Руководство по проекту PropMan

Дата последней редакции: 22-е сентября 2024 года
Ред. 1
Автор: Сергей Пономаренко

Содержание
1. Общая информация …….…… 1
2. Настройка ……..…….. 3
3. Сущности и схема базы данных ……..…... 7
4. Типы пользователей и работа системы ……24
5. Применённые технологии 27
6. Возможности дальнейшего улучшения 28
7. Опыт разработки …….. 28

1. Общая информация

Этот раздел предоставляет общее описание проекта, его сути, текущего состояния и области применения, а также содержит некоторые детали, касающиеся ограничений в рамках проекта и потенциальных направлений для улучшения.

Данное руководство предназначено для всех, кто будет проверять и/или тестировать проект PropMan. Исходный код доступен по следующему адресу: https://github.com/SergejsP84/PropManProject/tree/master.

Программный продукт PropMan был разработан автором в качестве выпускного проекта в рамках его учебы в Tel-Ran. Название “PropMan” является сокращением от “Property Management” (управление недвижимостью), а фактическое название, под которым продукт будет работать при развёртывании, может быть выбрано администратором во время настройки. Основной целью было разработать надежную серверную систему для специфической отрасли, а в качестве предмета проекта автором была избрана платформа для управления недвижимостью. Учитывая ограничения по времени и предполагаемые объемы работы, было решено ограничить предметную область арендой недвижимости. Поэтому конечная цель проекта была определена как разработка серверной части для платформы аренды недвижимости, аналогичной Airbnb.com или Booking.com.

Таким образом, функциональность продукта PropMan частично имитирует функциональность вышеупомянутых платформ. Основной бизнес-процесс ориентирован на предоставление потенциальным арендаторам возможности арендовать недвижимость на относительно короткие сроки у менеджеров, которым, в свою очередь, предоставляется функционал для размещения своей недвижимости на аренду и получения прибыли от таковой аренды. Описание сопутствующих процессов и более подробная информация о ключевых функциях доступны в разделе 4 «Типы пользователей и работа системы» — проект обеспечивает 99 различных пользовательских

функций для пользователей, без учета запланированных сервисных задач, инструментов генерации сущностей и тестовых средств. Для тестирования этих функций предоставлена заранее подготовленная коллекция запросов Postman, включённая в проект.

Поскольку такая функциональность предполагает, помимо прочего, обработку платежей, автор старался уделить повышенное внимание вопросам безопасности, например, хранению данных платежных карт пользователей. Однако следует заметить, что автор не намеревался разрабатывать полностью функциональный механизм платежей, поскольку это очевидно превышало его текущие возможности по времени и усилиям. С учетом этого, функции, связанные с платежами, были разработаны с использованием “заглушек”, возвращающих простое логическое значение при их вызове. Тем не менее, в нужные сервисы включены соответствующие методы для извлечения, расшифровки и предоставления требуемых данных для подключения системы к внешнему поставщику платёжных услуг.

Следует отметить, что на момент последней редакции данного руководства в проекте отсутствуют юнит-тесты; автор был вынужден удалить устаревшие тесты, так как структура проекта значительно эволюционировала. В условиях актуальных графиков и временных ограничений приоритетом остается оптимизация существующего функционала. Тем не менее, на данной стадии проект может тестироваться следующими способами:

- Запуск программы из среды разработки IntelliJIDEA. Это требует ряда подготовительных действий, описанных в разделе 2 «Настройка». Доступ к функциональным точкам (endpoint) можно получить с помощью:
    - заранее подготовленной коллекции запросов Postman, доступной для скачивания здесь, или
    - интерфейса Swagger, который можно открыть, запустив программу и введя/скопировав в адресную строку браузера «http://localhost:8080/swagger-ui/index.html».
- Использование контейнеризированной версии проекта, созданной с помощью Docker. Однако настоятельно рекомендуется сначала запустить программу через IDEA, чтобы она корректно сгенерировала обязательные значения, что, по состоянию на данный момент, требует ввода некоторых данных в консоль.

В заключение данного вводного раздела автор хотел бы отметить, что проект нуждается в доработке; например, код может быть дополнительно отрефакторен для повышения эффективности, особенно при работе с большими базами данных. Однако, учитывая временные ограничения, это остается задачей на этап уже после защиты проекта. Код по-прежнему содержит несколько неиспользуемых и/или удаленных полей сущностей, методов и точек доступа; это связано с тем, что проект несколько раз изменялся в ходе разработки. Эти элементы также будут очищены или переработаны в процессе дальнейшего рефакторинга.

Другие потенциальные направления для улучшения описаны в разделе 6 «Возможности дальнейшего развития». Все последующие изменения будут отражены в дальнейших редакциях данного Руководства.

2. Настройка

В этом разделе описаны те действия, которые надлежит предпринять для обеспечения запуска программы в целях её тестирования.

a) Настройка системных переменных

Прежде всего, некоторые настройки программы, необходимые для полноценного функционирования проекта, для своей работы требуют наличия некоторых системных переменных. Чтобы несколько облегчить настройку системы, автор подготовил файл *propman_env_setup.bat*, который поможет сразу настроить все эти переменные. Этот файл можно скачать здесь; при этом, Вам нужно будет вручную выставить значения переменных, и запускать файл только после того, как эти значения будут выставлены. Всего нужно выставить 10 системных переменных.

Содержимое файла таково:

```
@echo off
SETLOCAL

REM Set environment variables temporarily (for the current session)
set "AES_SECRET_KEY=your_aes_secret_key"
set "JWT_SECRET_KEY=your_jwt_secret_key"
set "PROPMAN_DB_PASSWORD=your_db_password"
set "PROPMAN_DB_URL=your_mysql_database_address"
set "PROPMAN_DB_USERNAME=your_db_username"
set "PROPMAN_MAIL_HOST=your_mail_server"
set "PROPMAN_MAIL_PORT=your_mail_port"
set "PROPMAN_MAIL_USERNAME=your_email_address"
set "SPRING_MAIL_PASSWORD=your_spring_mail_password"
set "PROPMAN_PLATFORM_NAME=YourPlatformName"

REM Set environment variables permanently
setx AES_SECRET_KEY "your_aes_secret_key"
setx JWT_SECRET_KEY "your_jwt_secret_key"
setx PROPMAN_DB_PASSWORD "your_db_password"
setx PROPMAN_DB_URL "your_mysql_database_address"
setx PROPMAN_DB_USERNAME "your_db_username"
setx PROPMAN_MAIL_HOST "your_mail_server"
setx PROPMAN_MAIL_PORT "your_mail_port"
setx PROPMAN_MAIL_USERNAME "your_email_address"
setx SPRING_MAIL_PASSWORD "your_spring_mail_password"
setx PROPMAN_PLATFORM_NAME "YourPlatformName"
```

```
REM Display the variables to confirm they are set
echo SPRING_MAIL_PASSWORD=%SPRING_MAIL_PASSWORD%
echo AES_SECRET_KEY=%AES_SECRET_KEY%
echo JWT_SECRET_KEY=%JWT_SECRET_KEY%
echo PROPMAN_DB_PASSWORD=%PROPMAN_DB_PASSWORD%
echo PROPMAN_DB_URL=%PROPMAN_DB_URL%
echo PROPMAN_DB_USERNAME=%PROPMAN_DB_USERNAME%
echo PROPMAN_MAIL_HOST=%PROPMAN_MAIL_HOST%
echo PROPMAN_MAIL_PORT=%PROPMAN_MAIL_PORT%
echo PROPMAN_MAIL_USERNAME=%PROPMAN_MAIL_USERNAME%
echo PROPMAN_PLATFORM_NAME=%PROPMAN_PLATFORM_NAME%

ENDLOCAL
pause
```

Таким образом, чтобы запустить программу, Вам понадобится выставить в bat-файле следующие значения (пожалуйста, имейте в виду, что это нужно будет сделать дважды - для блока текущей сессии и для блока постоянных настроек):

№	**Фрагмент, который нужно заменить значением**	**Описание**
1.	your_aes_secret_key	Секретный ключ AES. Он необходим, так как некоторые значения в базе данных зашифрованы с использованием алгоритма AES.
2.	your_jwt_secret_key	Секретный ключ JWT. JWT-токены используются в UserResponseDTO, поэтому этот ключ также необходим.
3.	your_db_password	Пароль для соответствующего пользователя Вашей базы данных SQL, который нужен для того, чтобы программа могла получить к ней доступ.
4.	your_mysql_database_address	Адрес базы данных - обычно что-то вроде jdbc:mysql://localhost:3306/destiny_database.
5.	your_db_username	Имя пользователя (логин) для соответствующего пользователя вашей SQL базы данных, который нужен для того, чтобы программа могла получить к ней доступ.
6.	your_mail_server	Сервер для адреса электронной почты, который вы собираетесь использовать в качестве системной почты.
7.	your_mail_port	Порт вашей почты (например, 587).
8.	your_email_address	Фактический адрес электронной почты, который вы собираетесь использовать в качестве системной почты.
9.	your_spring_mail_password	Пароль к вышеупомянутому почтовому ящику.
10.	YourPlatformName	Название платформы, под которым она

		будет "работать" (можете выбрать любое по Вашему усмотрению).

Когда Вы закончите заменять эти значения, сохраните и закройте bat-файл, а затем запустите его. При этом в консоли должен отобразиться список выставленных переменных.

b) Запуск программы из среды IntelliJ IDEA

Для целей тестирования наиболее удобным и гибким подходом будет запустить программу из среды разработки IntelliJ IDEA. Обратите внимание, что вам всё равно необходимо будет установить вышеупомянутые переменные окружения, чтобы программа могла корректно работать.

Просто скачайте программу с https://github.com/SergejsP84/PropManProject.git в свою систему и запустите её. При первом запуске консоль предложит вам изменить пароль для DefaultAdmin и ввести его адрес электронной почты. Пожалуйста, не забудьте пароль для пользователя DefaultAdmin, так как он будет сохраняться даже при повторной генерации базы данных. Затем консоль спросит, хотите ли вы сгенерировать тестовую базу данных. Если вы запускаете программу для тестирования её функционала, рекомендуется создать такую базу данных: тестировать гораздо удобнее в уже заполненной среде. Кроме того, коллекция запросов Postman, также предоставленная для целей тестирования (её описание приведено далее в этом разделе), разработана для работы с предварительно сгенерированной тестовой базой данных.

После этого система будет готова к работе. Последний рекомендованный шаг — загрузить вышеупомянутую коллекцию запросов Postman, доступную по этой ссылке. Импортировав её в Postman, вы сможете начать процесс ознакомления и тестирования.

c) Запуск контейнеризированной версии программы

Проект содержит необходимые средства для настройки контейнеризированной версии с использованием Docker. Обратите внимание, что настоятельно рекомендуется сначала запустить проект из среды IDEA хотя бы дважды (или один раз, если вы решите не генерировать тестовую базу данных при первом запуске, либо вручную установите значение для NumericalConfig "ProposeSampleDatabaseCreation" в базе данных, созданной после первого запуска). В противном случае консольный ввод, который программа запрашивает во время этих первых запусков, может помешать правильному созданию контейнера. Попытка контейнеризировать проект без этих начальных шагов также исключит возможность генерации тестовой базы данных.
После завершения начальной конфигурации через консоль, вы можете создать Docker-контейнер, используя команду “**docker compose up --build**” в GitBash терминале IDEA.

d) Другая информация

Также был разработан набор Swagger-аннотаций, предоставляющий дополнительную возможность для ознакомления с проектом. Чтобы получить доступ к интерфейсу Swagger, запустите программу и введите http://localhost:8080/swagger-ui/index.html в адресную строку браузера. Вам понадобятся учетные данные доступа для DefaultAdmin или другого пользователя с необходимыми правами.

Далее приведены логины и пароли для автоматически сгенерированных пользователей из тестовой базы данных:

ID пользователя	ИМЯ ПОЛЬЗОВАТЕЛЯ	ПАРОЛЬ
Админы		
1.	DefaultAdmin	<Пароль, который Вы выставите>
2.	AnotherAdmin	AdminPassword123
Менеджеры		
1.	SNettleson	Ductus
2.	SUllman	HoraceDerwent
3.	CPaul	HeavensDoor
4.	GGolden	Nosferatu
5.	LitaC	SanookMakMak
Арендаторы		
1.	KennyM	CheatingDeath
2.	StanM	PinewoodDerby
3.	KyleB	Jewpacabra
4.	EricC	RespectMyAuth
5.	WendyT	ScienceRules
6.	ButtersS	AwGeeWhiz
7.	TolkienB	TokenOfHope
8.	CraigT	TouchTheSky
9.	TweekT	StayAwake
10.	GregoryY	ClassyTouch
11.	BebeS	BestFriend
12.	RebeccaR	RedHair
13.	ClydeD	Mysterion
14.	AnnieK	Butterfly

Здесь стоит отметить, что сгенерированные "тестовые" сущности не проходят обычный процесс создания: в частности, нет обязательного требования для более сложных паролей (у некоторых "пользователей" они достаточно простые), и многие сущности пользователей имеют одинаковый адрес электронной почты. Электронная почта является обязательным полем для всех сущностей пользователей, и система обычно не позволяет регистрировать пользователей с одинаковыми адресами электронной почты в пределах одной и той же роли. Однако, для целей тестирования большинство созданных пользователей используют одинаковые почтовые адреса. Это сделано специально, чтобы упростить процесс тестирования. Для тестирования были также созданы два почтовых аккаунта, и Вы можете использовать их на

платформе www.inbox.lv.

Email 1: propman_testmail1@inbox.lv
Логин: propman_testmail1
Пароль: <запросите его у автора через Discord>
Email 2: propman_testmail2@inbox.lv
Логин: propman_testmail2
Пароль: <запросите его у автора через Discord>

3. Сущности и схема базы данных

В этом разделе описываются сущности проекта, а также содержится схема базы данных как иллюстрация взаимоотношений.

Всего в системе наличествуют 26 сущностей. «Живущие» в системе сущности, приведённые далее, могут быть сгруппированы в пять категорий: пользовательские сущности, сущности процесса аренды, финансовые сущности, коммуникационные сущности и вспомогательные сущности.

I. Пользовательские сущности:
1) Админ (Admin)
Админ отвечает за контроль работы платформы, и имеет доступ к спектру инструментов для изменения настроек системы; создания сущностей, которые обычно генерируются автоматически, например, Возвратов (Refunds) или Выплат (Payouts), а также для иного вида вмешательства в работу системы сообразно требованиям ситуации, вплоть до создания, блокирования и удаления Объектов, Арендаторов и Менеджеров. Также имеется статический пользователь по имени DefaultAdmin, который не может быть удалён обычными средствами, и имеет возможность создавать и удалять аккаунты других Админов. Более подробно функции Админа описаны в разделе 4 «Типы пользователей и работа системы».

Поля объекта типа Admin:

№	Имя поля	Тип поля	Комментарий
1.	id	Long	ID сущности
2.	name	String	Полное имя Админа
3.	login	String	НЕОБХОДИМО, должно быть уникальным среди всех пользователей системы
4.	password	String	НЕОБХОДИМО, шифруется после ввода, требования к паролям к Админам НЕ применяются
5.	authorities	Collection	Связано с другой таблицей, admin_authorities
6.	knownIps	List	IP-адреса, с которых пользователь ранее заходил
7.	email	String	Адрес электронной почты пользователя

2) Менеджер (Manager)
Менеджер — это пользователь, который использует платформу для того, чтобы предлагать и сдавать в аренду свою(и) собственность(и). В этом качестве ему/ей предоставлены полномочия управлять своими объектами недвижимости различными способами, подтверждать запросы на бронирование от арендаторов, управлять своим профилем, включая учетные данные и т.д. Менеджер может владеть несколькими объектами недвижимости и может также зарегистрироваться в качестве Арендатора с той же электронной почтой (но с другим логином). Как и в случае с предыдущей сущностью, функции Менеджера более подробно описаны в Разделе 4 «Типы пользователей и работа системы».

Поля объекта типа Manager:

№	Имя поля	Тип поля	Комментарий
1.	id	Long	ID сущности
2.	type	ManagerType	Перечисляемое поле: может быть ЧАСТНЫЙ или КОРПОРАТИВНЫЙ
3.	managerName	String	Имя / фамилия менеджера или название компании, ОБЯЗАТЕЛЬНОЕ
4.	description	String	Может быть использовано Менеджером для добавления своего описания
5.	isActive	Boolean	Менеджер должен быть активен, чтобы его объекты недвижимости были доступны для аренды
6.	joindate	Timestamp	Указывает время, когда данный Менеджер присоединился к платформе
7.	login	String	Логин: должен быть уникальным в системе, ОБЯЗАТЕЛЬНОЕ
8.	password	String	Пароль: ОБЯЗАТЕЛЬНОЕ, должен содержать минимум 8 символов, включать хотя бы одну заглавную букву / строчную букву / цифру, хранится в зашифрованном виде
9.	properties	Set	Хранит объекты недвижимости, принадлежащие этому Менеджеру
10.	phone	String	Номер телефона Менеджера
11.	email	String	Электронная почта пользователя
12.	iban	String	IBAN пользователя. Не используется в текущей конфигурации, но предназначен для будущего

			использования при расширении платёжного функционала
13.	paymentCardNo	String	Номер платежной карты пользователя, хранится в зашифрованном виде, ключи хранятся в отдельном файле
14.	cardValidityDate	YearMonth	Месяц окончания действия карты
15.	cvv	char array	CVV пользователя, хранится как массив символов и дополнительно зашифрован, как и номер карты
16.	confirmationToken	String	Токен, используемый для подтверждения адреса электронной почты пользователя во время регистрации
17.	expirationTime	LocalDateTime	Время истечения действия вышеупомянутого токена
18.	authorities	Collection	Связан с другой таблицей, manager_authorities
19.	knownIps	List	IP-адреса, с которых этот пользователь когда-либо заходил
20.	temporaryEmail	String	Используется, когда пользователь хочет изменить свою электронную почту
21.	temporaryPassword	String	Используется, когда пользователь хочет изменить свой пароль

3) Арендатор (Tenant)

Арендатор — это пользователь, который использует платформу для аренды недвижимости на определенный срок. Таким образом, этот пользователь инициирует бронирования и совершает платежи Арендатора. Арендатор также может управлять своим профилем, включая учетные данные и т.д. Другие функции Арендатора включают оставление отзывов, общение через сообщения, подачу жалоб и многое другое. Арендатор может одновременно быть и Менеджером с той же электронной почтой (но с другим логином). Как и в случае с предыдущей сущностью, функции Арендатора более подробно описаны в Разделе 4 «Типы пользователей и работа системы».

Поля объекта типа Tenant:

№	Имя поля	Тип поля	Комментарий
1.	id	Long	ID сущности
2.	firstName	String	Имя Арендатора
3.	lastName	String	Фамилия Арендатора
4.	currentProperty	Property	Собственность, в которой Арендатор проживает на данный момент
5.	isActive	Boolean	Арендатор должен быть

			активен, чтобы иметь возможность делать бронирования
6.	phone	String	Номер телефона Арендатора
7.	email	String	Электронная почта пользователя
8.	iban	String	IBAN пользователя. Не используется в текущей конфигурации, но предназначен для будущего использования при расширении платежной функциональности
9.	paymentCardNo	String	Номер платежной карты пользователя, хранится в зашифрованном виде, ключи хранятся в отдельном файле
10.	cardValidityDate	YearMonth	Месяц окончания действия карты
11.	cvv	char array	CVV пользователя, хранится как массив символов и дополнительно зашифрован, как и номер карты
12.	rating	float	Рейтинг Арендатора, который составляется из оценок, данных Менеджерами недвижимости, где этот Арендатор когда-либо проживал
13.	login	String	Логин: должен быть уникальным в системе, ОБЯЗАТЕЛЬНОЕ
14.	password	String	Пароль: ОБЯЗАТЕЛЬНОЕ, должен содержать минимум 8 символов, включать хотя бы одну заглавную букву / строчную букву / цифру, хранится в зашифрованном виде
9.	leasingHistories	List	Хранит историю прошлых бронирований Арендатора
10.	tenantPayments	Set	Платежи, произведенные Арендатором
11.	confirmationToken	String	Токен, используемый для подтверждения адреса электронной почты пользователя во время регистрации
12.	expirationTime	LocalDateTime	Время истечения действия вышеупомянутого токена
13.	preferredCurrency	Currency	Валюта, которую Арендатор предпочитает видеть в ценах и суммах платежей
14.	authorities	Collection	Связан с другой таблицей,

			tenant_authorities
15.	knownIps	List	IP-адреса, с которых этот пользователь когда-либо заходил
16.	temporaryEmail	String	Используется, когда пользователь хочет изменить свою электронную почту
17.	temporaryPassword	String	Используется, когда пользователь хочет изменить свой пароль

II. Сущности процесса аренды:

4) Недвижимость (Property)

Недвижимость — это объект, предлагаемый Менеджером через платформу и доступный Арендаторам для аренды. Недвижимость может быть различных типов, находиться в разных точках мира и иметь множество параметров. Это одна из ключевых сущностей в системе

Поля объекта типа Property:

№	Имя поля	Тип поля	Комментарий
1.	id	Long	ID сущности
2.	manager	Manager	Менеджер этой Недвижимости
3.	propertyStatus	PropertyStatus	Enum: может быть Available (доступна) или Blocked (заблокирована), так как статус Busy (занята) оказался не совсем подходящим для финальной конфигурации
4.	createdAt	Timestamp	Дата создания Недвижимости. Не используется в текущей версии, но может пригодиться в будущем, например, для промо-акций новых объектов
5.	type	PropertyType	Enum: Недвижимость может быть Room (комната), Hotel Room (номер в отеле), Apartment (квартира), House (дом), Commercial property (коммерческая недвижимость) или Other (другой тип недвижимости)
6.	address	String	Обычно включает название улицы и номер дома/квартиры
7.	country	String	Страна, в которой находится Недвижимость
8.	settlement	String	Населённый пункт — город, деревня и т.д.
9.	sizeM2	Float	Площадь Недвижимости
10.	description	String	Описание Недвижимости, предоставленное её Менеджером
11.	rating	Float	Общий рейтинг Недвижимости, основанный на

			предыдущих PropertyRatings
12.	pricePerDay	Double	Используется для расчета цены Бронирования сроком менее одной недели или для дополнительных дней при бронировании на более долгий срок
13.	pricePerWeek	Double	Используется для расчета цены Бронирования сроком более 6 дней, но менее одного месяца
14.	pricePerMonth	Double	Используется для расчета цены за полный 30-дневный период
15.	bills	Set	Набор Счётов (Bills), связанных с этой Недвижимостью
16.	tenant	Tenant	Текущий Арендатор, проживающий в этой Недвижимости
17.	photos	List	Список ссылок на фотографии этой Недвижимости

2) Бронирование (Booking)
Основной элемент системы, Бронирование — это центральная точка всего спектра бизнес-процессов, поддерживаемых программным продуктом. Оно создается, когда Арендатор бронирует Недвижимость, и существует до тех пор, пока вся транзакция не будет завершена тем или иным образом.

Поля объекта типа Booking

№	**Имя поля**	**Тип поля**	**Комментарий**
1.	id	Long	ID сущности
2.	property	Property	Забронированная Недывижимость
3.	tenanted	Long	ID Арендатора, сделавшего Бронирование
4.	startDate	Timestamp	Дата начала Бронирования. Для этого можно было использовать LocalDate, но тогда автор этого ещё не знал. НЕОБХОДИМО.
5.	endDate	Timestamp	Аналогично предыдущему полю, но определяет дату окончания Бронирования
6.	isPaid	Boolean	Обозначает, оплачено ли данное Бронирование
7.	status	BookingStatus	Enum, важный для множества процессов. Может быть: • PENDING_APPROVAL — Бронирование еще не подтверждено Менеджером Недвижимости • PENDING_PAYMENT — Бронирование подтверждено, но еще не оплачено • CONFIRMED — Бронирование оплачено Арендатором • CURRENT — Арендатор в данный

			момент проживает в Недвижимости • CANCELLED — Бронирование было отменено Арендатором или автоматически, из-за неуплаты • OVER — Бронирование успешно завершено, и сейчас идет период для подачи Претензий • PROPERTY_LOCKED_BY_MANAGER — специальный статус для случаев, когда Менеджер хочет заблокировать Недвижимость на некоторое время (для этого система создает своеобразное "фиктивное Бронирование")

3) Удобство (Amenity)
Любое Удобство, которым может обладать Недвижимость. Они добавляются в систему Админами, и указываются для объектов Недвижимости соответствующими Менеджерами.

Поля объекта типа Amenity:

№	Имя поля	Тип поля	Комментарий
1.	id	Long	ID сущности
2.	description	String	НЕОБХОДИМО. Краткое описание удобства по сути, например, «бассейн», «кондиционер» и т.д.

4) История аренды (LeasingHistory)
Своего рода "сущность-хранилище". Чтобы избежать перегрузки системы миллионами давно завершенных Бронирований, по окончании претензионного периода Бронирования создается новая сущность «История аренды». Само Бронирование удаляется из системы, а его запись сохраняется в виде Истории аренды.

Поля объекта типа LeasingHistory:

№	Имя поля	Тип поля	Комментарий
1.	id	Long	ID сущности
2.	tenant	Tenant	Сохраняет всю сущность Арендатора — достаточно было бы и ID, но автор решил не переделывать систему из-за нехватки времени
3.	propertyId	Long	Сохраняет ID Недвижимости, которая была забронирована
4,	startDate	Timestamp	Время начала соответствующего Бронирования
5,	endDate	Timestamp	Время окончания соответствующего Бронирования

5) Избранные объекты Арендатора (TenantFavorites)

Как можно понять по названию, это сущность, обеспечивающая хранение и извлечение списка избранных объектов Недвижимости Арендатора.

Поля объекта типа TenantFavorites:

№	Имя поля	Тип поля	Комментарий
1.	id	Long	ID сущности
2.	tenantId	Long	Хранит ID Арендатора, чьи избранные объекты хранятся в следующем поле
3.	favoritePropertyIDs	List	Хранит ID объектов Недвижимости, которые Арендатор отметил кА Избранные

6) Запрос на досрочное прекращение (EarlyTerminationRequest)
Эта сущность создана специально для случаев, когда Арендатор нуждается в досрочном завершении своего Бронирования. В рамках данной платформы, этот запрос может быть принят или отклонён соответствующим Менеджером.

Поля объекта типа EarlyTerminationRequest

№	Имя поля	Тип поля	Комментарий
1.	id	Long	ID сущности
2.	tenantId	Long	ID Арендатора, запрашивающего досрочное прекращение Бронирования
3.	bookingId	Long	Сохраняет ID досрочно прекращаемого Бронирования
4.	terminationDate	LocalDateTime	Указывает, когда именно Арендатор хочет, чтобы его/ее Бронирование было прекращено
5.	Комментарий	String	Арендатор вероятно захочет объяснить причину, по которой он/она хочет досрочно прекратить Бронирование. Это поле служит этой цели
6.	managersResponse	String	Менеджер может либо принять, либо отклонить запрос; в любом случае, ответ Менеджера будет помещен сюда
7.	status	ETRequestStatus	Enum, отражающий стадию обработки запроса. Может быть: PENDING (Ожидает), APPROVED (Одобрено) или DECLINED (Отклонено)
8.	processedAt	LocalDate	Дата, когда запрос был обработан

III. Финансовые сущности:
1) Счет (Bill)

Следует сразу отметить, что это НЕ счет, выставленный Арендатору для оплаты аренды. На самом деле, эта сущность представляет собой некий остаток с тех времен, когда автор считал возможным разработать полнофункциональную систему, охватывающую все аспекты управления недвижимостью. В текущей схеме Счет — это просто любой коммунальный / ипотечный / другой счет, который Менеджер должен оплачивать в связи со своей Недвижимостью. Тем не менее, это предоставляет некоторую степень дополнительной функциональности, позволяя Менеджерам добавлять счета, которые они должны оплачивать в отношении конкретной Недвижимости, и помогает создавать финансовые отчеты для Менеджеров.

Поля объекта типа Bill

№	Имя поля	Тип поля	Комментарий
1.	id	Long	ID сущности
2.	amount	double	Довольно элементарно — это сумма, подлежащая оплате по данному конкретному счету
3.	currency	Currency	Валюта, в которой выставлен счет
4.	property	Property	Недвижимость, к которой относится счет
5.	expenseCategory	String	Должен позволять Менеджерам лучше группировать свои счета
6.	dueDate	Timestamp	Дата, до которой счет должен быть оплачен. Да, я знаю, что это должно быть LocalDate, но это старая настройка, которая пугает автора
7.	recipient	String	Получатель платежа
8.	recipientIBAN	String	IBAN получателя
9.	isPaid	Boolean	Указывает, был ли счет уже оплачен
10.	issuedAt	Timestamp	Дата выдачи счета
11.	addedAt	Timestamp	Время, когда счет был добавлен в систему

2) **Валюта (Currency)**
Автор признаёт, что процессы, связанные с валютой, могут быть доработаны для обеспечения более целостного процесса. Тем не менее, существующая функциональность учитывает валюту, в которой выражена любая денежная сумма: одна из них является базовой валютой системы, а другие облегчают просмотр и оплату для Арендаторов, которые могут предпочитать, чтобы суммы были указаны в валюте, отличной от базовой. Содержит несколько избыточных полей, которые автор все же решил оставить для возможного будущего рефакторинга.

Поля объекта типа Currency:

№	Имя поля	Тип поля	Комментарий
1.	id	Long	ID сущности

2.	designation	String	Обязательно. Трехбуквенное обозначение валюты, например, EUR, USD, THB и т.д.
3.	isBaseCurrency	Boolean	Определяет, является ли эта валюта базовой валютой системы
4.	rateToBase	Double	Курс обмена валюты — 1,00 для базовой валюты, соответствующие ставки для других валют. Могут управляться Администраторами
5.	bills	Set	Не используется в проекте, но сохранено для возможного будущего использования
6.	numericalConfigs	Set	Не используется в проекте, но сохранено для возможного будущего использования
7.	refunds	Set	Не используется в проекте, но сохранено для возможного будущего использования
8.	payouts	Set	Не используется в проекте, но сохранено для возможного будущего использования
9.	tenants	Set	Не используется в проекте, но сохранено для возможного будущего использования
10.	tenantPayments	Set	Не используется в проекте, но сохранено для возможного будущего использования

3) Выплата (Payout)
Сущность отражает фактический платёж, который платформа должна будет выплатить Менеджеру по успешном завершении Бронирования. Это сущности генерируются автоматически, но мгут и создаваться вручную Администрами.

Поля объекта типа Payout:

№	Имя поля	Тип поля	Комментарий
1.	id	Long	ID сущности
2.	managerId	Long	НЕОБХОДИМО. ID Менеджера, которму причитается Выплата
3.	bookingId	Long	НЕОБХОДИМО. ID Бронирования, закоторое делается Выплата
4.	amount	double	НЕОБХОДИМО. Сумма Выплаты
5.	createdAt	Timestamp	Время создания Выплаты
6.	currency	Currency	Вылюта Выплаты (в текущей среде - базовая валюта системы. В будущем может

			быть реализована поддержка Выплат в других валютах)

4) Скидка (PropertyDiscount)
Сущность, используемая в системе для того, чтобы Менеджеры могли устанавливать скидки (например, в целях рекламы или не в сезон) или наценки (в праздничные дни или в сезон) на цену аренды их Недвижимости на конкретные даты или периоды.

Поля объекта типа PropertyDiscount:

№	Имя поля	Тип поля	Комментарий
1.	id	Long	ID сущности
2.	property	Property	Недвижимость, к которой относится скидка
3.	percentage	Double	Отрицательное значение для скидок, положительное для наценок. В пределах от -100 (в этот период Недвижимость можно арендовать бесплатно) до беконечности (стоимость аренды может быть настолько высокой, насоклько этого пожелает Менеджер)
4.	periodStart	LocalDate	Начало периода действия скидки
5.	periodEnd	LocalDate	Конец периода действия скидки
6.	createdAt	Timestamo	Время, когда скидка была добавлена

5) Возврат (Refund)
В некоторых случаях, например, в случае заблаговременной отмены Бронирования или одобренного Запроса на досрочное прекрщение, Арендатор может иметь право на получение Возврата, то есть части его/её изначального Платежа. Эта ситуация обрабатывается посредством сущности «Возврат» (Refund).

Поля объекта типа Refund:

№	Имя поля	Тип поля	Комментарий
1.	id	Long	ID сущности
2.	tenanted	Long	НЕОБХОДИМО. ID Арендатора, которому причитается Возврат
3.	bookingId	Long	НЕОБХОДИМО. ID Бронирования за которое осуществляется Возврат
4.	amount	double	НЕОБХОДИМО. Сумма Возврата
5.	createdAt	Timestamp	Время создания Возврата
6.	currency	Currency	Валюта Возврата (в текущей среде - базовая валюта системы. В будущем может

			быть реализована поддержка Возвратов в других валютах)

6) Платёж (TenantPayment)

Платёж создаётся вместе с Бронированием для последующей обработки всех аспектов платежа, который Арендатор долежн будет внести за бронирование данной Недвижимости. Система получает Платёж от Арендатора, а затем перечисляет его в виде Выплаты Менеджеру за вычетом установленных системой комиссионных.

Поля объекта типа TenantPayment:

№	Имя поля	Тип поля	Комментарий
1.	id	Long	ID сущности
2.	amount	Double	Сумма платежа: Общая денежная сумма, которую необходимо оплатить.
3.	currency	Currency	Валюта платежа: В настоящее время это базовая валюта, используемая в предыдущих транзакциях. В будущем потребуются обновления для обработки нескольких валют.
4.	tenant	Tenant	Арендатор: Арендатор, ответственный за осуществление этого Платежа.
5.	managerId	Long	ID менеджера: Уникальный идентификатор Менеджера, которому причитается Выплата.
6.	associatedPropertyId	Long	ID объекта: Уникальный идентификатор объекта Недвижимости, за который поступает Платёж.
7.	associatedBookingId	Long	ID бронирования: Уникальный идентификатор Бронирования, связанного с этим Платежом.
8.	receivedFromTenant	Boolean	Флажок получения платежа: Логический флажок, указывающий, был ли уже получен платёж от Арендатора.
9.	managerPayment	Double	Сумма выплаты: Сумма, которая будет выплачена соответствующему Менеджеру.
10.	feePaidToManager	Boolean	Флажок выплаты: Логический флажок, указывающий, была ли соответствующая сумма выплачена Менеджеру (платеж считается полностью завершенным, когда оба флажка (8 и 10) равны TRUE).

11.	receiptDue	Timestamp	Срок получения платежа: Время, к которому платеж Арендатора должен быть получен.

IV. Коммуникационные сущности:

1) Претензия (Claim)

Когда Арендатор недоволен предоставленной услугой по какой-либо причине, или Менеджер желает пожаловаться на действия Арендатора, данный Арендатор или Менеджер может подать соответствующую Претензию против своего партнёра в этой конкретной сделке (Бронирование). Претензия обрабатываются Админами; дополнительная функциональность для жалоб будет добавлена в более поздних версиях.

Поля объекта типа Claim:

№	Имя поля	Тип поля	Комментарий
1.	id	Long	ID сущности
2.	bookingId	Long	ID бронирования, в отношении которого подается жалоба
3.	description	String	Здесь истец должен описать суть жалобы
4.	claimantType	ClaimantType	Enum - определяет, является ли истцом Арендатор или Менеджер
5.	claimStatus	ClaimStatus	Enum - жалоба может быть OPEN (ожидает рассмотрения), RESOLVED (разрешена), CLOSED (закрыта по любой причине без разрешения) или WITHDRAWN (если истец её отзывает).
6.	admitted	Boolean	Устанавливается куак TRUE, если предполагаемая сторона, нарушившая правила, признает жалобу без оспаривания
7.	createdAt	Timestamp	Время, когда была подана жалоба
8.	resolvedAt	Timestamp	Время, когда жалоба была разрешена
9.	resolution	String	Разрешение жалобы, своего рода объяснение, данное Админом

2) Сообщение (Message)

Сообщение - сущность, обеспечивающая общение между Арендаторами и Менеджерами на сайте. Упорядоченные Сообщения составляют собой отдельную беседу.

Поля объекта типа Message

№	Имя поля	Тип поля	Комментарий
1.	id	Long	ID сущности
2.	content	String	Сам текст Сообщения
3.	senderId	Long	ID отправителя
4.	receiverId	Long	ID получателя
5.	receiverType	UserType	Определяет, является ли получатель Арендатором или Менеджером. при этом отправитель автоматически идентицифируется, как принадлежащий к «противоположному» типу
6.	sentAt	LocalDateTime	Время, когда было отправлено Сообщение
7.	isRead	Boolean	«Флажок», показывающий, было ли Сообщение прочитано

3) Обзор (Review)
При желании, Арендатор может оставить Обзор на Недвижимость, гнде он останавливался, но только после того, как соответствующее Бронирование будет завершено.

Поля объекта типа Review

№	Имя поля	Тип поля	Комментарий
1.	id	Long	ID сущности
2.	tenantId	Long	ID of the Tenant leaving the Review
3.	propertyId	Long	REQUIRED: ID of the Property being reviewed
4.	text	String	Text of the Review
5.	rating	int	The Tenant’s rating of the Property, on a scale of 1 to 5

4) PropertyRating
The overall rating of a Property is calculated in reliance upon these entities.

Поля объекта типа PropertyRating

№	Имя поля	Тип поля	Комментарий
1.	id	Long	ID сущности
2.	tenanted	Long	ID Арендатора, оценивавшего Недвижимость
3.	propertyId	Long	ID оцениваемой Недвижимости
4.	bookingId	Long	ID Бронирования, по которому Арендатор оцениал Недвижимость
5.	rating	int	Оценка Арендатором Недвижимости по шкале от 1 до 5

5) Рейтинг Арендатора (TenantRating)

Несколько похожая на предыдущую, эта сущность служит в качестве основы для общего рейтинга Арендатора на основании оценок соотетствующих Менеджеров

Поля объекта типа TenantRating

№	Имя поля	Тип поля	Комментарий
1.	id	Long	ID сущности
2.	tenanted	Long	ID оцениваемого Арендатора
3.	bookingId	Long	ID Бронирования, на основании которого Менеджер оценивал Арендатора
5.	rating	Integer	Оценка Арендатора Менеджером, от 1 до 5

V. Вспомогательные сущности:

1) Числовая настройка (NumericalConfig)

Изначально эта сущность предназначалась для обработки скидок и других подобных функций. Только когда проект расширился до базового уровня, из которого произросло то, чем проект является сейчас, автор понял, что скидки должны обрабатываться иным образом, учитывая определенные временные рамки и другие факторы. Тем не менее, именно в это время возникла необходимость в хранении различных числовых данных, особенно для системных настроек, и эта уже существующая сущность оказалась весьма удобной для этой цели. Таким образом, она не была удалена в ходе реструктуризации проекта, и теперь используется для хранения различных числовых значений для системы.

Поля объекта типа NumericalConfig

№	Имя поля	Тип поля	Комментарий
1.	id	Long	ID сущности
2.	name	String	Имя числовой настройки; используется для помощи системе в извлечении необходимой конфигурации из базы данных.
3.	value	Double	Фактическое значение, хранящееся в сущности.
4.	type	NumConfigType	Enum. Может быть DISCOUNT, SURCHARGE, SYSTEM_SETTING, SWITCH или COUNTER. Означает, соответственно, скидку, наценку, системную настройку, переключатель или счётчик. Также используется для целей извлечения данных.
5.	currency	Currency	Устаревшее поле.

12) Связь Удобство-Недвижимость (PropertyAmenity)

Эта сущность служит связующим звеном между Недвижимостью и конкретным Удобством, позволяя присваивать Удобства Недвижимости и извлекать эти ассоциации. Автор признаёт, что этот подход несколько неуклюжий, но это была одна из первых техник операций с базами данных, которые нам когда-либо демонстрировали на занятиях, и эта сущность была одной из первых, созданных в системе, поэтому, чтобы избежать серьёзных изменений, она была сохранена.

Поля объекта типа PropertyAmenity:

№	Имя поля	Тип поля	Комментарий
1.	id	Long	ID сущности
2.	property_id	Long	ID Недвижимости, которой присваивается Удобство
3.	amenity_id	Long	ID присвыаиваемого Удобства

3) Сущность для сброса токена (TokenResetter)
Ещё одна вспомогательная сущность. Содержащаяся здесь отметка о времени определяет время, в течение которого то\кен существует, что, в свою очередь, позволяет специальной плановой задаче удалять просроченные токены, тем самым препятствуя попыткам доступа с таковыми и помогая очищать базу данных.

Поля объекта типа TokenResetter

№	Имя поля	Тип поля	Комментарий
1.	id	Long	ID сущности
2.	userId	Long	ID пользователя (Арендатора или Менеджера), чей токен необходимо сбросить
3.	userType	UserType	Перечисляемая сущность – здесь может быть АРЕНДАТОР или МЕНЕДЖЕР
4.	createdAt	Timestamp	Время создания токена и соответствующей сущности типа TokenResetter

4) Картирование числоых данных (NumericDataMapping)
Эта сущность содержит трехуровневую карту, которая хранит ключи (SecretKeys) для зашифрованных алгоритмом AES номеров платежных карт пользователей и кодов CVV на нижнем уровне. Средний уровень хранит тип пользователя, а верхний уровень содержит идентификатор пользователя. Однако, проконсультировавшись с экспертом из существующей компании-разработчика, автор отказался от идеи хранить эти ключи в одной базе данных с самимизашифрованными значениями. Соответствующие механизмы были переработаны в соответствии с этим советом, и, в результате, такие сущности всё ещё создаются, однако они не сохраняются в базе данных. Вместо этого они преобразуются в текстовые строки, зашифрованные с использованием разработанного автором механизма шифрования (хотя и довольно простого, основанного на случайных сдвигах ASCII-значений) и записываются в отдельный файл, откуда они извлекаются в случае необходимости использования платёжных данных пользователя. Этот

процесс также требует посредничества сущности KeyLink, описанной далее.

Поля объекта типа NumericDataMapping:

№	Имя поля	Тип поля	Комментарий
1.	id	Long	ID сущности
2.	map	Map<Long, Map<UserType, Map<Boolean, SecretKey>>>	Вышеописанная трёхурровнеая структура: Уровень 1: ключ типа Long: ID пользовательской сущности Уровень 2: ключ типа UserType: тип пользователя Уровень 3: ключ типа Boolean: true означает ключ для номера карты, false - для CVV-кода Значение SecretKey: ключ

5) Связка для ключей (KeyLink)

KeyLink — это вспомогательная сущность, предназначенная для упрощения доступа к внешне хранимым зашифрованным значениям SecretKey. Чтобы избежать перегрузки файла хранения и снизить риск потери или кражи данных, зашифрованные записи SecretKey ограничены 100 записями на файл. Сущность KeyLink позволяет программе идентифицировать файл, в котором хранятся секретные ключи конкретного пользователя.

Поля объекта типа KeyLink:

№	Имя поля	Тип поля	Комментарий
1.	id	Long	ID сущности
2.	userId	Long	ID пользователя (Арендатор или Менеджер), чьи ключи хранятся в файле
3.	userType	UserType	ENUM, указывающий тип пользователя: АРЕНДАТОР или МЕНЕДЖЕР
4.	fileNumber	Integer	Номер файла, в котором хранится запись

6) Закрытие Недвижимости (PropertyLock)

Когда Менеджер желает заблокировать свою Недвижимость на определённый период времени, создаётся «заглушка» в виде объекта Бронирования, чтобы заблокировать соответствующий срок, тем самым делая Недвижимость недоступной для аренды в этот период. Однако поскольку Менеджеры не могут отменять Бронирования, необходима специальная сущность PropertyLock, чтобы позволить им снять свои собственные блокировки.

Поля объекта типа PropertyLock

№	Имя поля	Тип поля	Комментарий
1.	id	Long	ID сущности
2.	bookingStubId	Long	ID Бронирования-« заглушки», блокирующего

			соответствующую Недвижимость
3.	propertyId	Long	ID Недвижимости, о которой идет речь

Помимо того, есть три таблицы, связывающие полномочия с пользователями; однако, поскольку они служат только для проверки доступа, они не участвуют в бизнес-логике.

Общая схема базы данных, получающейся в результате взаимодействия сущностей, такова:
https://dbdiagram.io/d/PROPMAN_DATABASE-66e93f9ba0828f8aa61a0251

4. Типы пользователей и работа системы

В этом разделе описываются типы и функции пользователей в системе, формируя общее представление о нормальном ходе работы системы. Это общие описания; для подробного анализа всех функций, пожалуйста, обращайтесь к автору.

Пользователи платформы делятся на три типа: арендаторы, менеджеры и администраторы. Эти типы не иерархические; каждый из этих типов пользователей подразумевает определённый набор доступных функций. Следовательно, функционал можно разделить на шесть категорий:

1. Общие функции (как правило, доступные всем, включая неавторизованных пользователей)
2. Функции авторизации
3. Функции арендаторов
4. Функции менеджеров
5. Функции администраторов
6. Вспомогательные функции (плановые или стартовые задачи, запускаемые без вмешательства пользователя)

Общие функции:
Помимо авторизационных действий, неавторизованные пользователи имеют доступ к следующим функциям:

- Поиск объектов Недвижимости с указанием ряда критериев
- Просмотр информации об объекте Недвижимости
- Чтение Отзывов об объекте Недвижимости
- Получение общедоступной информации о Менеджере Недвижимости
- Получение галереи объектов Недвижимости конкретного Менеджера

Функции авторизации
Эти функции позволяют пользователям регистрироваться, входить в систему в доступной степени управлять своими учетными данными. Реализованы следующие функции, связанные с доступом:

- Регистрация в качестве Арендатора
- Регистрация в качестве Менеджера
- Вход в систему (общая конечная точка для Арендаторов и Менеджеров)
- Отдельная конечная точка для входа Админов
- Процедуры проверки OTP для Арендаторов, Менеджеров и Админов
- Подтверждение регистрации по электронной почте (для Арендаторов и Менеджеров)
- Подтверждение изменения адреса электронной почты
- Смена пароля (пользователь к этому моменту должен быть авторизован в системе)
- Сброс пароля (две конечные точки: одна для запроса, другая для завершения)

Функции Арендатора
Функционал Арендаторов построен вокруг процесса аренды доступных объектов Недвижимости. Помимо возможности поиска и просмотра объектов Недвижимости иМменеджеров (общие функции), Арендаторам предоставляется следующий набор функций:

- Просмотр и редактирование своих профилей
- Бронирование и отмена Бронирований объектов Недвижимости
- Сохранение, получение и удаление своих любимых объектов из списка избранного
- Выполнение Платежей, просмотр завершённых и непогашенных платежей
- Просмотр своих Бронирований
- Отправка и получение Сообщений от Менеджеров
- Подача Запросов на досрочное прекращение бронирования
- Оценка объектов Недвижимости
- Предоставление отзывов об объектах Недвижимости
- Направление Претензий

Функции Менеджеров
Менеджерам предоставляется ряд функций, облегчающих управление их арендными объектами, управление соответствующими бронированиями, общение с арендаторами и получение выплат. Конкретно, функционал Менеджеров составляют следующие функции:

- Просмотр и редактирование своих профилей
- Просмотр своих Бронирований — всех, только тех, которые относятся к конкретному объекту (недвижимости), или любых Бронирований, ожидающих утверждения с их стороны
- Просмотр профилей Арендаторов, бронирующих их объекты
- Добавление объектов в систему
- Обновление данных о своих объектах
- Блокировка объектов для бронирования на определённый период, и разблокировка любых ранее заблокированных объектов
- Управление удобствами, относящимися к их объектам

- Загрузка или удаление фотографий своих объектов
- Удаление своих объектов из системы
- Установка и сброс скидок или наценок для своих объектов на определённые периоды
- Добавление и удаление счетов, относящихся к их объектам, просмотр всех непогашенных счетов или только тех, которые относятся к конкретному объекту
- Генерация финансовых отчетов для Менеджеров
- Утверждение или отклонение Бронирований их объектов, сделанных Арендаторами
- Принятие или отклонение Запросов на досрочное прекращение бронирования от Арендаторов
- Отправка и получение Сообщений от Арендаторов
- Оценка Арендаторов, проживающих в их объектах
- Направление Претензий

Фугнкции Админов

Как правило, Админы не участвуют в основном бизнес-процессе; вместо этого им предоставляется ряд функций, необходимых для контроля работы системы и выполнения ручного вмешательства при необходимости. Все Админы могут:

- Переключать активный статус любого Арендатора или Менеджера, тем самым блокируя или восстанавливая соответствующих пользователей
- Регистрировать новых Арендаторов и Менеджеров, обновлять их информацию по запросу соответствующих пользователей или по другим причинам, а также удалять эти учетные записи пользователей, если это необходимо
- Добавлять и удалять объекты Недвижимости
- Добавлять новые Валюты, изменять курсы Валют или устанавливать новую базовую Валюту
- Добавлять новые типы Удобств в систему или удалять их при необходимости
- Редактировать системные настройки (Числовые Настройки)
- Просматривать и разрешать Претеннии
- Вручную создавать, просматривать и закрывать Выплаты и Возвраты
- Удалять Отзывы, если это необходимо

Кроме того, имеется «наивысшая» пользовательская сущность DefaultAdmin, которую нельзя удалить из системы обычными средствами, и которая, помимо вышеописанных функций, может:

- Создаватьи удалять аккаунты других Админов

Вспомогательные функции

Эти функции в основном не требуют взаимодействия с пользователем и предназначены для упрощения работы платформы, запускаясь при старте или как запланированные задачи:

- Создание нового DefaultAdmin при первом запуске, если его нет, и запрос новых данных DefaultAdmin через консоль
- Генерация образца базы данных — если это запрошено через ввод в консоль, программа перезапишет существующую базу данных и создаст новую, заполнённую рядом сущностей. Это очень полезно для тестирования. Тестовую базу данных также легко восстанавливать при необходимости. Поскольку это действие удаляет любые старые записи базы данных, оно требует ввода пароля DefaultAdmin
- Изменение статуса Бронирований в соответствии с текущей датой
- Удаление уже обработанных или просроченных Запросов на досрочное прекращение бронирования
- Отмена Бронирований, которые не были оплачены вовремя
- Очистка просроченных Токенов
- Закрытие завершённых Бронирований, если нет Претензий, и преобразование их в Истории аренды
- Удаление устаревших Скидок
- Напоминание Арендаторам о приближающихся сроках Платежей по электронной почте
- Напоминание пользователям о сроках действия их платежных карт
- Напоминание Менеджерам о приезде гостей
- Удаление устаревших сущностей Блокировки объектов
- Удаление старых разрешённых Претензий

5. Применённые технологии

В данном разделе представлен краткий список технологий, использованных в ходе разработки.

Spring Boot (Web, Security, Data JPA, Thymeleaf)
Hibernate (для реализации JPA)
Jakarta (управление данными, валидация)
MySQL (основная база данных, с H2 для тестирования)
Swagger (документация API)
Lombok (генерация кода)
MapStruct (маппинг объектов)
JWT (аутентификация)
Apache POI (работа с Excel-файлами)
JUnit, Hamcrest (фреймворк для тестирования)
Jackson: «ручная» работа с JSON (сериалайзеры для объектов времени и других)
SLF4J и Log4j: для логирования
Java Cryptography Extension (JCE): для работы с ключами шифрования
JavaMail — для отправки сообщений по электронной почте
BCrypt для хеширования паролей

AES для шифрования чувствительных данных
Docker (контейнеризация)

6. Возможности дальнейшего улучшения

В этом разделе представлены предложения по возможному дальнейшему развитию данного выпускного проекта.

Автор намерен развивать этот проект для своего портфолио и после защиты, поэтому основным направлением стратегии дальнейшего улучшения станет добавление компонентов, которых на данный момент не хватает. Среди этих элементов будут модульные тесты для всех методов, реализующих непосредственную бизнес-логику (как уже говорилось, ранее написанные тесты пришлось удалить, так как они устарели по мере развития проекта). Далее усилия будут сосредоточены на рефакторинге кода для облегчения взаимодействия с базой данных.

Если оптимизированный бэкенд-компонент будет признан пригодным для развертывания в реальной среде, потребуется небольшая команда для разработки фронтенд-компонента для настольных и мобильных устройств, а также для проведения полноценных тестов всех аспектов итогового продукта с акцентом, в том числе, на вопросы эффективности и безопасности. В случае успеха продукт может быть предложен как готовое решение для частных лиц и компаний, работающих в сфере аренды недвижимости, либо развиваться далее с охватом более широкого круга вопросов управления недвижимостью.

7. Опыт разработки

Личные впечатления автора, которыми он хотел бы поделиться в отношении процесса разработки.

Поскольку проект в основном предназначен для образовательных целей, естественно, первое, о чем следует упомянуть, — это его ценность для самого автора с точки зрения обучения. Проект предоставил возможность как применить навыки, полученные на предыдущем этапе учебы автора, так и опробовать несколько технологий, которые пришлось изучить специально для достижения целей выпускной работы. В обоих случаях автору пришлось преодолевать множество препятствий, что определенно улучшило его навыки решения проблем, и это может оказаться ценным при столкновении с аналогичными или схожими задачами в реальных ситуация, связанных с программированием.

Помимо того, приобретённый опыт выходит далеко за рамки простого повышения уровня владения программированием. Автору выпала возможность на практике осознать важность предварительного планирования и составления графика работы. Отсутствие таковых на начальном этапе этого конкретного проекта привело к значительной потере времени и усилий; в результате процесс разработки занял

гораздо больше времени, чем ожидалось, а сам проект пришлось неоднократно реструктурировать. Некоторые реализации, которые нельзя назвать оптимальными, являются наследием более ранних схем проекта и, возможно, снижают эффективность работы. Тем не менее, следует отметить, что автор вряд ли смог бы составить чёткий и жизнеспособный план разработки проекта без опыта, полученного в ходе решения вышеупомянутых проблем. Поэтому, хотя результат далеко не идеален, он, несомненно, положительно скажется на всех будущих проектах, в которых автор примет участие.

Еще один важный аспект, который стоит отметить - возможность попрактиковаться в соблюдении требований, приближённых к реальным сценариям. Хорошим примером можно считать эволюцию механизмов хранения данных банковских карт. Изначально это были всего лишь поля соответствующих сущностей. После получения совета о том, что это чувствительные данные, непригодные для хранения в незашифрованном виде, автор внедрил механизмы их хранения с использованием алгоритма AES. Впоследствии, после комментария от одного авторитетного специалиста в отрасли, который указал на недостатки подхода, при котором зашифрованные значения и секретные ключи хранятся в одной базе данных, автор изменил систему хранения ключей, внедрив их дальнейшее шифрование с помощью созданного им самим механизма и их сохранение в отдельных файлах. Хотя реальные меры безопасности в этой области, вероятно, потребовали бы ещё более сложного подхода, этот шаг стал важной вехой в образовательном процессе автора.

И, конечно же, данное руководство нельзя завершить без упоминания опыта, полученного автором в плане собственно обучения и решения проблем. К концу разработки автор существенно улучшил свои навыки поиска решений, используя такие ресурсы, как ИИ, доступные текстовые и мультимедийные материалы по конкретным вопросам, а также бесценную помощь профессионалов, которые помогали автору в поиске решений или способствовали образовательному процессу, который автор прошел перед началом работы над проектом. В связи с этим автор считает уместным завершить руководство словами благодарности и признательности всем преподавателям Tel-Ran и внешним консультантам, оказавшим свою помощь в реализации проекта.