся в ветке develop. По завершении работы, делается тег и изменения подливаются в master.

**Всю работу надо делать в ветке develop или её ветке. Изменения применять к ней же.**

### 3.3.5 Требования к изменениям

Предлагаемые вами изменения в обязательном порядке должны:

- быть в виде одного патча (pull request должен содержать 1 коммит);
- быть покрыты тестами;
- не использовать устаревшую функциональность;
- применяться к ветке develop (не master!), как и положено по git-flow.

Желательно, чтобы они сопровождались документацией.

### 3.3.6 Описание импортов в исходниках на Python

В проекте используется smart_imports. Это упрощает редактирование и анализ кода, плюс, упорядочивает имена сущностей.

Документацию по smart_imports можно найти на странице проекта.

Конфигурация библиотеки (правила импорта) находится в файлах smart_imports.json Например, для основного сайта игры — the_tale — конфигурацию можно найти в файле ./src/the_tale/the_tale/smart_imports.json

В целом, правила импорта можно описать следующими эвристиками (по приоритету):

- Если имя переменной является стандартным (сторонним) пакетом, оно импортируется напрямую.
- Если имя переменной находится в списке предопределённых значений, оно импортируется по собственному правилу. Правила определены в smart_imports.json.
- Если имя переменной совпадает с именем локального модуля (из той же директории), то импортируется модуль.
- Если имя переменной начинается с имени одного из базовых модулей (например, places_logic, где places — стандартный модуль), то импортируется его подмодуль, определённый после префикса (например, places_logic is the_tale.game.places.logic). Префиксы определены в smart_imports.json.
- если имя переменной является модулем верхнего уровня стандартной библиотеки, то он импортируется.

## 3.4 Дочерние проекты

Из игры, по мере готовности, выделяются небольшие проекты, которые можно использовать отдельно от неё.

Все проекты небольшие и имеют чётко выделенную функциональность. По каждому из них есть интересные задачи.

### 3.4.1 smart_imports — умные импорты для Python

Smart imports, анализируя код на Python, определяют что куда и под каким именем необходимо импортировать (и импортируют). Позволяют избавиться от длинных списков импортов и навести порядок в именах.

**pypi** https://pypi.org/project/smart_imports/

**github** https://github.com/Tiendil/smart-imports

### 3.4.2 pynames — генератор имён

Генератор имён персонажей. Может создавать имена на русском и английском. На русском выдаёт все формы имени.

**pypi** https://pypi.org/project/Pynames/

**github** https://github.com/the-tale/pynames

### 3.4.3 utg — генератор русского текста

Шаблонный генератор русского текста с учётом зависимостей слов и их форм. Используется для генерации всего текста в игре.

**pypi** https://pypi.org/project/UTG/

**github** https://github.com/the-tale/utg

### 3.4.4 deworld — генератор карты для игры

Библиотека моделирует изменения ландшафта в игре. Учитываются:

- высота
- температура у земли
- температура воздуха
- влажность почвы
- влажность воздуха
- качество почвы
- направление ветра
- растительность

Каждому параметру соответствует отдельный слой данных, который пересчитывается каджый шаг на основе данных всех слоёв предыдущего шага.

Игровые объекты могут влиять на изменение параметров, устанавливая «точки влияния», которые изменяют параметры слоя в заданной области.

За счёт подобного подхода удаётся получить «реалистичное» изменение ланшафта не сходящееся к стабильному состоянию (так как влияние игровых объектов постоянно меняется, мешая сходимости).

**github** https://github.com/the-tale/deworld

### 3.4.5 questgen — генератор квестов

Генератор цепочек квестов в игре с учётом контекста (какой NPC с каким дружит или враждует, etc). Принцип работы описан на хабре.

**habrahabr** https://habrahabr.ru/post/201680/

**github** https://github.com/the-tale/questgen

## 3.5 Направления разработки

Планов по развитию «Сказки» у нас больше, чем свободных рук, поэтому всегда есть крупные задачи, ожидающие своей реализации.

Эта страница посвящена как раз таким задачам и будет интересна опытным разработчикам с большим количеством свободного времени.

Если вы хотите активно участвовать в разработке игры — эти задачи для вас. Подробно про каждую из них может рассказать Tiendil.

**Наиболее важной задаче сейчас является рефакторинг проекта с монолитного на набор микросервисов.**

Планируемая архитектура описана на *отдельной странице*.

### 3.5.1 Рефакторинг

Как сказано в *описании процесса разработки*, игра находится в состоянии непрерывного рефакторинга. У нас всегда найдётся место, в котором нужно навести порядок :-)

- Оптимизация тестов — в проекте более 4000 автоматических тестов, работу которых можно ускорить и упростить.
- Эксперименты с организацией кода привели к неудобной системе обёрток над Django view, которую надо заменить на более удобную.
- Те же эксперименты переусложнили работу с объектами игры.
- Бразуерный интерфейс игры реализован на старых технологиях, необходимо его актуализировать (или переписать полностью).

Отдельно хочется отметить идущий рефакторинг проекта с монолитного на набор микросервисов. У этой активности две цели:

- уменьшение связанности логики игры (чтобы упростить внесение изменений);
- получение независимых от логики игры переиспользуемых сервисов.

Большую часть подсистем игры предполагается выделять в отдельные сервисы, поэтому сейчас можно выбрать для работы наиболее интересный (и нужный для ваших проектов) кусок функциональности.

### 3.5.2 Существующая функциональность

Часть фунциональности игры реализована в минимально необходимом виде, поэтому её можно (и нужно) развивать.

- Форум (нужны система модерации, доработка статистики, исправление ошибок, приведение кода в порядок).
- Гильдии (нужны звания, взаимодействие с форумом, продвинутые настройки, статистика, развитие геймплея).
- Игровая статистика (увеличение собираемых параметров, их анализ).
- Лингвистика (упрощение интерфейса и логики работы).
- На сайте игры необходимо внедрить семантическую разметку.
- Необходимо развивать функциональность генерации и отображения карты (нужны библиотеки GUI для разных платформ, улучшение интерфейса и расширение функциональности).

### 3.5.3 Новая функциональность

Новая функцональность реализуется в виде отдельных сервисов.

Список текущих задач по новым сервисам.

Кроме этого, мы привествуем прототипирование новых фич веб-интерфейса.

### 3.5.4 Проекты-спутники

Вы можете присоединиться к одному из *проектов игроков* или начать свой.

## 3.6 Архитектура



По историческим причинам «Сказка» — монолитный проект с сильной связанностью, это начало усложнять реализацию новых фич. Поэтому сейчас он переписывается на микросервисы. Пока идёт активная фаза рефакторинга, сервисы выделяются логически, но не выносятся из основного репозитория (ложаться в соседние каталоги). Когда крупные изменения закончатся, сервисы будут разнесены в разные репозитории.

На этой странице описана архитектура игры, какой она должна стать.

### 3.6.1 Базовая структура

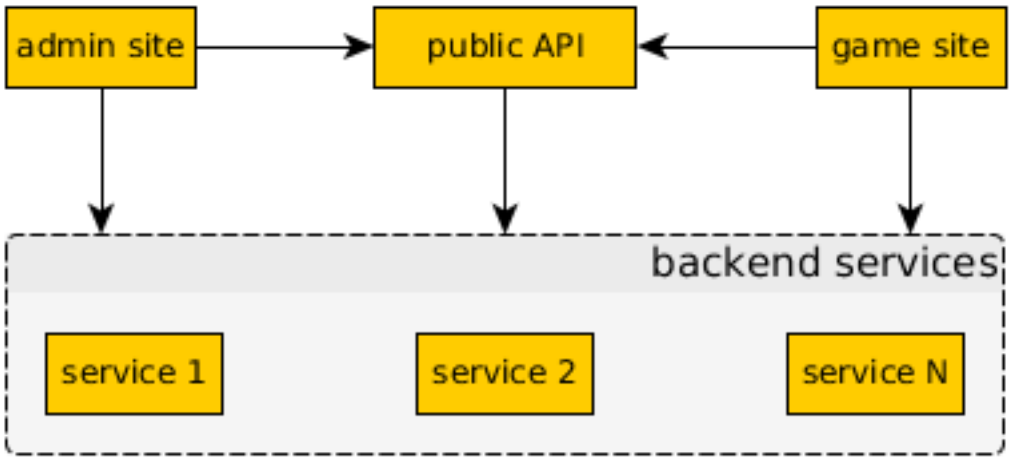


Рис. 1: Стрелками указано использование сервисами друг друга.

Все сервисы игры можно разделить на две группы:

- **frontend** — сервисы, с которыми так или иначе взаимодействуют пользователи.
- **backend** — сервисы, не доступные напрямую для пользователей.

**Frontend** сервисы проверяют права доступа и используют **backend** сервисы для получения данных и инициации изменений. **Backend** сервисы реализуют специализированные куски логики и не делают неспецифичных проверок (например, не проверяют права доступа).

**Backend** сервисы недоступны извне, доверяют всем клиентам и не реализуют GUI (предоставляют только API).

Коммуникация сервисов происходит через передачу сообщений в формате Protocol Buffers 3 по протоколу HTTP.

### 3.6.2 Существующие сервисы

Текущий список сервисов.

- the_tale — сайт игры (сейчас содержит и основную логику);
- tt_diary — хранилище сообщений в дневнике героя;
- tt_market — рынок;
- tt_personal_messages — личные сообщения;
- tt_storage — хранилище предметов игрока (в рамках Сказки, карты судьбы).
- tt_bank — хранилище игровой валюты с поддержкой транзакций.
- tt_timers — управление таймерами (например, на получение карты), по истечении таймера делает заданные http запрос.
- tt_impacts — учёт численных «воздействий» одних сущностей на другие. Например, влияние героев на Мастеров.

- tt_events_log — хранилище истории игровых событий с привязкой к «тегам» и фильтром по ним.
- tt_effects — хранилище эффектов, действующих на объекты игры.
- tt_matchmaker — простой сервис поиска соперника для pvp и отслеживания активных боёв.
- tt_properties — хранилище свойств объектов, нужных в частных случаях логики (например, флаг «разрешить приглашать игрока в клан»).
- tt_uniquer — сервис выделения уникальных целочисленных идентификаторов для строк.

### the_tale

Вся функциональность игры: как логика, так и инфраструктура (форум, регистрация, лингквистика, etc).

По мере рефакторинга в этом компоненте останется только сам сайт.

### tt_diary

Сервис хранит последние сообщения в дневнике героя и управляет их количеством.

### tt_market

Реализация рынка торговли предметами.

**Не привязан к логике игры, может быть использован отдельно.**

Функциональность:

- выставление предмета на продажу;
- возвращение предмета с продажи;
- покупка предмета;
- получение информации о выставленных предметах;
- получение списка выставленных предметов конкретного типа.
- логирование всех операций.

### tt_personal_messages

Личные сообщения игроков.

**Не привязан к логике игры, может быть использован отдельно.**

Функциональность:

- создание сообщений;
- удаление сообщений;
- получение списка сообщений (входящих/исходящих, по фильтру);
- поиск сообщений;

### tt_storage

Предметы игрока. Используется для хранения карт судьбы.

**Не привязан к логике игры, может быть использован отдельно.**

Функциональность:

- помещение предмета в хранилище;
- удаление предмета из хранилища;
- передача предмета другому игроку;
- перемещение предмета между «карманами» хранилища;
- логирование всех операций.

### tt_bank

Хранилище игровой валюты с поддержкой транзакций.

**Не привязан к логике игры, может быть использован отдельно.**

Функциональность:

- болучение баланса аккаунта в каждой валюте;
- транзакции: начал, окончание, откат;

### tt_timers

Управление таймерами (например, на получение карты или постройки здания), по истечении таймера делает заданные http запрос.

**Не привязан к логике игры, может быть использован отдельно.**

Функциональность:

- создание таймера с заданными параметрами;
- изменение скорости выполнения таймера;
- http запрос по истечению таймера;
- автопродление таймера.

### tt_impacts

Учёт численных «воздействий» одних сущностей на другие. Например, влияние героев на Мастеров, учёт известности героев, учёт голосов за запись в Книге Судеб, etc.

**Не привязан к логике игры, может быть использован отдельно.**

Функциональность:

- добавление воздействия;
- получение списка последних воздействий;
- получение суммарных воздействий на перечисленные сущности;
- получение суммарных воздействий от указанной сущнсти на сущности переданных типов;

- получение рейтинга сущностей, больше всего повлиявших на указанные сущности;
- скалирование итоговых сумм воздействий (например, чтобы уменьшать их со временем).

### tt_events_log

Хранилище истории игровых событий с привязкой к «тегам» и фильтром по ним. Позволяет организовать отображение такую функиональность как «последние события в гильдии».

**Не привязан к логике игры, может быть использован отдельно.**

Функциональность:

- добавить событие;
- получить события по фильтру;
- получить последние события;

### tt_effects

Хранилище эффектов, действующих на объекты игры. Предполагается, что эффекты могут вешаться на объекты из разных мест игры. Чтобы унифицировать этот процесс, все они шлются в этот сервис, из которого уже выбираются соответствующими объектами (по таймеру или по команде).

**Не привязан к логике игры, может быть использован отдельно.**

Функциональность:

- зарегистрировать эффект;
- удалить эффект;
- изменить эффект;
- получить список эффектов;

### tt_matchmaker

Простой сервис поиска соперника для pvp и отслеживания активных боёв.

**Не привязан к логике игры, может быть использован отдельно.**

Функциональность:

- создать запрос на битву;
- отменить запрос на битву;
- принять запрос на битву;
- создать битву;
- получить список запросов на битву;
- получить текущую статисику (количество битв, количество запросов);
- завершить битву;
- получить список участников битвы;

#### tt_properties

Хранилище свойств объектов, нужных в частных случаях логики (например, флаг «разрешить приглашать игрока в клан»). Позволяет избежать раздувания основных объектов игры (аккаунт, герой, гильдия, etc).

**Не привязан к логике игры, может быть использован отдельно.**

Функциональность:

- установить свойства;
- получить свойства;

#### tt_uniquer

Сервис выделения (и хранения) уникальных целочисленных идентификаторов для строк.

**Не привязан к логике игры, может быть использован отдельно.**

Функциональность:

- получить уникальный идентификатор по строке;

### 3.6.3 Планируемые сервисы

- HTTP API 2.0;
- Галерея изображений;
- Фольклор;
- Сервис генерации информеров;
- Сервис выдачи краткой информации по объектам игры;
- Форум;
- Регистрация (плюс поддержка авторизации через популярные OAuth провайдеры);
- Кланы;
- Достижения;
- Друзья;
- Приём платежей от XSolla;
- Сервис рассылки сообщений на почту игроков;
- Статистика;
- Игровая логика (разобьётся на несколько сервисов);
- Карта — ландшафт;
- Карта — логика (города, дороги, etc).

## 3.7 Устройство сервисов

Все backend сервисы однообразны:

- именуются как tt_*;
- устанавливаются и запускаются под собственным пользователем;
- подключаются к отдельной базе;
- общаются по API, описанному с помощью protocol buffers 3;
- используют одинаковую структуру файлов исходного кода;
- используют Django миграции для изменения схемы и данных в базе.

Сервисы, по возможности, не привязываются к логике игры и делаются реиспользуемыми.

### 3.7.1 Зависимости

Каждый сервис зависит от следующих компонентов:

- tt_web - общий минифреймворк
- tt_protocol - описание структур данных api в формате protocol buffers

### 3.7.2 Структура файлов

```
├── MANIFEST.in                # включаемые в Python пакет файлы
├── setup.py                   # описание Python пакета
├── tt_service                 # исходный код сервиса
│   ├── migrations             # Django миграции
│   ├── fixtures               # дополнительные данные для сервиса
│   ├── handlers.py            # обработчики HTTP API
│   ├── models.py              # модели Django
│   ├── objects.py             # внутренние структур данных
│   ├── logic.py               # внутренняя логики
│   ├── operations.py          # операции с базой данных и другими внешними сервисами
│   ├── protobuf.py            # конвертация из/в структур данных protocol buffers во внутренние␣
↪структуры данных
│   ├── relations.py           # перечисления (enums)
│   ├── service.py             # код запуска сервиса
│   ├── settings.py            # настройки Django
│   ├── conf.py                # константы и прочие параметры
│   └── tests                  # тесты
│       ├── fixtures           # дополнительные данные для тестов
│       │   └── config.json    # конфигурация сервиса для тестов
│       ├── helpers.py         # вспомогательный код
│       └── test_something.py  # тесты
```

Если в одном из файлов образуется слишком много кода (например, в operations.py) он преобразуется в подмодуль (operations/...).

### 3.7.3 Архитектура

Код сервиса можно разделить на несколько слоёв:

- интерфейс HTTP API: разбор запроса, проверка параметров, инициирование операции;
- внутренняя логика: операции над внутренними структурами данных, без обращения к любым внешним сущностям;
- внешняя логика: взаимодействие со внешними сущностями (например, с базой данных, другими сервисами)

Каждый из файлов, описанных ранее, можно явно отнести к одной из частей.

### 3.7.4 Тестирование

Весь код сервиса должен быть покрыть тестами (в отличии от кода сайта игры, который не всегда можно разумными усилиями качественно проверить).

Для каждой сущности (кроме тривиальных) внутренней и внешней логики пишутся тесты на каждый вариант поведения.

Код обработчиков HTTP API покрывается тестами:

- для проверки верификации и преобразования данных (из запроса во внутренний формат и из них в формат ответа);
- для проверки всей последовательности операций.

Покрывать тестами каждый вариант поведения обработчиков HTTP API не надо, так как в них не реализуется логика. Мы считаем, что логику проверяют тесты логики.

## 3.8 Руководства

### 3.8.1 Конфигурация работы с почтой

На этой странице будет представлено подробное руководство по настройке работы игры с почтой. Данный момент не существенен для разработки, но критичен для публикации игры, поскольку многие активности (регистрация, изменение пароля, нотификации) требуют отправки почты.

Конфигурацию работы игры с почтой можно разделить на 2 части: настройка отправки писем и настройка соединения с почтовым сервером.

Поскольку вопрос почтовых рассылок сложный, мы приветствуем всяческие дополнения в это руководство.

#### Отправка писем

За отправку писем отвечает отдельный фоновый рабочий `post_service`. Часть его настроек можно найти в `the_tale.post_service.conf` и переопределить в `settings_local.py` дописав к их именам `POST_SERVICE_`. Эти настройки снабжены соответствующими комментариями, которые повторяться тут не будут.

Для включения отправки писем необходимо установить

```
POST_SERVICE_ENABLE_MESSAGE_SENDER = True
```

И установить значение `allowed` в таблице `settings` (через админку Django) по ключу равному значению `SETTINGS_ALLOWED_KEY`.

Кроме того, в `settings_local.py` необходимо установить следующие параметры

- `SERVER_EMAIL` — почтовый адрес от которого по умолчанию будут отправляться письма.
- `EMAIL_NOREPLY` — почта, которая будет писаться в письмах, на которые игроки не должны отвечать, тут можно указать длинное значение вроде `u'«Сказка» <no-reply@the-tale.org>'`
- `EMAIL_SUPPORT` — почта службы поддержки, тут можно указать длинное значение вроде `u'«Сказка» <support@the-tale.org>'`
- `EMAIL_SUPPORT_SHORT` — короткий адрес службы поддержки (только сама почта, без вставки имени и прочего)

#### Настройка соединения с почтовым сервером

Отправлять почту мы можем как через собственный почтовый сервис (например, настроив Postfix) либо через один из существующих (например, через GMail).

В каждом случае есть свои нюансы, касающиеся массовых рассылок и попадания под спам фильтры. Но в начале проще использовать существующий сервис (при запуске, Сказка использовала GMail, потом перешла на Postfix).

Соединение с почтовым сервисом настраивается стандартным для Django способом.

Для собственного сервиса будет достаточно указать следующие настройки

- `EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'`
- `EMAIL_HOST_USER` — пользователь сервиса с правами отправки почты
- `EMAIL_HOST_PASSWORD` — пароль пользователя

Для использования сторонних сервисов, возможно, надо будет добавить несколько других параметров (см. в конфигурации Django и этот пост)

#### Отправка писем во время разработки

Для разработки игры обычно нет необходимости настраивать отправку почты, поскольку для большинства случаев она не нужна.

Поэтому отправку почты можно настроить в каталог на файловой системе, например, так:

- `EMAIL_BACKEND = 'django.core.mail.backends.filebased.EmailBackend'`
- `EMAIL_FILE_PATH = '/tmp/emails'`

### 3.8.2 Карта мира и работа с ней

На этой странице в общих чертах описывается техническое устройство карты мира.

Карта изменяется раз в час под влиянием параметров, задаваемых действиями игроков, и своего предыдущего состояния.

Карта всегда представляется клеточным полем.

Весь код (кроме низкоуровневого моделирования ландшафта) можно найти:

- Python: модуль the_tale.game.map.

- JavaScript модуль map2.js

### Уровни формирования карты

Формирование карты можно разбить на несколько уровней, на каждом из которых она обрабатывается особым образом.

- Моделирование базовых параметров мира.
- Выделение биомов — узнаваемых типов ландшафта, которые лучше всего соответствуют параметрам клетки карты (горы, джунгли, равнина, etc.).
- Фомирование схемы тайлового отображения карты — именно эта схема отдаётся браузеру и по ней рисуется карта.
- Отображение карты браузером.

### Моделирование параметров мира

Моделирование карты осуществляется отдельной библиотекой *deworld*:

- github: https://github.com/the-tale/deworld
- *немного документации*

Моделируется несколько базовых параметров (высота, температура, растительность, etc.) с учётом влияния игровых объектов.

На выходе этого этапа карта представлена набором слоёв, с нормированными величинами каждого параметра (например, температуры).

Игровые объекты влияют на эти параметры, задавая «точки влияния». Кажая точка описывается:

- координатами клетки на карте;
- описанием области, на которую распространяются изменения;
- слоем, который она изменяет;
- величиной и правилами изменения параметра слоя.

Описание «точек влияния» можно найти в исходниках игры.

Также, по этим данным формируется текстовое описание клетки. Исходный код.

### Выделение биомов

Поскольку реализовать отображение карты для всех возможных сочетаний параметров сложно, у нас выделено несколько выделяющихся типов ландшафта, которые мы называем биомами.

Для каждой клетки карты бием выбирается по следующему алгоритму:

- Пространство значений каждого параметра бьётся на интервалы (например, влажность, измеряемая от 0 до 1 бьётся, на 10 интервалов с шагом 0.1).
- Для каждого сочетания биома с интервалом значения в балансе игры прописано количество «очков», которое биом получит, если значение параметра в клетке попадает в интервал.
- Для каждого биома считается сумма очков всех параметров клетки.
- Биомом клетки выбирается биом с максимальным количеством очков.

Файл с очками биомов

### Фомирование схемы отображения карты

Данных о биомах и объектах карты уже хватило бы, чтобы нарисовать её в клиентах игры. Но, чтобы упростить реализацию новых клиентов, мы делаем предобработку данных карты.

В итоге клиенту возвращается двухмерный массив. В каждой клетке массива находится список идентификаторов спрайтов, которые нужно отобразить и угол их поворота. Спрайты указаны в порядке отрисовки.

Получение карты и её формат описаны *документации нашего API*.

### Отображение карты клиентом

Клиенту остаётся получить инструкцию по отрисовке карты и исполнить её.

- Python код с отрисовкой карыт
- Javascript код с отрисовко карты

### Обновление карты на сервере

Для обновления карты на сервере есть отдельная команда.

Глава 4

# Внешнее API

Игра предоставляет программный интерфейс для разработки игровых клиентов и иных вспомогательных приложений. При реализации любого рода сторонних приложений настоятельно рекомендуется пользоваться именно этим интерфейсом. В случае, если требуется дополнительная функциональность — смело обращайтесь к разработчикам.

Обсудить API можно на форуме.

## 4.1 Введение

### 4.1.1 Принципы взаимодействия

Взаимодействие происходит по протоколу HTTPS (все запросы по HTTP будут перенаправлены на аналогичный HTTPS адрес).

Ответ возвращается в формате JSON.

Сессии реализуются через передачу cookie с именем sessionid; в случае, если в запросе cookie не указана, сервер создаст новую сессию и вернёт соответствующее значение cookie.

---

**Примечание:** Для защиты от CSRF (так как фунционал браузерной версии также построен на API), каждый POST запрос должен иметь cookie с именем csrftoken И либо POST параметр csrfmiddlewaretoken, либо заголовок X-CSRFToken, установленные в значение этой cookie. Значение для csrftoken cookie можно установить случайное (либо значение cookie будет установлено при 1-ом запросе к API). Значение csrftoken должно быть строкой из 64 символов (0-9,a-z).

Кроме того, необходимо передавать заголовок *Referer: https://the-tale.org/*.

---

Примеры запросов на логин к локальному серверу с использованием curl:

```
curl --insecure --referer "https://the-tale.org/" -b "sessionid=kwc2ngq02dilu56ti76nj21z18wzaghe;␣
↪csrftoken=wxiefxk7i6kvkUeyi4jU2xO0B96RwvJc" -d "email=email@gmail.com&password=11111"  -H "X-
↪CSRFToken: wxiefxk7i6kvkUeyi4jU2xO0B96RwvJc" "https://local.the-tale/accounts/auth/api/login?api_
↪version=1.0&api_client=SASS-asas"

curl --insecure --referer "https://the-tale.org/" -b "sessionid=kwc2ngq02dilu56ti76nj21z18wzaghe;␣
↪csrftoken=wxiefxk7i6kvkUeyi4jU2xO0B96RwvJc" -d "email=email@gmail.com&password=111111&
↪csrfmiddlewaretoken=wxiefxk7i6kvUeyi4jU2xO0B96RwvJc" "https://local.the-tale/accounts/auth/api/
↪login?api_version=1.0&api_client=SASS-asas"
```

### 4.1.2 Формат запроса

Адрес каждого метода имеет следующий формат:

```
https://the-tale.org/<адрес>?api_version=<версия>&api_client=<id клиента>&<параметры>
```

**адрес** путь к методу

**версия** версия метода (в стандартном формате через точку: 1.0, 1.1 и etc.)

**id клиента** идентификатор клиентского приложения

**параметры** аргументы, необходимые методу

### 4.1.3 Версионность

Каждый метод может иметь одну или несколько версий.

Старшая версия считается основной, остальные — устаревшими.

При обращении к устаревшей версии метода, в ответе будет присутствовать соответствующий флаг-индикатор.

В рамках одной версии метода гарантируется сохранение формата вызова.

В рамках одной версии метода гарантируется недеструктивное сохранение поведения (сторонние эффекты, коды возвращаемых ошибок).

В рамках одной версии метода возможно недеструктивное изменение формата возвращаемых данных (могут быть добавлены новые поля, без изменения структуры существовавших ранее).

Устаревшие версии методов будут удаляться из игры по прошествии времени, достаточного для перехода активных проектов на новые версии методов.

### 4.1.4 Идентификатор клиента

Передача идентификатора клиента необходима для сбора разработчиками статистики и возможности проведения разного рода «магических» действий в случае его неадекватного поведения.

Формат идентификатора: <идентификатор программы>-<версия программы>, в идентификаторе должен быть ровно 1 дефис.

Значение идентификатора выбирается пользователями API, регистрировать идентификатор нигде не требуется.

На наш взгляд, лучшим решением будет составление идентификатора из аббревиатуры названия клиента и версии (или даты) его сборки.

### 4.1.5 Неблокирующие операции

Некоторые запросы требуют длительного времени на свою обработку, поэтому для работы с ними реализован специальный механизм, позволяющий не блокировать выполнение клиента (и сервера) дожидаясь ответа на запрос. В описании таких операций это свойство отмечено отдельно.

Принцип работы неблокирующей операции:

1. запрос клиента вызывает выполнение фоновой задачи на сервере;
2. сервер возвращает ответ с информацией, по которой можно проверять статус выполнения задачи (со статусом «processing»);
3. клиент периодически проверяет статус выполнения (желательно, не чаще раза в секунду), пока не получит сообщение о её завершении (или об ошибке);
4. после этого операция считается выполненой.

### 4.1.6 Разница между «входом в игру» и «авторизацией в игре»

Для получения и изменения информации конкретного пользователя в API предусмотрено два метода: «вход в игру» и «авторизация в игре».

1. «Вход в игру» требует от пользователя ввода логина и пароля. Это позволяет получать доступ к любым данным и производить любые действия от имени пользователя;
2. «Авторизация в игре» не требует ввода логина и пароля. Она производится через выдачу пользователем разрешения на сайте игры. Приложение, получившее разрешение, не будет иметь доступ к наиболее важным данным и операциям пользователя (профилю, магазину и так далее).

Для выбора метода рекомендуется пользоваться следующим правилом: если Вы делаете приложение для себя или нескольких друзей (доверяющих вам), то можно использовать метод «вход в игру», во всех остальных случаях используйте «авторизацию в игре».

### 4.1.7 Формат ответа

Ответ на любой корректный запрос в случае корректной работы сервера возвращается с кодом 200.

```
{
  "depricated": true,                  // поле устанавливается, при обращении к устаревшей версии␣
↪метода

  "status": "ok"|"error"|"processing", // ok - запрос обработан корректно
                                       // error - произошла ошибка
                                       // processing - запрошена неблокирующая операция, идёт␣
↪обработка запроса
  "code": "error.code",                // строка - уникальный код ошибки (присутствует только в␣
↪случае ошибки)
  "error": "сообщение",                // сообщение об ошибке для пользователя (присутствует␣
↪только в случае ошибки)

  // если ошибка в данных, вводимых пользователем, вместо "error" в ответ вставляется "errors" с␣
↪перечислением
  // идентификатор поля - имя параметра, в котором передавался ввод пользователя
  "errors": {"идентификатор поля": ["сообщение 1", "сообщение 2"]},

  "status_url": "url"                  // адрес проверки статуса неблокирующей операции (формат␣
↪статуса такой же)
```

(continues on next page)

(продолжение с предыдущей страницы)

```
  "data": {},                           // запрошенные данные, в случае корректного выполнения␣
↪запроса
                                        // дополнительная информация об ошибке, в случае␣
↪некорректного выполнения запроса
}
```

## 4.2 Методы

### 4.2.1 Базовая информация

Получение базовой информации о текущих параметрах игры и некоторых других данных.

**адрес** /api/info/

**http-метод** GET

**версии** 1.1

**параметры** нет

**ошибки** нет

Формат данных в ответе.

```
{
  "static_content": "абсолютный url",     // базовый абсолютный путь к статическим игровым данным␣
↪(например, картинкам)
  "game_version": "текущая.версия.игры", // текущая версия игры
  "turn_delta": <целое>,                  // задержка между ходами в секундах
  "account_id": <целое>|null,             // идентификатор аккаунта, если пользователь вошёл в игру,
↪ иначе null
  "account_name": <строка>|null           // имя пользователя, если он вошёл в игру, иначе null
}
```

Абсолютные адреса возвращаются без указания протокола: `//path/to/entity`

### 4.2.2 Авторизация в игре

Авторизация приложения для проведения операций от имени пользователя. Приложению не будут доступны «критические» операции и данные (связанные с профилем пользователя, магазином и так далее).

**адрес** /accounts/third-party/tokens/api/request-authorisation

**http-метод** POST

**версии** 1.0

**параметры**

- POST: application_name — короткое название приложения (например, его название в google play).
- POST: application_info — краткое описание информации об устройстве пользователя (чтобы пользователь мог понять откуда пришёл запрос).

- POST: application_description — описание приложения (без html разметки)

**ошибки** нет

Формат данных в ответе.

```
{
  "authorisation_page": <url>, // адрес, на который необходимо направить пользователя для
↪подтверждения авторизации
}
```

Алгоритм авторизации:

1. приложение делает запрос к этому методу;
2. в ответе приходит ссылка, по которой надо направить пользователя, и устанавливается значение cookie с именем sessionid, которое и является идентификатором сессии пользователя;
3. пользователь переходит по ссылке, на странице у него спрашивают разрешение на доступ к своим данным для данного приложения;
4. приложение (по таймеру или по нажатию кнопки пользователем) делает запрос к методу получения состояния авторизации;
5. ответ метода будет содержать информацию о статусе авторизации и о пользователе;
6. после успешной авторизации с API можно работать точно так же, как и после обычного входа в игру.

**Значение cookie с именем sessionid будет изменено сервером при первом запросе к нему после того как пользователь подтвердил разрешение** (скорее всего это будет запрос состояния авторизации). После этого необходимо будет использовать установленное сервером значение cookie.

Запрос авторизации не хранится вечно, гарантируется его доступность в течение 10 минут после создания.

В случае обращения к закрытой функциональности (профилю пользователя, магазину и так далее) в ответ вернётся ошибка `third_party.access_restricted`.

При «выходе из игры» разрешение, выданное приложению, удаляется.

Рекомендации:

- **Функцию выхода из игры рекомендуется реализовывать. Также рекомендуется выходить из игры при любой нобходимости релогина.**
- Не указывайте версию своей программы ни в одном из параметров запроса, т.к. они сохраняются на сервере и не будут изменяться при изменении версии.
- Делайте подробное описание. Расскажите подробно о функциональности программы.

### 4.2.3 Состояние авторизации

Метод возвращает состояние авторизации для текущей сессии. Обычно вызывается после запроса авторизации.

**адрес** /accounts/third-party/tokens/api/authorisation-state

**http-метод** GET

**версии** 1.0

**параметры** нет

**ошибки** нет

Формат данных в ответе.

```
{
  "next_url": "относительный url",  // адрес, переданный при вызове метода или "/"
  "account_id": <целое число>,      // идентификатор аккаунта
  "account_name": <строка>,         // имя игрока
  "session_expire_at": <timestamp>, // время окончания сессии пользователя
  "state": <целое число>            // состояние авторизации, см. в списке типов
}
```

### 4.2.4 Вход в игру

Вход в игру. **Используйте этот метод только если разрабатываете приложение для себя и друзей**. В остальных случаях пользуйтесь «авторизацией в игре».

**адрес** /accounts/auth/api/login

**http-метод** POST

**версии** 1.0

**параметры**

- GET: next_url — вернётся в ответе метода в случае успешного входа, по умолчанию равен «/»
- POST: email — email адрес пользователя
- POST: password — пароль пользователя
- POST: remember — если флаг указан, сессия игрока будет сохранена на длительное время

**ошибки**

- accounts.auth.login.wrong_credentials — неверный логин или пароль
- accounts.auth.login.form_errors — ошибка(-и) в заполнении полей

Формат данных в ответе.

```
{
  "next_url": "относительный url", // адрес, переданный при вызове метода или "/"
  "account_id": <целое число>,     // идентификатор аккаунта
  "account_name": <строка>,        // имя игрока
  "session_expire_at": <timestamp> // время окончания сессии пользователя
}
```

При успешном выполнении запроса, будет установлено значение cookie с именем sessionid, которая и является идентификатором сессии пользователя.

В случае, если от имени не вошедшего в игру пользователя будет произведён запрос функциональности, доступнойй только авторизованным пользователям, API вернёт ошибку с кодом `common.login_required` (см. секцию с описанием общих ошибок).

### 4.2.5 Выход из игры

Выйти из игры.

**адрес** /accounts/auth/api/logout

**http-метод** POST

**версии** 1.0

**параметры** нет

**ошибки** нет

Формат данных в ответе.

```
{
}
```

### 4.2.6 Информация об игроке

Получить информацию об игроке.

**адрес** /accounts/<account>/api/show

**http-метод** GET

**версии** 1.0

**параметры**

- URL account — идентификатор игрока

**ошибки**

- account.wrong_value — аккаунт с таким идентификатором не найден

Формат данных в ответе.

```
{
  "id": <целое число>,           // идентификатор игрока
  "registered": true|false,      // маркер завершения регистрации
  "name": "строка",              // имя игрока
  "hero_id": <целое число>,      // идентификатор героя
  "places_history": [            // список истории помощи городам
    "place": {                   // город
      "id": <целое число>,       // идентификатор города
      "name": "строка"           // название города
    },
    "count": <целое число>       // количество фактов помощи
  ],
  "might": <дробное число>,      // могущество
  "achievements": <целое число>, // очки достижений
  "collections": <целое число>,  // количество предметов в коллекции
  "referrals": <целое число>,    // количество последователей (рефералов)
  "ratings": {                                // рейтинги
    "строка": {                               // идентификатор рейтинга:
      "name": "строка",                       // название рейтинга: иинформация о рейтинге
      "place": <целое число>,                 // место
      "value": <целое число>|<дробное число>  // величина рейтингового значения
    }
```

(continues on next page)

(продолжение с предыдущей страницы)

```
  },
  "permissions": {                // права на выполнение различных операций
    "can_affect_game": true|false // оказывает ли влияние на игру
  },
  "description": "строка"         // описание игока, введённое им сами (в формате html)
  "clan": null | {                // информация о гильдии, null, если в гильдии не состоит
     "id": <целое>,               // идентификатор
     "abbr": "строка",            // аббревиатура
     "name": "строка"             // название
  }
}
```

### 4.2.7 Количество новых сообщений

Получить количество новых сообщений для игрока.

**адрес** /accounts/messages/api/new-messages-number

**http-метод** GET

**версии** 0.1

**параметры** нет

**ошибки** нет

Формат данных в ответе.

```
{
  "number": <целое число> // количество новых сообщений
}
```

### 4.2.8 Информация об игре/герое

Информация о текущем ходе и герое

**адрес** /game/api/info

**http-метод** GET

**версии** 1.10

**параметры**

- GET: account — идентификатор аккаунта
- GET: client_turns — номера ходов, по отношению к которым можно вернуть сокращённую информацию о герое (только изменённые с этого времени поля).

**ошибки** нет

Если параметр account не будет указан, то вернётся информация об игре текущего пользователя, а на запрос от неавторизованного пользователя — общая информация об игре (без информации об аккаунте и герое).

Часть информации в ответе является личной и доступна только залогиненному игроку, для остальных на её месте будет валидная с точки зрения формата заглушка. Такая информация обозначена следующим образом: **[личная информация]**.

Если информация о герое устаревшая `is_old == true`, то следует повторить запрос через несколько секунд (но лучше не злоупотреблять).

Полный ответ имеет большой размер, поэтому реализован следующий механизм его сжатия:

- в параметре client_turns можно передать список номеров ходов (через запятую), для которых на клиенте есть полная информация;
- если сервер сможет, в ответе он вернёт только изменившуюся информацию о герое;
- сокращению подвергается только информация в `hero_info`;
- сокращение происходит удалением неизменившихся полей `hero_info` (только на верхнем уровне, без рекурсии);
- чтобы получить полную информацию, скопируйте недостающие поля из закэшированной на стороне клиента информации для хода, указанного в `.account.hero.patch_turn`;
- сервер не гарантирует, что вернёт сокращённую информацию;
- сервер может вернуть патч для любого из переданных в `client_turns` ходов;
- имеет смысл в параметре `client_turns` передавать последние 2-3 хода;
- обратите внимание, сжатие ответа применяется и к информации о противнике в PvP! Поэтому первый запрос при PvP всегда должен требовать полную информацию.

Формат данных в ответе.

```
{
  "mode": "pve"|"pvp",              // режим героя
  "turn": {                         // информация о номере хода
    "number": <целое число>,        // номер хода
    "verbose_date": "строка",       // дата для игроков (в мире Сказки)
    "verbose_time": "строка"        // время для игроков (в мире Сказки)
  },
  "game_state": <целое число>,      // состояние игры (остановлена/запущена, см. в описании API)
  "map_version": "строка",          // версия актуальной карты игры
  "account": <account_info>|null,   // информация о запрашиваемом аккаунте и герое
  "enemy": <account_info>|null      // информация о противнике, если идёт pvp сражение
}

<account_info> = {
  "new_messages": <целое число>, // количество личных сообщений
  "id": <целое число>,           // идентификатор аккаунта
  "last_visit": <timestamp>,     // примерное время последнего посещения игры
  "is_own": true|false,          // информация о собственном герое или о чужом
  "is_old": true|false,          // информация устаревшая или нет
  "hero": <hero_info>            // информация о герое
}

<hero_info> = {
  "patch_turn": null|<целое число>,  // номер хода, для которого возвращается патч или null, если␣
↪информация полная

  "equipment":{                      // экипировка героя, словарь <идентификатор типа экипировки,␣
↪информация об артефакте>
    "<целое число>": <artifact_info> // идентификатор типа экипировки: информация об артефакте
  },

  "companion": <companion_info>|null,// информация о спутнике
```

(continues on next page)

(продолжение с предыдущей страницы)

```
  "bag":{                              // содержимое рюкзака, словарь <внутренний идентификатор␣
↪предмета, описание> ()
    "<целое число>": <artifact_info> // идентификатор слота: информация об артефакте
  },

  "base":{                                 // базовые параметры героя
    "experience": <целое число>,           // текущий опыт
    "race": <целое число>,                 // раса
    "health": <целое число>,               // здоровье
    "name": "строка",                      // имя героя
    "level": <целое число>,                // уровень героя
    "gender": <целое число>,               // пол
    "experience_to_level": <целое число>, // абсолютное количество опыта до следующего уровня
    "max_health": <целое число>,           // максимальное количество здоровья
    "destiny_points": <целое число>        // сколько способностей сейчас может выбрать
    "money": <целое число>,                // количество денег у героя
    "alive": true|false,                   // жив герой или мёртв
  },

  "secondary":{                              // второстепенные параметры
    "max_bag_size": <целое число>,           // максимальный размер рюкзака
    "power": [<целое число>, <целое число>], // физическая сила, магическая сила
    "move_speed": <дробное число>,           // скорость движения
    "loot_items_count": <целое число>,       // количество лута в рюкзаке
    "initiative": <дробное число>            // инициатива героя
  },

  "diary": "строка",       // версия дневника героя, если она изменилась, необходимо перезапросить␣
↪дневни

  "messages":[             // сообщения из журнала
    [                      // запись в задании
      <timestamp>,         // timestamp создания сообщения
      "строка",            // текстовое описание времени в игре
      "строка",            // текст
      <целое число>|null,  // идентификатор типа фразы, найти идентификатор типа фразы можно в␣
↪адресе страницы лингвистики с фразами этого типа
      {"строка": "строка"} // словарь соотношения переменных и их значений (ВНИМАНИЕ! перечень␣
↪переменных может изменяться без изменения версии этого метода)
    ]
  ],

  "habits": { // черты
    "строка": {              // идентификатор черты
      "verbose": "строка",   // текущее текстовое значение черты для игрока (название характера)
      "raw": <дробное число> // текущее числовое значение черты
    }
  },

  "quests": {     // информация о заданиях
    "quests": [   // список глобальных заданий
      {
        "line": [ // список «базовых» заданий (цепочка последовательных заданий)
           {
             "type": "строка",            // тип задания
```

(continues on next page)

(продолжение с предыдущей страницы)

```
            "uid":  "строка",             // уникальный идентификатор задания
            "name": "строка",             // название задания
            "action": "строка",           // описание текущего действия героя в задании
            "choice": "строка"|null,      // текущий выбор героя в задании
            "choice_alternatives": [      // альтернативные выборы
              [
                "строка",                 // уникальный идентификатор выбора
                "строка"                  // текстовое описание выбора
              ]
            ],
            "experience": <целое число>, // количество опыта за задание
            "power": <целое число>,       // количество влияния за задание
            "actors": [                   // список «актёров», участвующих в задании
              [
                "строка",                 // название актёра
                <целое число>,            // тип актёра (список типов приведён в описании API)
                <quest_actor_info>        // данные, специфичные для конкретного типа актёра
              ]
            ]
          }
        ]
      }
    ]
},

"action":{                      // текущее действие
  "percents": <дробное число>, // процент выполнения
  "description": "строка",      // описание
  "info_link": "url"|null       // ссылка на доп. информацию
  "type": <целое число>         // идентификатор типа действия
  "data": null|<словарь>        // дополнительная информация о действии или null, если такой нет
},

"position":{                        // позиция героя на клеточной карте
  "x": <дробное число>,             // координата x
  "y": <дробное число>,             // координата y
  "dx": <дробное число>,            // направление взгляда по x
  "dy": <дробное число>,            // направленеи взгляда по y
},

"permissions": {                        // права на выполнение различных операций
  "can_participate_in_pvp": true|false, // может ли участвовать в pvp
  "can_repair_building": true|false,    // может ли чинить здания
},

"might": {                                    // могущество игрока
  "value": <дробное число>,                   // величина
  "pvp_effectiveness_bonus": <дробное число>, // бонус к эффективности в pvp от могущества
  "politics_power": <дробное число>            // бонус к политическому влиянию героя
},

"id": <целое число>,                          // идентификатор
"actual_on_turn": <целое число>,              // данные на какой ход предоставлены

"sprite": <целое число>  // идентификатор спрайта, которым отображается герой
}
```

(continues on next page)

(продолжение с предыдущей страницы)

```
<quest_actor_info> = <quest_actor_place_info>|<quest_actor_person_info>|<quest_actor_spending_info>

<quest_actor_place_info> = { // информация о городе
  "id": <целое числое>,      // идентификатор
  "name": "строка"           // название города
}

<quest_actor_person_info> = {     // информация о жителе города
    "id": <целое числое>          // идентификатор
    "name": "строка",             // имя
    "race": <целое числое>,       // раса
    "gender": <целое числое>,     // пол
    "profession": <целое числое>, // профессия
    "mastery_verbose": "строка",  // профессия
    "place": <целое числое>       // идентификатор города
}

<quest_actor_spending_info> = { // информация о целях накопления
  "goal": "строка"              // описание цели накопления
}

<artifact_info> = {                             // информация об артефакте
    "name": "строка",                           // название
    "power": [<целое число>, <целое число>],    // сила [физическая, магическая]
    "type": <целое число>,                      // тип
    "integrity": [<целое число>, <целое число>], // целостность [текущая, максимальная]
    "rarity": <целое число>,                    // редкость
    "effect": <целое число>,                    // тип эффекта на артефакте
    "special_effect": <целое число>,            // тип особого свойства артефакта (эффекта,␣
↪который действует независимо от редкости)
    "preference_rating": <дробное число>,       // «полезность» артефакта с точки зрения героя
    "equipped": true|false,                     // может ли быть экипирован
    "id": <целое число>                         // уникальный идентификатор рода артефакта
}

<companion_info> = {                        // информация о спутнике героя
    "type": <целое число>,                  // тип спутника
    "name": "строка",                       // название/имя спутника
    "health": <целое число>,                // текущее здоровье
    "max_health": <целое число>,            // максимальное здоровье
    "experience": <целое число>,            // текущий опыт слаженности
    "experience_to_level": <целое число>,   // опыта до следующего уровня слаженности
    "coherence": <целое число>,             // текущая слаженность
    "real_coherence": <целое число>         // полная слаженность (без учёта ограничений на␣
↪максимум слаженности)
}
```

### 4.2.9 Дневник героя

Информация о дневнике героя

**адрес** /game/api/diary

**http-метод** GET

**версии** 1.0

**параметры** нет

**ошибки** нет

Формат данных в ответе.

```
{
    "version": <целое>,                         // версия дневника
    "messages": [                               // список последних сообщений в дневнике
       {                                        // запись в дневнике
          "timestamp": <timestamp>,             // timestamp создания сообщения
          "game_time": "строка",                // текстовое описание времени в игре
          "game_date": "строка",                // текстовое описание даты в игре
          "message": "строка",                  // текст
          "type": <целое число>|null,           // идентификатор типа фразы, найти идентификатор␣
↪типа фразы можно в адресе страницы лингвистики с фразами этого типа
          "variables": {"строка": "строка"},    // словарь соотношения переменных и их значений␣
↪(ВНИМАНИЕ! перечень переменных может изменяться без изменения версии этого метода)
          "position": "строка"                  // текстовое описание места, где герой находился во␣
↪время создания записи
        }
    ]
}
```

### 4.2.10 Выбор в задании

Изменение пути выполнения задания героем

**адрес** /game/quests/api/choose/

**http-метод** POST

**версии** 1.0

**параметры**

- GET: option_uid — уникальный идентификатор выбора в задании (получается с информацией о состоянии игры)

**ошибки** нет

Метод является «неблокирующей операцией» (см. документацию), формат ответа соответствует ответу для всех «неблокирующих операций».

### 4.2.11 Карты: получить новые карты

Получить все накопившиеся новые карты.

**адрес** /game/cards/api/receive

**http-метод** POST

**версии** 1.0

**параметры** нет

**ошибки** нет

При завершении операции возвращается дополнительная инфрмация:

```
{
  "cards": [
      <card_info>, // описание полученной карты
      ...]
}

<card_info> = {               // информация о карте в колоде игрока
    "name": "строка",         // название
    "type": <целое число>,    // тип
    "full_type": "строка",    // полный тип карты (с учётом эффектов)
    "rarity": <целое число>,  // редкость карты
    "uid": "строка",          // уникальный идентификатор в колоде игрока
    "auction": true|false,    // может быть продана на рынке
    "in_storage": true|false  // находится ли карты в хранилище или в руке
}
```

### 4.2.12 Карты: превратить

Превратить карты из колоды игрока.

**адрес** /game/cards/api/combine

**http-метод** POST

**версии** 3.0

**параметры**

- POST: card — идентификатор карты, участвующей в трансформации, может быть несколько

**ошибки**

- cards.combine.wrong_cards — указанные карты нельзя превращать

Формат данных в ответе.

```
{
  "message": "строка",  // описаниеb результата в формате html
  "cards": [
      <card_info>,      // описание полученной карты, формат см. в описании метода получения новой␣
↪карты
      // ...
      ]
}
```

### 4.2.13 Карты: использовать

Использовать карту из колоды игрока.

**адрес** /game/cards/api/use

**http-метод** POST

**версии** 2.0

**параметры**

- GET: card — уникальный идентификатор карты в калоде
- POST: value — параметр использования карты, если у карты есть параметр, обычно это идентификатор объекта (Мастера, города, etc)
- POST: name — название гильдии для карты создания гильдии
- POST: abbr — аббревиатура гильдии для карты создания гильдии

**ошибки**

- cards.use.form_errors — ошибка в одном из POST параметров

Метод является «неблокирующей операцией» (см. документацию), формат ответа соответствует ответу для всех «неблокирующих операций».

### 4.2.14 Карты: перечень карт игрока

Возвращает список всех карт игрока.

**адрес** /game/cards/api/get-cards

**http-метод** GET

**версии** 2.0

**параметры** нет

**ошибки** нет

Формат данных в ответе.

```
{
  "cards": [
      <card_info>,       // описание полученной карты, формат см. в описании метода получения␣
↪новой карты
      ...
   ],
  "new_cards": <целое число>,       // количество новых карт, которые можно получить

  "new_card_timer": {               // таймер, отсчитывающий время получения следующей карт
      "speed": <дробное число>,     // скорость накопления ресурсов (в секунду)
      "border": <дробное число>,    // сколько «ресурсов» надо накопить, чтобы сработал таймер
      "resources": <дробное число>, // количество ресурсов, которые накопились к моменту resources_
↪at
      "resources_at": <timestamp>,  // время, на которое указано количество «ресурсов» в resources
      "finish_at": <timestamp>      // время срабатывания таймера
  }
}
```

### 4.2.15 Карты: переместить в хранилище

Перемещает карты в хранилище.

**адрес** /game/cards/api/move-to-storage

**http-метод** POST

**версии** 2.0

**параметры**

- POST: card — идентификатор перемещаемой карты, может быть несколько

**ошибки**

- card.wrong_value — указанные карты нельзя превращать

Формат данных в ответе.

```
{
}
```

### 4.2.16 Карты: переместить в руку

Перемещает карты в руку.

**адрес** /game/cards/api/move-to-hand

**http-метод** POST

**версии** 2.0

**параметры**

- POST: card — идентификатор перемещаемой карты, может быть несколько

**ошибки**

- card.wrong_value — указанные карты нельзя превращать

Формат данных в ответе.

```
{
}
```

### 4.2.17 Города: перечень всех городов

Получить перечень всех городов с их основными параметрами

**адрес** /game/places/api/list

**http-метод** GET

**версии** 1.1

**параметры** нет

**ошибки** нет

Формат данных в ответе.

```
{
  "places": {                 // перечень всех городов
    "<целое число>": {        // идентификатор города: информация о нём
      "id": <целое число>,    // идентификатор города
      "name": "строка",       // название города
      "frontier": true|false, // находится ли город на фронтире
      "position": { "x": <целое число>,   // координаты города на карте
                    "y": <целое число> }, // (могут меняться при изменении размера карты!)
      "size": <целое число>,              // размер города
      "specialization": <целое число>     // идентификатор специализации
    }
  }
}
```

### 4.2.18 Города: подробная информация о городе

Подробная информация о конкретном городе

**адрес** /game/places/<place>/api/show

**http-метод** GET

**версии** 2.3

**параметры**

- URL place — идентификатор города

**ошибки** нет

**Это экспериментальный метод, при появлении новой версии не гарантируется работоспособность предыдущей!**

Формат данных в ответе.

```
{
  "id": <целое число>,        // идентификатор города
  "name": "строка",           // название города
  "frontier": true|false,     // является ли город фронтиром
  "new_for": <timestamp>,     // время, до которого город считается новым
  "updated_at": <timestamp>,  // время последнего обновления информации
  "description": "строка",    // описание города

  "position": {"x": <целое число>, "y": <целое число>},   // координаты города

  "politic_power": <politic_power>,     // политическое влияние города
  "persons": <persons_info>,            // Мастера
  "attributes": <attributes_info>,      // все параметры города
  "demographics": <demographics_info>,  // расовый состав
  "bills": <bills_info>,                // действующие записи в книге судеб
  "habits": <habits_info>,              // черты города
  "chronicle": <chronicle_info>,        // последние записи в летописи
  "accounts": <accounts_info>,          // краткая дополнительная информация об игроках, связанных␣
↪с городом
  "clans": <clans_info>                 // краткая дополнительная информация о кланах, связанных с␣
↪городом
}
```

(continues on next page)

(продолжение с предыдущей страницы)

```
<politic_power> = {
    "power": {
        "inner": { "value": <дробное число>,            // суммарное влияние ближнего круга
                   "fraction": <дробное число> },       // доля среди остальных городов
                                                        // (Фронтир и Ядро считаются отдельно)

        "outer": { "value": <дробное число>,            // суммарное влияние «толпы»
                   "fraction": <дробное число> },       // доля среди остальных городов
                                                        // (Фронтир и Ядро считаются отдельно)

        "fraction": <дробное число>                     // доля общего влияния среди остальных␣
↪городов
                                                        //(Фронтир и Ядро считаются отдельно)
    },
    "heroes": {                                         // влияющие на город герои (ближний круг и␣
↪кандидаты в него)
        "size": <целое число>,                          // количество героев сверху списка, на␣
↪которых может действовать проект
        "rating": [[<целое число>, <дробное число>],    // перечень героев ближнего круга с␣
↪влиянием <id героя, влияние>
                   //...
                  ]
    }
}

<persons_info> = [
   { "id": <целое число>,                       // идентификатор Мастера
     "name": "строка",                          // имя
     "gender": <целое число>,                   // пол
     "race": <целое число>,                     // раса
     "type": <целое число>,                     // профессия
     "next_move_available_in": <дробное число>, // количество секунда до момента, когда Мастер␣
↪может переехать в другой город
     "politic_power_fraction": <дробное число>, // доля влияния в городе
     "building": {                              // информация о здании, если оно есть
         "id": <целое число>,                   // идентификатор
         "position": {"x": <целое число>,       // позиция
                      "y": <целое число>},      //
         "type": <целое число>,                 // тип, значения перечислены на странице API
         "created_at_turn": <целое число>,      // номер хода на котором создано
     } | null,
     "personality": {                           // характер
         "cosmetic": <целое число>,             // идентификатор косметической особенности␣
↪характера
         "practical": <целое число>             // идентификатор практической особенности характера
     }
   },
   ...
]

<attributes_info> = {                               // информация о всех параметрах города
    "effects": [                                    // эффекты, действующие на город
        { "name": "<строка>",                       // название эффекта
          "attribute": <целое число>,               // на какой аттрибут влияет
          "value": <целое>|<дробное>|"<строка>"|null // значение, null, если значение комплексное␣
↪и пока не сериализуется в API
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        },
        ...
    ],
    "attributes": [                                 // итоговые значения аттрибутов
        { "id": <целое число>,                      // идентификатор аттрибута
          "value": <целое>|<дробное>|"<строка>"|null // значение, null, если значение комплексное␣
↪и пока не сериализуется в API
        }
    ]
}

<demographics_info> = [
  { "race": <целое число>,       // раса
    "percents": <дробное число>, // текущая доля (от 0 до 1)
    "delta": <дробное число>,    // изменение в день
    "persons": <дробное число>}, // влияние Мастеров (от 0 до 1)
  ...
]

<bills_info> = [
  { "id": <целое число>,           // идентификатор записи
    "caption": "строка",           // название записи
    "properties": ["строка", ...] }, // перечень описаний эффектов
  ...
]

<habits_info> = {  // информация о каждой черте города в формате "идентификатор черты":
↪{информация о черте}
  "<целое число>": {"interval": <целое число>,             // идентификатор текущего «уровня» черты
                    "value": <дробное число>,              // текущее абсолютное значенеи черты
                    "delta": <дробное число>,              // величина изменения черты за один час
                    "positive_points": <дробное число>,    // суммарное позитивное влияние героев␣
↪на черту города
                    "negative_points": <дробное число> },  // суммарное негативное влияние героев␣
↪на черту города
  ...
}

<chronicle_info> = [("строка", "строка", "строка"), ...] // последние записи из летописи о городе␣
↪в формате ("короткая дата", "длинная дата", "текст")

<accounts_info> = {
  "<целое число>": {            // идентификатор игрока
    "id": <целое число>,        // идентификатор игрока
    "name": "строка",           // ник
    "hero": {                   // краткая информация о герое
      "id": <целое число>,                    // идентификатор
      "name": "строка",                       // имя
      "race": <целое число>,                  // раса
      "gender": <целое число>,                // пол
      "power": [<целое число>, <целое число>], // физическая сила, магическая сила
      "level": <целое число> },               // уровень
    "clan": <целое число>|null  // идентификатор клана
  },
  ...
}
```

(continues on next page)

(продолжение с предыдущей страницы)

```
<clans_info> = {
 "<целое число>": <clan_info>  // <идентификатор клана, информация о клане>
 ...
}

<clan_info> = {
   "id": <целое число>, // идентификатор клана
   "abbr": "строка",    // аббревиатура
   "name": "строка"     // полное название
}
```

### 4.2.19 Мастера: подробная информация о Мастере

Подробная информация о конкретном Мастере

**адрес** /game/persons/<person>/api/show

**http-метод** GET

**версии** 1.2

**параметры**

- URL person — идентификатор Мастера

**ошибки** нет

**Это экспериментальный метод, при появлении новой версии не гарантируется работоспособность предыдущей!**

Формат данных в ответе.

```
{
  "id": <целое число>,         // идентификатор Мастера
  "name": "строка",            // имя
  "updated_at": <timestamp>,   // время последнего обновления информации

  "profession": <целое число>,  // профессия
  "race": <целое число>,        // раса
  "gender": <целое число>,      // пол

  "place": {                            // краткая информация о городе
      "id": <целое число>,              // идентификатор
      "name": "<строка>",               // название
      "size": <целое число>,            // размер
      "specialization": <целое число>, // специализация
      "position": {                     // координаты
          "x": <целое число>,
          "y": <целое число> }
  },

  // формат следующих параметров такой же, как в методе получения информации о городе

  "building": <словарь>|null,          // информация о здании, если оно есть
  "politic_power": <politic_power>,    // политическое влияние
  "attributes": <attributes_info>,     // все параметры Мастера
  "chronicle": <chronicle_info>,       // последние записи в летописи
```

(continues on next page)

(продолжение с предыдущей страницы)

```
  "job": <job_info>,                    // информация о проекте
  "accounts": <accounts_info>,          // краткая дополнительная информация об игроках, связанных␣
↪с Мастером
  "clans": <clans_info>                 // краткая дополнительная информация о кланах, связанных с␣
↪Мастером
}

<job_info> = {
  "name": "строка",                  // название проекта
  "effect": <целое число>,           // тип эффекта по завершению проекта
  "positive_power": <целое число>,   // накопленное положительное влияние
  "negative_power": <целое число>,   // накопленное отрицательное влияние
  "power_required": <целое число>    // требуемое влияние
}
```

### 4.2.20 Карта: получить карту

Карта мира на указанный (или последний) ход. Версия карты есть не для каждого хода. Получить список ходов, для которых есть сохранённая карта, можно отдельным запросом.

**адрес** /game/map/api/region

**http-метод** GET

**версии** 0.1

**параметры**

- GET: turn - целое, номер хода, на который необходимо получить карту

**ошибки**

- no_region_found - для заданного хода нет сохранённых данных

Формат данных в ответе.

```
{
    "turn": <целое>,                // номер хода, на котором была создана эта версия карты
    "region": {                     // описание карты
        "format_version": "строка", // версия формата
        "map_version": "строка",    // уникальный идентификатор этой версии карты
        "width": <целое>,           // размер карты по ширине
        "height": <целое>,          // размер карты по высоте

        "draw_info":            // инструкция по отрисовке
        [                       // строки карты
            [                   // столбцы карты
                [               // список спрайтов конкретной клетки
                    [<целое>,   // идентикаитор спрайта
                     <целое>], // идентификатор поворота:
                               // 0 - 0 градусов
                               // 1 - 90 градусов
                               // 2 - 180 градусов
                               // 3 - 270 градусов
                    ...
                ]
            ]
        ],
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        "places": {                     // перечень городов
            "<целое>": {                // идентификатор города
                "name": "строка",       // название
                "race": <целое>,        // раса
                "pos": {"y": <целое>,   // координаты клетки на карте
                        "x": <целое>},
                "id": <целое>,          // идентификатор города
                "size": <целое>,        // размер
                "clan_protector": null|<clan_info> // информация о клане-протекторе города, если
↪он есть
            }
        },
        "roads": {                      // перечень дорог
            "<целое>": {
                "point_1_id": <целое>, // из какого объекта (города) идёт дорога
                "point_2_id": <целое>, // в какой объект (город) идёт дорога
                "id": <целое>,          // идентификатор дороги
                "exists": true|false,  // видим ли дорога на карет
                "length": <дробное>,   // длинна дороги
                "path": "строка"        // путь из точки 1 в точку 2 по клеткам, последовательность
↪символов:
                                        // l - left - влево
                                        // r - right - вправо
                                        // u - up - вверх
                                        // d - down - вниз
            }
        }
    }
}
```

Актуальный список спрайтов можено найти в константах браузерного клиента.

Более удобное описание спрайтов можно найти в python коде

### 4.2.21 Карта: получить список версий карт

Список ходов, для которых есть соответствующая версия карты.

**адрес** /game/map/api/region-versions

**http-метод** GET

**версии** 0.1

**параметры** нет

**ошибки** нет

Формат данных в ответе.

```
{
    "turns": [<целое>, ...] // номер хода, для которого есть карта
}
```

### 4.2.22 PvP: Отправить героя на арену

Отправить заявку на бой на арене.

**адрес** /game/pvp/api/call-to-arena

**http-метод** POST

**версии** 0.1

**параметры** нет

**ошибки** нет

Формат данных в ответе.

```
{
    "info": <pvp_info> // актуальная информация о pvp боях и вызовах
}

<pvp_info> = {"accounts": <accounts_info>,          // краткая дополнительная информация об игроках,
↪ относящихся к pvp
              "clans": <clans_info>                 // краткая дополнительная информация о кланах,␣
↪относящихся к pvp
              "active_arena_battles": <целое>,      // количество активных боёв между игроками
              "active_bot_battles": <целое>,        // количество активных боёв с ботами
              "arena_battle_requests": [<arena_battle_request>, ...]} // список активных вызовов␣
↪на арену

<arena_battle_request> = {"id": <целое>,              // идентификатор вызова
                          "initiator_id": <целое>,    // идентификатор игрока, отправившего вызов
                          "matchmaker_type": <целое>, // тип боя, см. в списке типов
                          "created_at": <timestamp>,  // дата создания вызова
                          "updated_at": <timestamp>}  // дата обновления вызова
```

### 4.2.23 PvP: Отозвать героя с арены

Отозвать заявку на бой на арене.

**адрес** /game/pvp/api/leave-arena

**http-метод** POST

**версии** 0.1

**параметры** нет

**ошибки** нет

Формат данных в ответе.

```
{
    "info": <pvp_info> // актуальная информация о pvp боях и вызовах
}
```

### 4.2.24 PvP: Принять вызов другого игрока

Принять вызов другого игрока.

**адрес** /game/pvp/api/accept-arena-battle

**http-метод** POST

**версии** 0.1

**параметры**

- GET: battle_request_id — идентификатор вызова

**ошибки**

- pvp.accept_arena_battle.no_battle_request_found — вызова не найден
- pvp.accept_arena_battle.already_in_battle — вы или противник уже сражаетесь на арене

Метод является «неблокирующей операцией» (см. документацию), формат ответа соответствует ответу для всех «неблокирующих операций».

### 4.2.25 PvP: Начать бой с ботом

Начинает бой с ботом, если есть свбодные боты.

**адрес** /game/pvp/api/create-arena-bot-battle

**http-метод** POST

**версии** 0.1

**параметры** нет

**ошибки**

- pvp.create_arena_bot_battle.no_free_bots — нет свободных ботов, надо повторить попытку позже

Метод является «неблокирующей операцией» (см. документацию), формат ответа соответствует ответу для всех «неблокирующих операций».

### 4.2.26 PvP: получить информацию о pvp боях и вызовах

Возвращает информацию о текущих вызовах и боях.

**адрес** /game/pvp/api/info

**http-метод** GET

**версии** 0.1

**параметры** нет

**ошибки** нет

Формат данных в ответе.

```
{
    "info": <pvp_info> // актуальная информация о pvp боях и вызовах
}
```

## 4.3 Перечисления

Все актуальные значения перечислений можно найти на специальной странице сайта игры.

Это позволяет избежать постоянной синхронизации их значений между игрой и документацией.

## 4.4 Общие ошибки

В этом разделе перечислены ошибки, которые могут возникнуть при вызове любого (или почти любого) метода.

- `api.no_method_version` — не указана версия метода;
- `api.no_client_indentificator` — не указана версия клиента;
- `api.wrong_client_identificator_format` — неверный формат идентификатора клиента;
- `api.wrong_method_version` — неверная версия метода;
- `common.login_required` — необходимо войти в игру;
- `common.staff_required` — необходимы права администратора;
- `common.superuser_required` — необходимы права суперпользователя;
- `third_party.access_restricted` — доступ к функциональности запрещён сторонним приложениям. Необходимо авторизоваться с помощью логина/пароля.

## 4.5 Ресурсы игры

Все актуальные ресурсы игры можно найти на специальной странице сайта игры.

На этой странице мы собрали все графические ресурсы, которые могут пригодиться в разработке ваших приложений для игры.

Это позволяет избежать постоянной синхронизации их значений между игрой и документацией.

Глава 5

# Геймдизайн

В этом разделе собраны устоявшиеся общие правила и соглашения по разработке лора, механик игры и их оформлению.

Много подробной информации о мире игры можно найти в разделе каноничного фольклора.

## 5.1 Расы

Каждая раса в обязательном порядке описывается следующими параметрами:

- название;
- список основополагающих характеристик;
- предпочитаемый ландшафт;
- особенности генерации имён;
- принципы задания мобов;
- принципы задания артефактов;
- принципы задания лута.

### 5.1.1 Люди

**характеристики** не имеют расовых способностей, пристрастий к профессиям; легко подвержены страстям и порокам; конфликтны; склонны вступать в обширные организации объединенные одной идеей, целью, верой; имеют огромное количество разнообразных религиозных культов и конфессий.

**ландшафт** поля

**имена** славянские языческие

**мобы** бандиты, сектанты, революционеры, робин-гуды, свихнувшиеся волшебники, маньяки, шарлатаны, наёмные убийцы.

**артефакты** мастерски изготовленные вещи в основном не волшебные, ювелирные украшения.

**лут** самодельные амулеты, талисманы, сломанные/старые вещи, письма/расписки, наследственные ценности (сервизы, ковры).

### 5.1.2 Гоблины

**характеристики** приверженцы научно технологического прогресса; рационалисты; дельцы и банкиры; учёные-(ал)химики-биологи; любят сырость (болотные жители, потому, что климат хороший и удобно отходы опытов уничтожать); любят всё классифицировать; неторопливы; любят сложный тонкий юмор; часто заводят питомцев (от улитки или апельсинового дерева, до дракона); обладают феноменальной памятью, но при этом имеют огромные библиотеки, призванные просвящать молодое поколение; щуплые; малы ростом; зачастую физически слабые; имеют высокий иммунитет к ядам и болезням; практически все банки Пандоры принадлежат им; поклоняются четырем великим силам-Богам: Гзанзар — бог-хранитель единого знания, Дабнглан — бог невидимых сил, Крангимз — богиня путей исследования и Чарнгранг — бог постоянного движения.

**ландшафт** болотистые местности

**имена** корейские

**мобы** сбежавшие питомцы и результаты экспериментов, свихнувшиеся ученые и исследователи, охотники за редкими реагентами

**артефакты** сложные механизмы магически заряженные, обработанные химией «обычные вещи», «живые» предметы (вроде браслета-ящерки)

**лут** дохлые «живые» предметы, алхимические-биологические реагенты, отходы экспериментов, приспособления для опытов (посуда и прочее), свитки и книги, долговые расписки.

### 5.1.3 Дварфы

**характеристики** кузнецы и шахтёры; не чураются грязной работы; горячий нрав; низкорослые; широкие в плечах; кряжистые; не болеют и легко переносят яды; долго помнят обиды; любят торговаться; высоко ценят дружбу, но нелегко на нее соглашаются; имеют огромное количество разнообразных производственных и промышленных гильдий; поклоняются Великому Творцу.

**ландшафт** горы

**имена** скандинавские

**мобы** подземная живность, обиженные/оскорблённые

**артефакты** высококлассные «обычные вещи», механическое оружие и походное снаряжение, «обычные вещи» с художественной выделкой

**лут** рабочие инструменты, материалы, незаконченные/разобранные предметы

### 5.1.4 Эльфы

**характеристики** в первую очередь маги; аристократы; культурные деятели(поэты, певцы, музыканты); садоводы; ценители всего возвышенного; любят красивую одежду; считают все остальные расы низшими созданиями; имеют неприязнь к животным; поклоняются магии, самим себе и энергии безграничного пространства.

**ландшафт** леса

**имена** эльфийские (сейчас используются DnD)

**мобы** свихнувшиеся маги и разнообразные повелители стихий, обиженные аристократы, лесные жители, магические создания

**артефакты** зачарованное оружие и доспехи, магические амулеты и кольца, скованные стихии

**лут** рукописи, проклятые предметы экипировки, затейливые безделушки, драгоценности, аристократическая одежда

### 5.1.5 Орки

**характеристики** гордые; сильные; высокие; крайне выносливые; помешаны на чести и славе; превосходные наездники и дрессировщики животных; не подвержены болезням; прирождённые воины; коллективисты; благо общества ставят превыше своего; хранители традиций предков; без лишних слов изгоняют из племени провинившихся соплеменников, но при этом готовы принять действительно достойного представителя другой расы в свое племя; крепко держат слово; поклоняются природе и жизни.

**ландшафт** пустыни

**имена** монгольские

**мобы** дикие и прирученные животные, разного рода изгнанники, партизаны, охотники за головами, варвары и амазонки

**артефакты** не магическое мастерски изготовленное оружие, амулеты и талисманы

**лут** трофеи, сломанные/бесполезные вещи, грубые статуэтки животных

## 5.2 Персонажи

Технические описания заметных персонажей игровой вселенной. Используются для правильной подачи текста от лица персонажа (например, при создании описания монстров и артефактов).

### 5.2.1 Эрмид Тёмный

Мужчина эльф.

Живший во вторую эпоху собиратель сказок и преданий эльфов. Издал большой сборник «Сказания леса».

### 5.2.2 Хан и Туан

Мужчины гоблины.

Самые известные учёные второй эпохи.

Прославились тем, что посветили свои жизни составлению большой энциклопедии, в которую постарались включить все объединенные знания о мире. На данный момент это самая полная из существующих энциклопедий. Она насчитывает более двадцати томов.

Хан и Туан ведут замкнутый, можно сказать отшельнический образ жизни. Лишь изредка они появляются в городах для посещения библиотек в богатых домах, там же они оставляют для переписи новые тома своей энциклопедии.

Стилистика описания мобов сухая. Не редко используются такие выражения как: основа рациона, среда обитания, окрас:

- Бородун или ползун обыкновенный. Обитает в заболоченной местности иногда на заросших камышом и кустарником берегах рек. Основными внешними видовыми особенностями являются: круглое тело с тонкой прозрачной кожей, шесть глаз расположенных по дуге и мелкие слабо развитые щупальца...

Похоже на смесь развернутого описания животных в вики и «Энциклопедию Максима Мунди» в саге о Ведьмаке и Цири Анджея Сапковского.

Хан и Туан периодически включают в энциклопедию куски из других книг например: «Упорядоченного видового сборника» Куанга и «Сказаний леса» Эрмида Тёмного.

Стиль в энциклопедии един, за исключением тех описаний, где упоминаются достижения гоблинов, их самые распространенные профессии, личности помогающие им и так далее. Там идут просто оды, расхваливание или заведомое умалчивание, переиначивание фактов. Например:

- «...Охотник за реагентами это одна из удивительнейших профессий жителей нашего мира. Она является фундаментом научных открытий и достижений. Не редко сопровождаемая большим, риском и опасностью для жизни она дает возможность почти любому приблизиться к свету учения и познания...»
- «...По своей природе благородные господа, принадлежащие к этой профессии, с большой ответственностью относятся к своей работе и порой чересчур резко реагируют, если их от неё отвлекают. Тем более что жертвами являются, как правило, глупцы мешающие науке...»

Приписка в конце описания:

> *«Большая энциклопедия Хана и Туана» том седьмой* (номер тома меняется)

### 5.2.3 Каниллин

Мужчина эльф.

Купец, живущий в одном из городов. Имеет лавку. Скупает и продаёт любые трофеи, добытые героями на трактах. Покупает, в основном, у героев, продаёт: если полезное снаряжение, то героям же, если «хлам», то коллекционерам.

За прилавком стоит сам. Говорит он один, но подразумевается, что «за кадром» есть собеседник (обычно это покупатель или продающий товар герой). Каниллин надменный, не дружелюбный (но мастер своего дела, и потому клиентов у него всегда много). Не торгуется, но всегда стремится назвать цену выгодную для себя, утверждая, что это «честная цена»... Так ли это, точно не известно, но у читателя должно быть иногда подозрение, что нет. Говорит порой витиевато, часто делает оскорбительные намёки (очень редко прямым текстом). Терпим к эльфам, представителей других рас считает отсталыми, грязными, не понимающими ценности конкретного товара. При этом, купец часто готов

блестнуть эрудицией и объяснить глупому клиенту, что за редкая вещь попала к нему в руки, или как она изготавливается.

Приписки в конце описания:

*Купец Каниллин «Торг с путешественником»*

*Купец Каниллин «Предложение задания путешественнику»*

*Очирбат и Йорл «Утро в городе»*

### 5.2.4 Куанг

Мужчина гоблин.

Умер во вторую эпоху.

В начале своей карьеры занимался изучением флоры и фауны окружающего мира. В дальнейшем уделил особое внимание проверке и систематизации информации изложенной в древних манускриптах и трактатах.

Результатом ученой деятельности Куанга стал «Упорядоченный видовой сборник». Эта книга является энциклопедическим словарем, по мнению многих ученых, лучшим из ныне существующих.

Стилистика описания мобов характерная для всех зоологических или ботанических энциклопедий и словарей. Пример:

```
Класс - лошадиные
Вид - единороги
Строение - крепкое
Особенности - имеют один исходящий из основания лба рог
Среда обитания - лес
Скорость передвижения - быстрая
Окрас -  варьируется от снежно белого до сизо-голубого
Кожа - покрыта мягкой чуть вьющейся шерстью, удлиняющейся на холке и образующей гриву
```

Вид классификации можно заменять, то есть:

- не «Класс», а «Род»;
- не «Вид», а «Подвид»;
- и так далее.

Что-то можно вообще убрать, например, такой пункт как «Кожа» или добавить, например, пункт «Хвост» (Хвост – имеет метельчатое строение с толстым длинным волосом).

В общем, должно напоминать краткую классификацию в вики, когда открываешь, допустим, животное какое, или насекомое.

В описаниях по отдельности не бывает, то есть «Упорядоченный видовой сборник» Куанга упоминается, допустим, в «Большой энциклопедии Хана и Туана» (в виде сносок или вводного текста).

Пример: Более-менее точное описание единорога приводится в «Упорядоченном видовом сборнике» Куанга: «Класс - лошадиные Вид -. . . »

### 5.2.5 Лялислав Бездомный

Мужчина человек.

Умер в конце второй эпохи.

Культовая личность. Является кумиром и примером большинства путешественников. Стал известен еще при жизни. Благодаря тому, что приколачивал в придорожных кабаках и постоялых домах краткие очерки, о своих странствиях подписываясь одним именем — Лялислав.

Скальды и барды подхватывали темы из этих записок и слагали баллады о неком Бездомном Лялиславе, неугомонном путнике. В результате это имя узнал весь мир. Записки в дальнейшем попали в коллекции библиотек и получили названия «Записки путешественника» как их автор Лялислав был уже указан со своим прозвищем Бездомный.

Лялислав Бездомный ассоциируется с веселым, смелым в меру романтичным странником, своим парнем в любой компании. Его образ полюбился не только людям, но и эльфам за умение ценить красоту и гармонию мироздания, дварфам за легкость и покладистость характера, гоблинам за ученость и стремление к знаниям, оркам за смелость. Есть, конечно, и те, кому этот путешественник не внушает уважения, но их не так уж и много, все-таки его заслуги в географических открытиях не признать нельзя.

Лялислав довольно критично порой высказывается о многих вещах. Например о безответственности в магии или алхимии. Ему не чужда ирония, он смело высмеивает предрассудки и глупость.

Многие в равной степени полюбили его и возненавидели за здоровый, трезвый, не прикрытый красивыми словами взгляд на мир.

В конце своей жизни Лялислав написал довольно большое произведение, своего рода энциклопедию странствий «Брильянтовый тракт», в которую вошли карты описания флоры и фауны, культуры и обычаи мира.

#### «Записки путешественника»

Лялислав пишет в легком почти разговорном стиле. Периодически проскакивают междометия: ну... э...э...э... или вводные слова: в общем, короче говоря. Он просто рассказывает, иногда задумываясь. В тексте это выглядит, например, так: «так... ну начнём...».

Также Лялистав довольно часто использует пояснения и уточнения на пример: «дело в том, что»; «однако»; «как оказалось». «Записки путешественника» написаны так, чтобы читатель слегка улыбался, и ему было все максимально понятно.

Иногда автор вставляет личное отношение к тому или иному факту. Пример:

- «...Тогда конечно меня, мягко говоря, обескуражило его наличие в топях, но позднее всё стало на свои места. Дело в том, что некоторые дварфы филонят, избавляясь от своих големов...».
- «Неповоротливое существо оказалось агрессивным, крепким здоровьем и весьма неплохо справляющимся с боем».
- «Казалось бы, милые ни чем, ни кому не мешающие создания, крылышки, забавное пузице ещё и светится, но встретившись с одичавшим светлячком мне, в общем, эти насекомые перестали нравиться».
- «...Кому-то когда-то давно пришла в голову «гениальная» мысль – приручить светлячка...».

Как правило Лялислав делает предисловия к описаниям. Как он встретил монстра, где это было, то есть предшествующие события. Например:

- «Желая увидеть неведомые края и рассказать о ещё ни кому не известных местах нашего мира, занесло меня как-то на болота. Не сказать что очень далёкие и глухие, но за клюквой и лечебными пиявками туда ни кто не ходил».

Также для Лялислава характерна манера задавать вопросы и самому на ник отвечать. Часто слегка шутливо.

- «И так, что же это такое. Это чудное твореньеце природы, имеющее в теле полтора локтя длины».

Ему свойственны такие обращения как: зверушка, милый зверёк, твореньеце и т.д. он часто использует их по отношению к опасным монстрам.

Но если речь его идет о действительно смертоносном враге он всегда дает это понять и предостерегает, без шуток:

- «Гидра это очень-очень опасная тварь, если нет крайней необходимости, не дразните судьбу и, не вступая в бой быстро-быстро ретируйтесь».

Описание монстра от Лялислава чем-то может напоминать описания упырицы в книге «Последнее желание» Анджея Сапковского (там где описывается принцесса, Фольтестова дочечка).

Приписка в конце описания:

> *Лялислав Бездомный «Записки путешественника»*

#### «Брильянтовый тракт»

В брильянтовом тракте стилистика очень похожа на «записки путешественника», только более строгая. Исключены междометия и личное отношение к вещам, не относящимся на прямую к теме.

Чаще встречается вводная информация. Например:

- «Если внимательно просмотреть летописи, то там можно встретить упоминания о нашествиях этих насекомых, но если лень возится с пыльными свитками, то те же знания можно подчерпнуть из уст простого деревенского народа».

Брильянтовый тракт — это те же записки путешественника только менее шутливые и изложенные более художественно, но в присущем Лялиславу стиле.

Приписка в конце описания:

> *Лялислав Бездомный «Брильянтовый тракт»*

### 5.2.6 Очирбат и Йорл

Очирбат - мужчина орк. Йорл - мужчина дварф.

Друзья-напраники занимающиеся патрулированием трактов между городами (эдакое отражение параметра «безопасность» города). Оба бывшие бессмертные герои, но сейчас «отпущенные» Хранителями на относительный отдых... Впрочем, этой информации у игроков нет. На самом деле, они по-прежнему бессмертны.

Повествование от них выглядит, как диалог. Йорл часто зовёт орка не полным именем, а Чирба или «зелень». Очирбат иногда зовёт дварфа «борода». В дороге они рассказывают друг другу истории из своего «геройского» прошлого», делятся опытом и мнением обо всём подряд. манера речи у обоих простая, грубоватая. Они понимают друг друга с полуслова, но так, что бы и читатель мог понять, что имел в виду говоривший. Часто подначивают друг друга, но беззлоюно, по-дружески. Шутят при каждой возможности.

Обычно орк и дварф ведут диалог на тракте, во время патрулирования. Но иногда разговор может состояться в городе при различных обстоятельствах.

Приписки в конце описания:

*Очирбат и Йорл «Патрулирование трактов»*

*Очирбат и Йорл «Снова за работу»*

*Очирбат и Йорл «Утро в городе»*

### 5.2.7 Переяр

Мужчина человек. Герой на пенсии.

Старенький, но ещё крепенький. Живёт в городе.

С юности отправился путешествовать и искать приключения. Странствовал очень долго, по всей видимости Хранитель «отпустил» его и дал состариться.

Сам называет себя охотником, хотя по сути типичный авантюрист, не брезговавший «охотой» не только на зверей и монстров, но и вполне разумных жителей мира. Коллекционировал трофеи (лут и артефакты). Часть коллекции находится в той комнате, где гость беседует с Переяром.

Описания мобов от его лица происходит следующим образом:

- Переяр сидит дома у камина и общается с молодым гостем. Гость соседский мальчишка (без отца, есть мать, Переяра любит и уважает, как дадю).
- Слова гостя не написаны, но иногда его реплики или вопросы как бы «есть за кадром». Пример: «. . . я раздумывать не стал, давай молотить его клинком. Вон тем самым, над камином висит, видишь? Во-о!.. На чём я? А, так вот. . . » На вопрос «видишь?» гость как бы ответил «да».
- Порой гость (опять же «за кадром») выражает сомнения в правдивости Переяра. Пример: речь идёт о пустыне — «. . . Песок вокруг, ночь, прохладно. . . Да, в пустынях, что б ты знал, ночами холодно!..» Гость удивляется, как это в пустыне холодно?
- Гость — «мальчик домашний» и потому впечатлительный. Когда Переяр в рассказе упоминает нечто нелицеприятное (кишки, мозги, слизь и т.п.), а делает он (Переяр) это часто, гостя мутит, он (гость) бледнеет и т.д. Пример: «. . . Взял его за шкирку и головой о камень придорожный. Как арбуз!.. Что ты морщишься? Побледнел чего-то. . . Нет? Нормально? Ну хорошо, слушай. . . »
- Рассказ Переяра в «разговорном стиле» (даже «сверх разговорном»). В его речи порой может проскакивать: «э-э», «на чём это я?.. А! Так вот. . . », «Нет, мол, говорю». Переяр часто (почти в каждом рассказе о своих «подвигах») упоминает «кровавые» подробности (на дурноту гостя обращает внимание иногда). Пример его манеры речи привести сложно (разве что люди многие так говорили в жизни, «до революции», а некоторые до сих пор), но прочитав два-три описания мобов от его лица, вам станет всё до конца понятно.
- Переяр, как правило, называет свой меч «сталью». Пример: «. . . сталь свою добрую об эту сволочь пачкать не хотелось. . . »
- Опять же, Переяр уже «в возрасте» и порой ворчит в адрес молодёжи, и что де времена не те нынче, раньше и мобы были опаснее и герои благороднее.
- Помимо слов «простых», «мужицко-крестьянских», Переяр может употребить «заумное» слово, услышанное в странствиях.

Приписка в конце описания:

*Из бесед у камина со старым Переяром*

### 5.2.8 Перкунос Седой

Мужчина человек.

Один из первых ученых попытавшихся описать мир и его обитателей в огромном многотомном труде называющемся «О мире и жителях егонных». Давным-давно умер.

Имеет прозвище, приросшее к имени – Седой. Оно отражает даже не седины, а состояние души автора. Старый раздираемый фобиями вредный дед. Очень много знает, но мыслит догматично. Такие личности как он жестоки и консервативны.

Его фундаментальный труд стал основой для многих современных энциклопедий, хоть и содержит кучу спорных моментов.

Описание мобов это выдержки из выше упомянутого труда. Язык архаичный, периодически используются такие слова как: оные, энти, акоже, смердящие, поганцы, супротив, особливо.

Похожая стилистика присутствует в произведении Анджея Сапковсково «Ведьмак» (в главе «Край света» упоминается книга, которой руководствуются кметы при ловле сильвана, вот стиль её изложения, почти идентичен стилю Перкуноса Седого за одним исключением — он не использует слово ибо).

Также Перкунос Седой часто даёт советы и личную оценку, выводя ее как догму, то есть, не используя слово я:

- ... Дабы избавиться от боли страшной осиного укуса возьми лист лопуха обыкновенного подорожника лист другой... (пример совета)
- ... Аристократ энто один из лучших обитателей нашего мира. Онный являет собой развитость умственную, доблести и чести образчик... (пример положительной оценки)
- ... Скверна эта появляется токма тогда, когда дурни всякого рода мёртвых не погребают, а без должного почтения к онным в болото гнить бросают... (пример отрицательной оценки)

Приписка в конце описания:

> *Перкунос Седой «О мире и жителях егонных» том шестой* (номер тома меняется).

### 5.2.9 Виен-Нгуен и Эллунир

Женщина гоблин и мужчина эльф.

Виен - ученица некроманта Эллунира. Описания «от них», обычно, выглядят либо как диалог, либо как выписка из лабораторного журнала. В перовм случае Эллунир зовёт ученицу полным именем или просто Виен. Она его «учитель».

Приписки в конце описания:

> *Виен-Нгуен «Наставления учителя Эллунира»*
>
> *Из лабораторного журнала Виен-Нгуен, ученицы Эллунира*
>
> *Очирбат и Йорл «Утро в городе»*

## 5.3 Требования к изображениям карты

Карта состоит из квадратных клеток — тайлов.

Текущие текстуры для карты могут быть найден в репозитории игры.

### 5.3.1 Требования к тайлам

- Размер 32x32 px.
- Явно выделенные границы (как на шахматной доске), кроме тайлов воды.
- Один доминирующий фоновый цвет, который отвечает основному биому тайла (например, трава — зелёный, горы — серый, вода — синий).
- Большинство тайлов должно иметь уникальные цвета.
- Если присутствует мелкая растительность (или другие мелкие общераспространённые элементы: трава, грязь, трещины, камни), они отоброжается условными знаками по всему тайлу.
- Пповерх рисуется изображение, характеризующее основной признак биома (гора, дерево, холм), если он есть. Признак изображается в единственном экземпляре (лес=дерево, горы=одна гора)
- Рукотворные постройки занимают большую часть клетки и изображаются вместо основного признака (если такой имеется), но с учётом правильного цвета тайла.
- Условные знаки и признаки изображаются в виде сбоку.
- Рукотворные постройки изображаются в виде сверху под углом (видно часть фронтальных стен).
- Изображения персонажей изображаются в плоский профиль (без полуоборотов).

### 5.3.2 Пример текстуры

### 5.3.3 Очерёдность тайлов в текстуре

#### Первый ряд (биомы)

- луга
- пустыня
- болото
- растрескавшаяся земля
- грязь
- высохрашая трава
- луга на холмах
- пустыня на холмах (дюны)
- болото на хомлах
- сухая трава на хомлах
- грязевые холмы
- высохшие растресковшиеся холмы

### Второй ряд (биомы)

- хвойный лес
- лиственный лес
- болотный лес
- джунгли
- джунгли (не используется)
- высохший лес в пустыне (не используется)
- хвойный лес на холмах
- лиственный лес на холмах
- болотный лес на хомлах
- джунгли на холмах
- высохший лес на дюнах (не используется)
- трава в джунглях

### Третий ряд (биомы)

- низкие горы
- высокие горы
- фон для гор
- мелкая вода
- глубокая вода
- высохший лес
- высохший лес на холмах

### Чертвёртый ряд (дороги)

Тайлы дороги.

---

**План:** описать структуру тайлов и правила их объединения.

---

### Пятый ряд (здания)

- лесопилка
- сторожевая башня
- башня мага
- плаха
- ранчо
- кузница

- домик охотника
- домик рыболова
- торговый пост
- трактир
- ферма
- шахта

#### Шестой ряд (здания)

- храм
- лаборатория
- больница
- цех
- логово вора
- ломбард
- конюшни
- сцена
- мастерская портного
- бюро/офис

#### Седьмой ряд (города)

По 4 тайла на расу (для городов разного размера).

- люди
- дварфы
- эльфы

#### Восьмой ряд (города)

По 4 тайла на расу (для городов разного размера).

- гоблины
- орки

### Девятый ряд (герои)

По 2 тайла на расу (мужчина и женщина).

- люди
- дварфы
- эльфы
- гоблины
- орки

### Десятый ряд (техническое)

Рамки выделения позиции курсора.

Глава 6

# Проекты игроков

За время существования игры она обросла большим количеством сторонних проектов, поддерживаемых игроками. На этой странице собраны наиболее интересные (и актуальные) из них.

Авторы этих проектов всегда будут рады вашей помощи.

Обсуждение всех проектов игроков можно найти на форуме игры.

## 6.1 Android клиент для Сказки

Игровой клиент для платформы Android.

**URL** Google Play

**Репозиторий** github

**Тема на форуме** https://the-tale.org/forum/threads/6029

**Автор** wrewolf

## 6.2 IOS клиент для Сказки

**URL** App Store

**Репозиторий** github

**Тема на форуме** https://the-tale.org/forum/threads/6233

**Автор** ntwf

## 6.3 Расширение для браузера «The Tale Extended»

Браузерное расширение, улучшающее интерфейс:

- выводит наглядную информацию о последних действиях;
- сохраняет и выводит статистику;
- отправляет уведомления;
- помогает в игре.

**URL** Chrome, Firefox

**Репозиторий** github

**Тема на форуме** https://the-tale.org/forum/threads/1687

**Автор** standy

## 6.4 Тёмная тема The Tale

**URL** —

**Репозиторий** —

**Тема на форуме** https://the-tale.org/forum/threads/1407

**Автор** Experienced

## 6.5 Телеграм канал: «Пандорийская Газета»

Последние внутриигровые новости: политические разборки, инициативы игроков, новости разработки игры.

**URL** https://t.me/talepp

**Репозиторий** —

**Тема на форуме** https://the-tale.org/forum/threads/6331

**Автор** Грустный Ворон

## 6.6 Информер

Картинка с информацией о герое игрока для размещения в подписях на форумах и в аналогичных местах.

**URL** —

**Репозиторий** —

**Тема на форуме** https://the-tale.org/forum/threads/515

**Автор** Yashko

## 6.7 Альтернативный информер

Картинка с информацией о герое игрока для размещения в подписях на форумах и в аналогичных местах.

**URL** —

**Репозиторий** —

**Тема на форуме** https://the-tale.org/forum/threads/4422

**Автор** Нико д`Лас

## 6.8 Статистика гильдий

**URL** http://ttgs.herokuapp.com/

**Репозиторий** —

**Тема на форуме** https://the-tale.org/forum/threads/2305

**Автор** Kiberbit

## 6.9 Книга-игра «Простор» (текстовый квест)

**URL** http://somegoodstory.ucoz.ru/prostor.html

**Репозиторий** —

**Тема на форуме** https://the-tale.org/forum/threads/879

**Автор** Oatmeal Pecheneg

Глава 7

# F.A.Q.

## 7.1 Почему всё на русском языке?

Мы придерживаемся мнения, что языком разработки должен быть язык общения команды. Поскольку мы все разговариваем на русском, а игра ориентирована только на русскоговорящих пользователей, то нет смысла играть в испорченный телефон.

## 7.2 Переведите «Сказку» на английский (китайский, клингонский)

Игра использует нетривиальный генератор текста. Отсутствие времени и уровень знания английского разработчиками не позволяет этого сделать.

Кроме того, необходимо будет переводить множество художественных текстов, что дорого и сложно. Сейчас мы не можем позволить такую роскошь.