Факультет
«Программной Инженерии и Компьютерной Техники»

Информационные системы и Базы данных

Курсовая работа

«Контроль ядерного реактора в Minecraft»

Преподаватель: Николаев Владимир Вячеславович

Выполнил: Андросов Иван Сергеевич

Группа: P33111

Санкт-Петербург,

2023 год

Описание предметной области

Темой данной курсовой работы является автоматизированное управление ядерным реактором из мода Industrial Craft 2 в игре Minecraft.

Ядерный реактор в моде Industrial Craft 2 (ic2) представляет собой сложную систему, которая позволяет игрокам генерировать большое количество энергии, используя силу ядерного деления. Реактор состоит из нескольких компонентов, включая теплоотводы, теплообменники и топливные стержни.

Топливные стержни являются сердцем реактора и отвечают за производство энергии. Эти стержни изготавливаются из комбинации урана и других материалов, и после помещения в реактор они начинают выделять энергию и тепло по мере своего распада с течением времени.

Для того чтобы и без того не простой контроль над состоянием реактора не стал слишком запутанным, схема управления над выделяемым теплом была упрощена до двух состовляющих: теплоотводов и теплообменников.

Теплообменники используются для балансировки тепла между соседними компонентами и корпусом реактора.

Теплоотводы используются только для охлаждения корпуса.

Существуют варианты схем расположения компонентов, в которых не бывает перегрева ядра. В таких схемах главной проблемой является необходимость постоянного слежения за состоянием топлива – если оно истощается, выработка энергии сокращается вплоть до полной остановки. В таком случае необходимо заменить топливные компоненты.

Когда реакторов становится много, постоянная слежка за топливом и его замена становится крайне утомительными, моя курсовая работа призвана автоматизировать эти процессы.

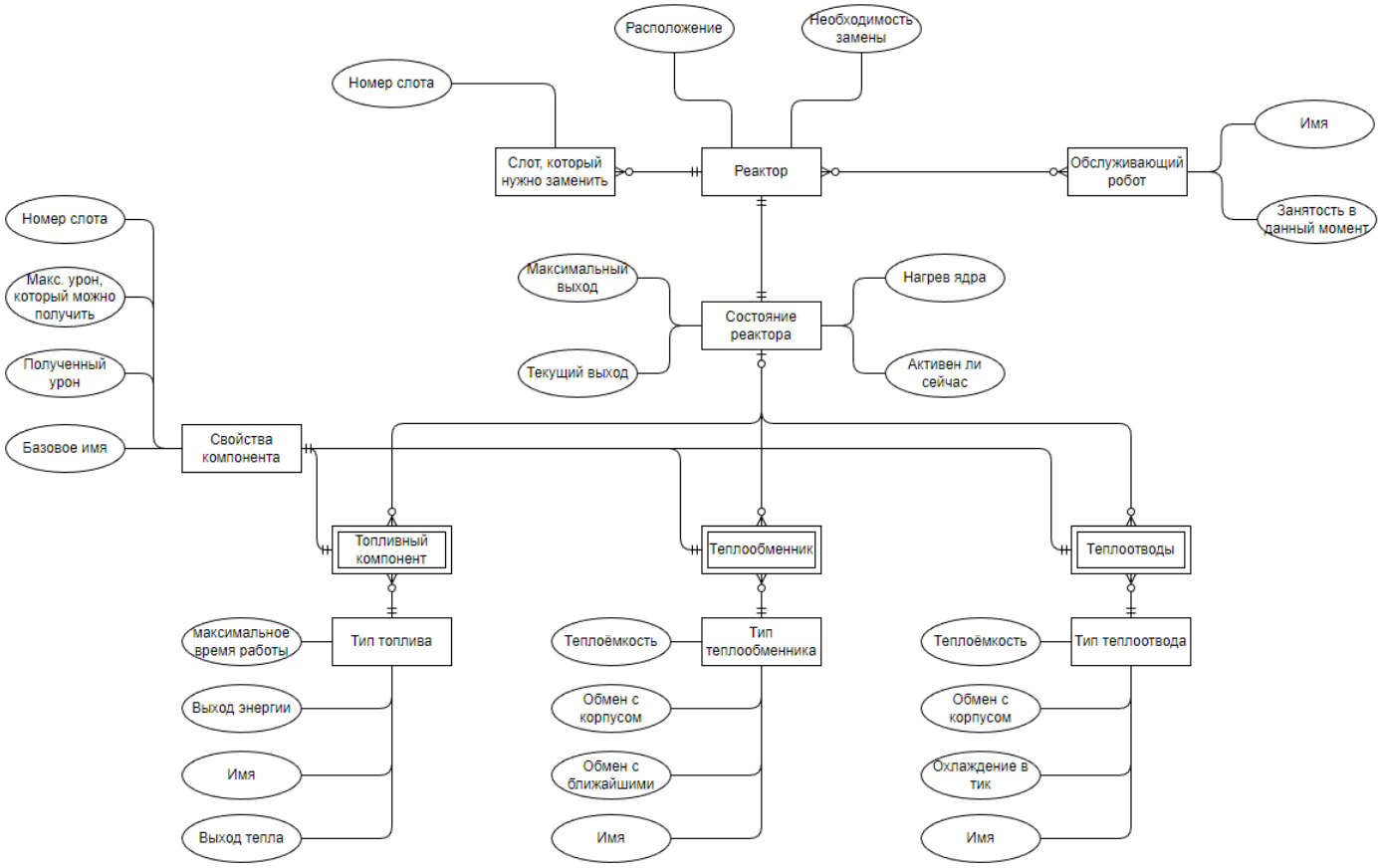
В данном случае для инициализации и контроля параметров компонентов реактора используются комьютеры из мода Open Computers. Они отслеживают состояния топливных компонентов и отправляют их на сервер Spring Boot.

Сервер Spring Boot - это сервер приложений, который используется для обработки данных, полученных от компьютера из Open Computers. Он принимает состояния компонентов и сохраняет их в базе данных Postgres. Если топливо в реакторе кончается, компьютер сообщает об этом серверу, который затем назначает робота для замены истощившегося топлива.

База данных Postgres используется для хранения состояний топливных компонентов, полученных от сервера Spring Boot. В данной системе она является необходимой, так как позволяет хранить большие объемы данных и обеспечивать быстрый доступ к ним.

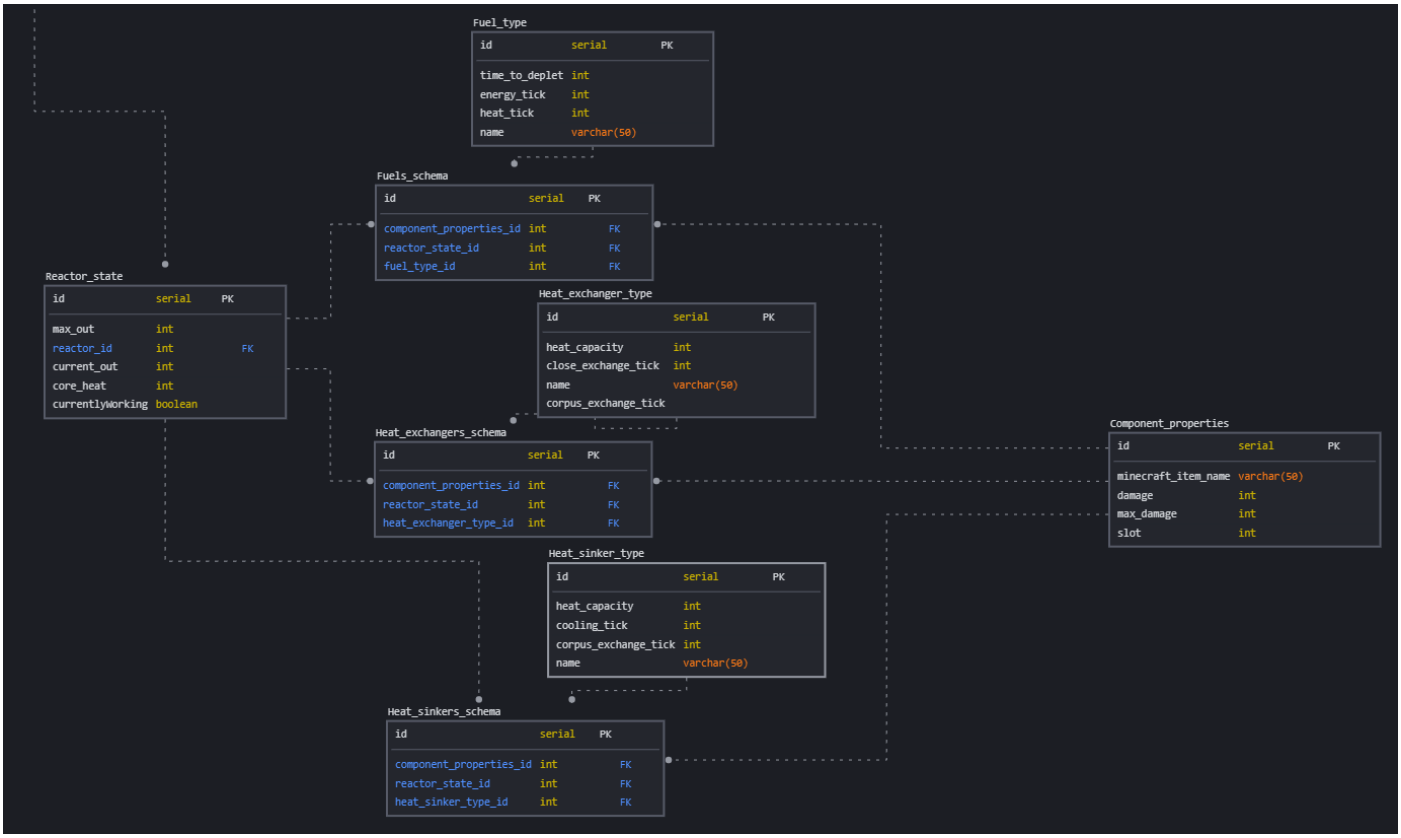
Кроме того, отслеживать состояние реактора можно также на сайте под управлением фреймворка React JS, непрерывно получающем информацию от сервера.

ER-диаграмма

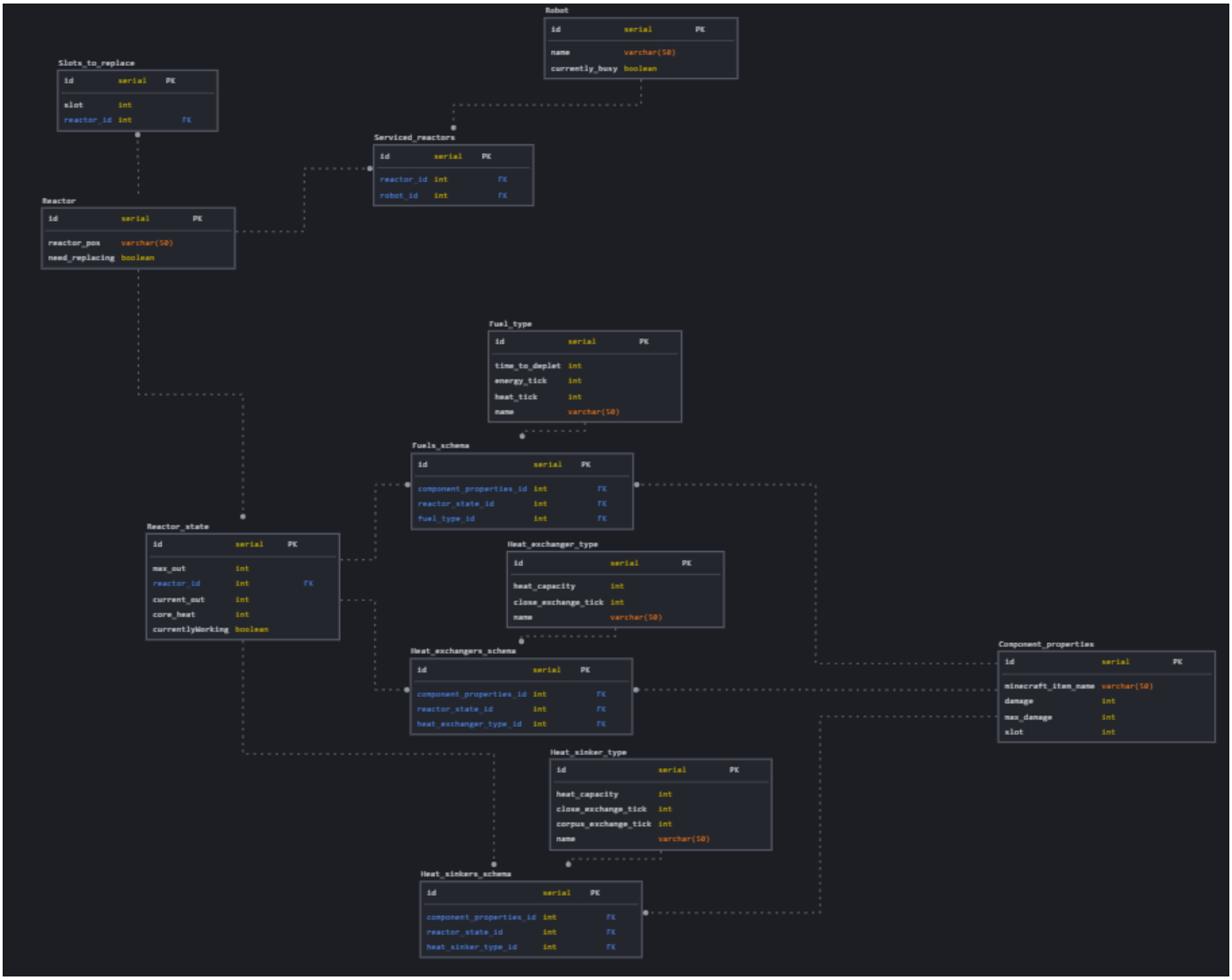


Даталогическая модель

Модель реактора и обслуживающих роботов



Модель состояния реактора



Модель целиком

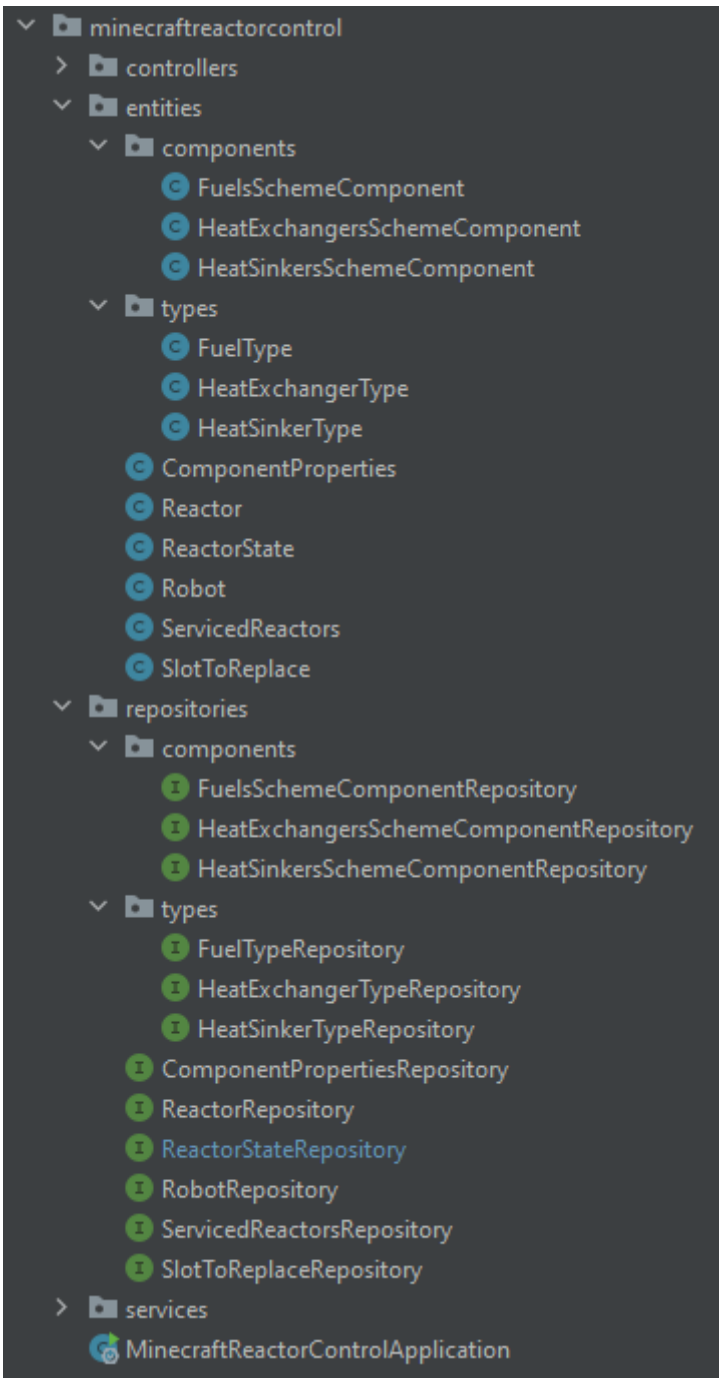
DDL

```
CREATE TABLE public.component_properties (
  id INTEGER PRIMARY KEY NOT NULL,
  damage REAL,
  max_damage REAL,
  minecraft_item_name CHARACTER varying(255),
  slot INTEGER
);

CREATE TABLE public.fuel_type (
  id INTEGER PRIMARY KEY NOT NULL,
  energy_tick INTEGER,
  heat_tick INTEGER,
  NAME CHARACTER varying(255),
  time_to_deplet INTEGER
);

CREATE TABLE public.fuels_scheme (
  id INTEGER PRIMARY KEY NOT NULL,
  component_properties_id INTEGER,
  fuel_type_id INTEGER,
  reactor_state_id INTEGER,
  FOREIGN KEY (component_properties_id) REFERENCES public.component_properties (id)
  MATCH SIMPLE ON UPDATE NO ACTION ON DELETE NO ACTION,
  FOREIGN KEY (reactor_state_id) REFERENCES public.reactor_state (id)
  MATCH SIMPLE ON UPDATE NO ACTION ON DELETE NO ACTION,
  FOREIGN KEY (fuel_type_id) REFERENCES public.fuel_type (id)
  MATCH SIMPLE ON UPDATE NO ACTION ON DELETE NO ACTION
);

CREATE TABLE public.heat_exchanger_type (
  id INTEGER PRIMARY KEY NOT NULL,
  close_exchange_tick INTEGER,
  corpus_exchange_tick INTEGER,
  heat_capacity INTEGER,
  NAME CHARACTER varying(255)
);

CREATE TABLE public.heat_exchangers_scheme (
  id INTEGER PRIMARY KEY NOT NULL,
  component_properties_id INTEGER,
  heat_exchanger_type_id INTEGER,
  reactor_state_id INTEGER,
  FOREIGN KEY (component_properties_id) REFERENCES public.component_properties (id)
  MATCH SIMPLE ON UPDATE NO ACTION ON DELETE NO ACTION,
  FOREIGN KEY (heat_exchanger_type_id) REFERENCES public.heat_exchanger_type (id)
  MATCH SIMPLE ON UPDATE NO ACTION ON DELETE NO ACTION,
  FOREIGN KEY (reactor_state_id) REFERENCES public.reactor_state (id)
  MATCH SIMPLE ON UPDATE NO ACTION ON DELETE NO ACTION
);

CREATE TABLE public.heat_sinker_type (
  id INTEGER PRIMARY KEY NOT NULL,
  cooling_tick INTEGER,
  corpus_exchange_tick INTEGER,
  heat_capacity INTEGER,
  NAME CHARACTER varying(255)
);
```

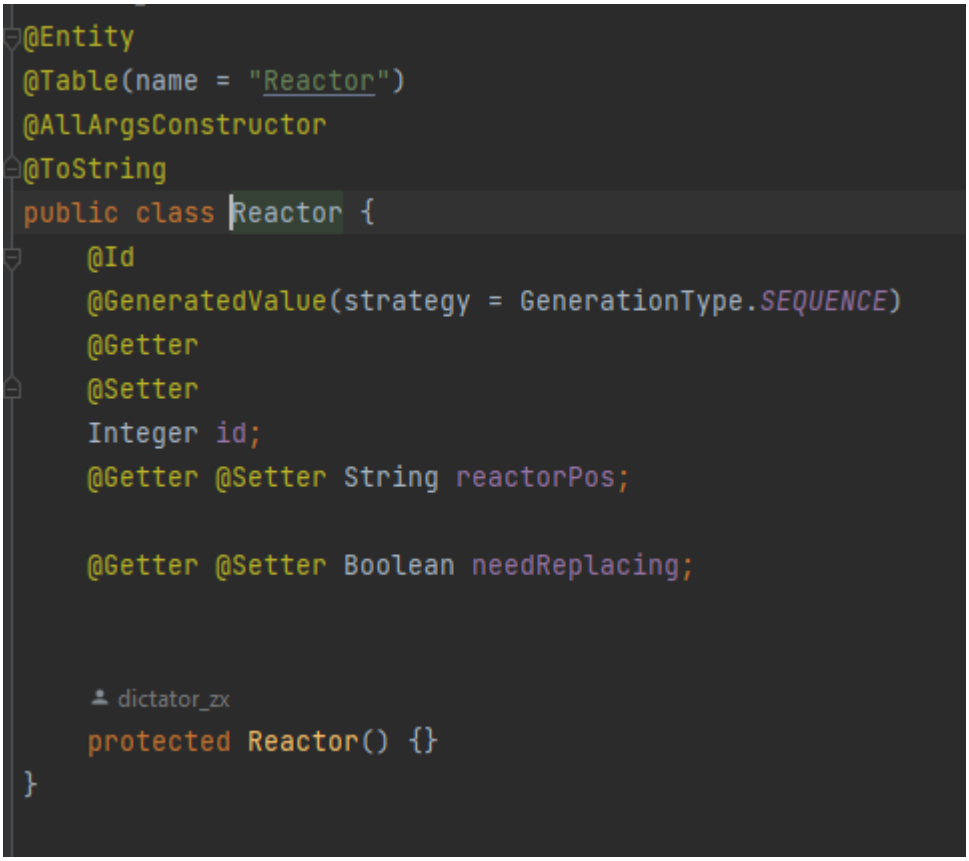
```
CREATE TABLE public.heat_sinkers_scheme (
  id INTEGER PRIMARY KEY NOT NULL,
  component_properties_id INTEGER,
  heat_sinker_type_id INTEGER,
  reactor_state_id INTEGER,
  FOREIGN KEY (component_properties_id) REFERENCES public.component_properties (id)
  MATCH SIMPLE ON UPDATE NO ACTION ON DELETE NO ACTION,
  FOREIGN KEY (heat_sinker_type_id) REFERENCES public.heat_sinker_type (id)
  MATCH SIMPLE ON UPDATE NO ACTION ON DELETE NO ACTION,
  FOREIGN KEY (reactor_state_id) REFERENCES public.reactor_state (id)
  MATCH SIMPLE ON UPDATE NO ACTION ON DELETE NO ACTION
);

CREATE TABLE public.reactor (
  id INTEGER PRIMARY KEY NOT NULL,
  need_replacing BOOLEAN,
  reactor_pos CHARACTER varying(255)
);

CREATE TABLE public.reactor_state (
  id INTEGER PRIMARY KEY NOT NULL,
  core_heat INTEGER,
  current_out INTEGER,
  currently_working BOOLEAN,
  max_out INTEGER,
  reactor_id INTEGER,
  FOREIGN KEY (reactor_id) REFERENCES public.reactor (id)
  MATCH SIMPLE ON UPDATE NO ACTION ON DELETE NO ACTION
);

CREATE TABLE public.robot (
  id INTEGER PRIMARY KEY NOT NULL,
  currently_busy BOOLEAN,
  NAME CHARACTER varying(255)
);

CREATE TABLE public.serviced_reactors (
  id INTEGER PRIMARY KEY NOT NULL,
  reactor_id INTEGER,
  robot_id INTEGER,
  FOREIGN KEY (robot_id) REFERENCES public.robot (id)
  MATCH SIMPLE ON UPDATE NO ACTION ON DELETE NO ACTION,
  FOREIGN KEY (reactor_id) REFERENCES public.reactor (id)
  MATCH SIMPLE ON UPDATE NO ACTION ON DELETE NO ACTION
);

CREATE TABLE public.slots_to_replace (
  id INTEGER PRIMARY KEY NOT NULL,
  slot INTEGER,
  reactor_id INTEGER,
  FOREIGN KEY (reactor_id) REFERENCES public.reactor (id)
  MATCH SIMPLE ON UPDATE NO ACTION ON DELETE NO ACTION
);
```

Управление, созданием/удалением объектов/таблиц, а также обеспечение целостности базы данных было реализованно с использованием Spring Data JPA.

Перечисление классов и интерфейсов, отвечающих за управление бд:



Пример сущности «Реактор»:



Весь исходный код, включая скрипты для Open Computers можно найти по ссылке:

https://github.com/HaCK3R322/Minecraft_ReactorControl

Самые частые запросы и индексация

Самым частым запросом является запрос на обновление свойств компонента в скрипте слежения за топливом:

```
for index, depletedComponent in ipairs(depletedComponents) do
    local slot = depletedComponent.properties.slot

    local componentProperties = reactorComponents.getFromSlot(slot)
    componentProperties.id = parse.parseInt(internet.request(urls.createComponentProperties, componentProperties))
```

Он посылается каждые 5 секунд в бесконечном цикле.

Так как на стороне бэкэнда обновление происходит через метод save Crud репозитория, было принято решение проиндексировать ключевой аттрибут id обновляемой сущности:

```
15 usages   dictator_zx *
@Entity
@Table(name = "Component_properties", indexes = {@Index(columnList = "id",
        name = "id_index")})
@AllArgsConstructor
@ToString
public class ComponentProperties {
    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE)
    @Getter @Setter Integer id;
    @Getter @Setter String minecraftItemName;
    @Getter @Setter Float damage;
    @Getter @Setter Float maxDamage;
    @Getter @Setter Integer slot;


    dictator_zx
    protected ComponentProperties() {}
}
```

Плюсы индексации:

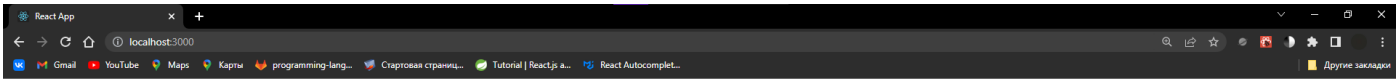
1. Ускорение запросов по идентификатору: Индекс на поле id ускоряет поиск и выборку записей из таблицы Component_properties по их идентификатору. Это особенно важно, когда таблица содержит большое количество записей, поскольку без индекса поиск занимает больше времени, чем с индексом.

2. Уменьшение нагрузки на базу данных: Индексация поля id позволяет базе данных быстрее обрабатывать запросы на выборку записей по их идентификатору, что уменьшает нагрузку на базу данных и повышает производительность.

3. Поддержка уникальности идентификатора: Поле id является первичным ключом таблицы Component_properties. Индекс на этом поле также гарантирует уникальность идентификатора для каждой записи в таблице.

Веб-приложение

Веб приложение работает на базе фреймворка React JS.
Позволяет в любой момент увидеть состояние реакторов и их компонентов, без обязательства подходить лично к каждому из реакторов.



Исходный код приложения также хранится в гит-репозитории по ссылке выше.