Университет ИТМО

Факультет Программной инженерии и компьютерной техники

Курсовая работа

по дисциплине “Информационные системы и базы данных”

Выполнил Керенский Н.А.

Группа: P33012

г. Санкт-Петербург

2021 г

Во вселенной Саги о ведьмаке, существуют ведьмаки, специально подготовленные наёмники, выполняющий заказы на уничтожение монстров. Ведьмаки умеют создавать масла, наносимые на меч, и противоядия направленные на конкретных монстров. Действие разворачивается в вымышленной стране Темерии.

**Доступные заказы**

Список состоит из старых и новых заказов. Новые заказы появляются после охоты на монстра. Новый заказ составляется таким образом: выбирается монстр со случайными характеристиками и количество примет, награда за заказ зависит от количества примет и силы противника. У заказа с большой наградой всегда сильный противник. В списке всегда есть 1 легкий(малая награда, много примет, слабый противник) 1 средний(все средне) 1 сложный(большая награда, мало примет, сильный противник).

**Заказ и Торг**

Ведьмак может выбрать любой заказ из списка доступных заказов. У заказов есть награда и список примет монстра. Чем сложнее бой тем больше будет награда и тем меньше будет примет монстра. Заказы выдают специальные люди. Награду выдают в темерских оренах. С выдавателями заказов можно торговаться и пытаться увеличить награду за заказ. У ведьмака есть характеристика харизмы, чем выше харизма, тем больше вероятность повысить награду за заказ. Вероятность успеха всегда больше нуля и меньше единицы. Если попытка повысить успешна, то харизма увеличивается на 1 и награда увеличивается на 20%, иначе харизма уменьшается на 1, а награда не изменяется.

**Расследование**

Когда ведьмак получает заказ, он может начать расследование. В заказе будут даны характеристики монстра(например: цвет, высота, количество рук). Ведьмак будет пытаться по характеристикам определить вид монстра(вампиры/призраки/духи стихий/трупоеды и тд). Если однозначно определить не удается, то ведьмак случайно выбирает из подходящих.

**Подготовка**

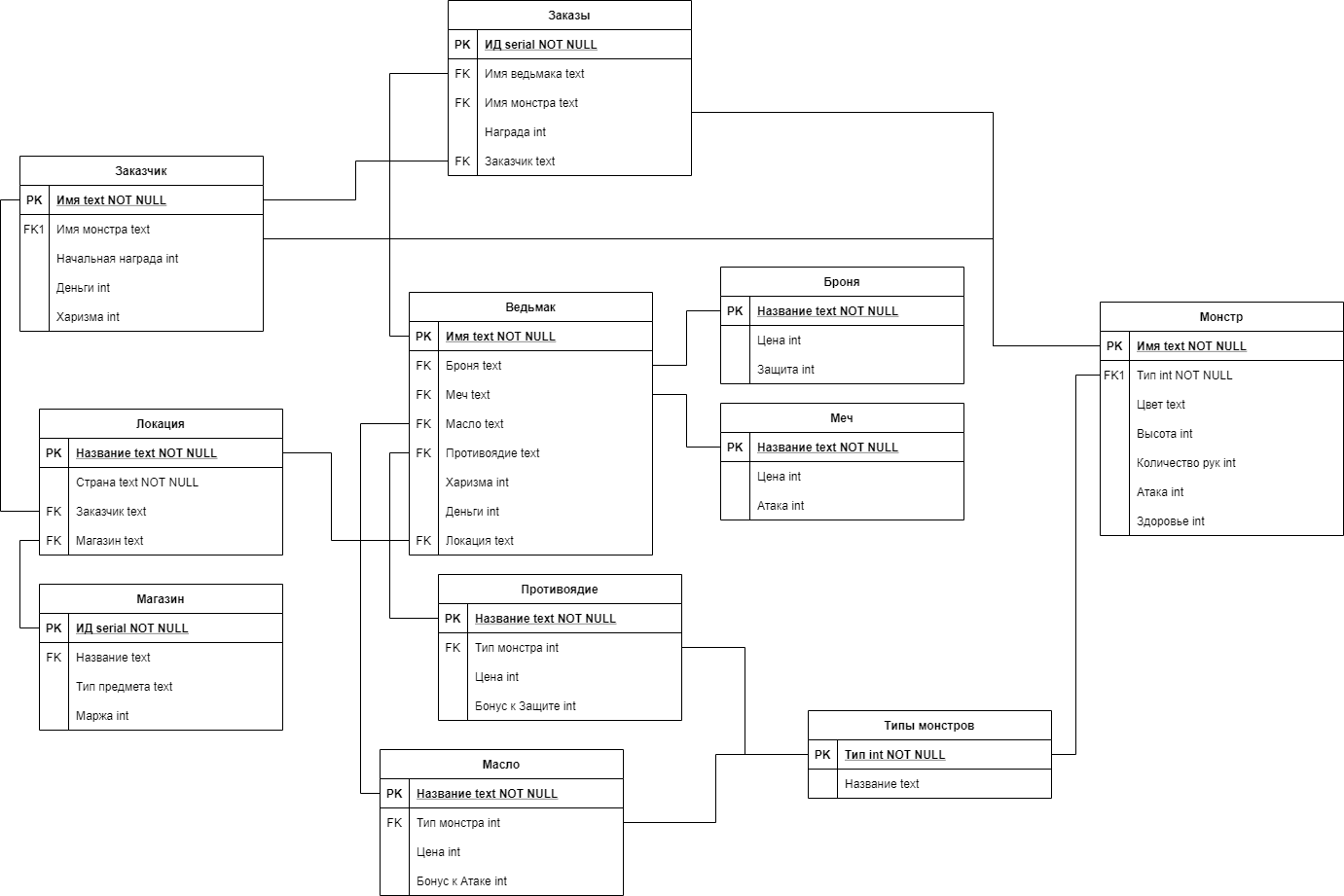
После определения (удачного или неудачного) идет подготовка к сражению, покупка ингредиентов, создание масел и зелий. Для каждого вида монстров существуют уникальные масла и зелья. Для различных видов монстров, средняя цена ингредиентов различна. Масла наносятся на меч и дают временный(на 1 бой) бонус к атаке. Зелья дают временный(на 1 бой) бонус к защите. Чем дороже ингредиенты, тем больше бонус. На 1 бой можно использовать только 1 масло и только 1 зелье.

**Охота**

Затем идет охота на монстра. У ведьмака есть постоянное значение атаки и защиты, которое зависит от снаряжение. Если ведьмак правильно определил вид монстра, то он получает бонус от масла и зелья. Ведьмак побеждает, если его урон больше защиты монстра и его защита больше атаки монстра. Если атака монстра меньше защиты ведьмака, а атака ведьмака меньше защиты монстра, то их значения удваиваются, и условие победы снова проверяется. Если ведьмак побеждает, то он получает награду за заказ. Если Ведьмак проигрывает, то он не получает награду. Заказ исчезает из доступных заказов.

**Отдых**

Ведьмак на полученные деньги может купить улучшенное снаряжение, тем самым постоянно увеличив свою атаку и защиту. Снаряжение состоит из брони и меча. Броня влияет на защиту, а меч влияет на атаку. Ведьмак может купить только то снаряжение, на которое ему хватает денег.



**3 этап**

На данном этапе необходимо было реализовать ER-модель в реляционной СУБД PostgreSQL. Разработанный мной скрипт для последовательного удаления таблиц, создания их и заполнения данными представлен ниже. Также в рамках данного этапа я реализовал несколько индексов для оптимизации запросов, триггеры и функции. К примеру, есть триггер, который вызывается перед Insert в таблицу emptyLocation, который проверяет если в локации есть что-то.

drop table if exists MonsterType,

Armor,

Sword,

Potion,

Oil,

Monster,

Shop,

QuestGiver,

Location,

Witcher,

Board;

CREATE TABLE MonsterType

(

typeId SERIAL PRIMARY KEY,

name TEXT NOT NULL

);

CREATE TABLE Armor

(

armorName TEXT PRIMARY KEY,

price INT NOT NULL CHECK (price > 0),

prot INT NOT NULL CHECK (prot > 0)

);

CREATE TABLE Sword

(

swordName TEXT PRIMARY KEY,

price INT NOT NULL CHECK (price > 0),

atack INT NOT NULL CHECK (atack > 0)

);

CREATE TABLE Potion

(

potionName TEXT PRIMARY KEY,

price INT NOT NULL CHECK (price > 0),

prot INT NOT NULL CHECK (prot > 0),

typeId SMALLINT REFERENCES MonsterType on delete cascade on update cascade

);

CREATE TABLE Oil

(

oilName TEXT PRIMARY KEY,

price INT NOT NULL CHECK (price > 0),

atack INT NOT NULL CHECK (atack > 0),

typeId SMALLINT REFERENCES MonsterType on delete cascade on update cascade

);

CREATE TABLE Monster

(

monsterName TEXT PRIMARY KEY,

type\_id SMALLINT REFERENCES MonsterType on delete cascade on update cascade,

color TEXT,

height INT CHECK (height > 0 and height < 500),

armsNum SMALLINT CHECK (armsNum > 0 and armsNum < 20),

atack INT NOT NULL CHECK (atack > 0),

health INT NOT NULL CHECK (health > 0)

);

CREATE TABLE Shop

(

shopId SERIAL PRIMARY KEY,

name TEXT,

type TEXT,

markup INT

);

CREATE TABLE QuestGiver

(

questGiver TEXT PRIMARY KEY,

monsterName TEXT REFERENCES Monster on delete cascade on update cascade,

startPrice INT NOT NULL CHECK (startPrice > 0),

money INT NOT NULL CHECK (money > 0),

charisma INT NOT NULL

);

CREATE TABLE Location

(

locationName TEXT PRIMARY KEY,

country TEXT NOT NULL,

questGiver TEXT REFERENCES QuestGiver on delete cascade on update cascade,

shopId INT REFERENCES Shop

);

CREATE TABLE Witcher

(

witcherName TEXT PRIMARY KEY,

armorName TEXT NOT NULL REFERENCES Armor on delete cascade on update cascade,

swordName TEXT NOT NULL REFERENCES Sword on delete cascade on update cascade,

potionName TEXT REFERENCES Potion on delete cascade on update cascade,

oilName TEXT REFERENCES Oil on delete cascade on update cascade,

health INT NOT NULL CHECK (health >= 0),

money INT NOT NULL,

charisma INT NOT NULL,

locationName TEXT REFERENCES Location

);

CREATE TABLE Board

(

boardId SERIAL PRIMARY KEY,

witcherName TEXT NOT NULL REFERENCES Witcher on delete cascade on update cascade,

monsterName TEXT NOT NULL REFERENCES Monster on delete cascade on update cascade,

price INT NOT NULL CHECK (price > 0),

questGiver TEXT NOT NULL REFERENCES QuestGiver on delete cascade on update cascade

);

insert into MonsterType values (1, 'Vampire');

insert into MonsterType values (2, 'Goul');

insert into MonsterType values (3, 'Ghost');

insert into Armor values ('Chest1', 10, 20);

insert into Armor values ('Chest2', 20, 30);

insert into Armor values ('Chest3', 30, 40);

insert into Sword values ('Sword1', 10, 20);

insert into Sword values ('Sword2', 20, 30);

insert into Sword values ('Sword3', 30, 40);

insert into Potion values ('Potion1', 10, 20, 1);

insert into Potion values ('Potion2', 20, 30, 2);

insert into Potion values ('Potion3', 30, 40, 3);

insert into Oil values ('Oil1', 10, 20, 1);

insert into Oil values ('Oil2', 20, 30, 2);

insert into Oil values ('Oil3', 30, 40, 3);

insert into Monster values ('Monster1', 1, 'Red', 20, 4, 10, 10);

insert into Monster values ('Monster2', 2, 'White', 20, 4, 20, 10);

insert into Monster values ('Monster3', 3, 'Red', 20, 4, 10, 30);

insert into Shop values (1, 'Sword1', 'Sword', 5);

insert into Shop values (2, 'Chest2', 'Armor', 10);

insert into Shop values (3, 'Oil3', 'Oil', 10);

insert into QuestGiver values ('Semen', 'Monster1', 10, 1000, 3);

insert into QuestGiver values ('Vika', 'Monster1', 20, 1000, 3);

insert into QuestGiver values ('L', 'Monster2', 10, 1000, 3);

insert into Location values ('Mid', 'Temeria', 'Vika', 1);

insert into Location values ('Up', 'Temeria', 'Vika', 1);

insert into Location values ('Bot', 'Nig');

insert into Witcher values ('Boris', 'Chest2', 'Sword3', 'Potion3', 'Oil2', 100, 20, 5, 'Up');

insert into Witcher values ('Viktor', 'Chest2', 'Sword3', 'Potion3', 'Oil2', 100, 20, 5, 'Bot');

insert into Witcher values ('Sasha', 'Chest1', 'Sword1', 'Potion1', 'Oil1', 100, 20, 5, 'Up');

insert into Board values (default, 'Viktor', 'Monster2', 22, 'Vika');

insert into Board values (default, 'Sasha', 'Monster2', 22, 'Vika');

insert into Board values (default, 'Sasha', 'Monster1', 22, 'L');

**Функции и триггеры**

create or replace function emptyLocation() returns trigger as

$$

begin

if new.shopid is NULL and new.questgiver is NULL

then

new.locationname = concat(text('useless '),new.locationname);

end if;

return new;

end;

$$

language plpgsql;

create trigger checkLocation

before insert or update

on location FOR EACH ROW

execute procedure emptyLocation();

create or replace function makeDeal(witcherNameIn text, questgiverName text) returns int as

$$

DECLARE

witcherCharisma int;

questgiverCharisma int;

price int;

maxprice int;

finalprice int;

monster text;

begin

select charisma into witcherCharisma from witcher where witcher.witchername = witcherNameIn;

select charisma into questgiverCharisma from questgiver where questgiver.questgiver = questgiverName;

select startprice into price from questgiver where questgiver.questgiver = questgiverName;

select money into maxprice from questgiver where questgiver.questgiver = questgiverName;

select monstername into monster from questgiver where questgiver.questgiver = questgiverName;

if witcherCharisma < questgiverCharisma then finalprice = price;

end if;

if witcherCharisma = questgiverCharisma then

finalprice = price;

update witcher set charisma = charisma + 1 where witchername = witcherNameIn;

end if;

if witcherCharisma > questgiverCharisma then

finalprice = price \* (witcherCharisma/questgiverCharisma);

if finalprice > maxprice then finalprice = maxprice; end if;

update witcher set charisma = charisma + 1 where witchername = witcherNameIn;

end if;

insert into board values (default, witcherNameIn, monster, finalprice, questgiverName);

return finalprice;

end;

$$

language plpgsql;

create or replace function battle(witcherNameIn text, monsterNameIn text) returns text as

$$

DECLARE

witcherHealth int;

witcherAttack int;

witcherAttackTemp int;

witcherDefence int;

witcherDefenceTemp int;

attackType int;

defenceType int;

monsterType int;

monsterHealth int;

monsterAttack int;

begin

select health into witcherHealth from witcher where witcher.witchername = witcherNameIn;

select atack into witcherAttack from witcher join sword s on witcher.swordname = s.swordname where witcher.witchername = witcherNameIn;

select prot into witcherDefence from witcher join armor a on a.armorname = witcher.armorname where witcher.witchername = witcherNameIn;

select atack into witcherAttackTemp from witcher join oil o on witcher.oilname = o.oilname where witcher.witchername = witcherNameIn;

select prot into witcherDefenceTemp from witcher join potion p on witcher.potionname = p.potionname where witcher.witchername = witcherNameIn;

select typeid into attackType from witcher join oil o on witcher.oilname = o.oilname where witcher.witchername = witcherNameIn;

select typeid into defenceType from witcher join potion p on witcher.potionname = p.potionname where witcher.witchername = witcherNameIn;

select health into monsterHealth from monster where monster.monstername = monsterNameIn;

select atack into monsterAttack from monster where monster.monstername = monsterNameIn;

select type\_id into monsterType from monster where monster.monstername = monsterNameIn;

if attackType = monsterType then witcherAttack = witcherAttack + witcherAttackTemp; end if;

if defenceType = monsterType then witcherDefence = witcherDefence + witcherDefenceTemp; end if;

if monsterAttack - witcherDefence <= 0 then monsterAttack = 0;

else monsterAttack = monsterAttack - witcherDefence; end if;

while (witcherHealth > 0 and monsterHealth > 0) loop

witcherHealth = witcherHealth - monsterAttack;

monsterHealth = monsterHealth - witcherAttack;

end loop;

update witcher set health=witcherHealth where witcher.witchername = witcherNameIn;

if EXISTS(select \* from board where witchername=witcherNameIn and monstername=monsterNameIn) then

update witcher set

money=money+(select sum(price) from board where witchername=witcherNameIn and monstername=monsterNameIn)

where witcher.witchername = witcherNameIn;

end if;

update monster set health=monsterHealth where monster.monstername = monsterNameIn;

if witcherHealth > 0 then return 'Победил ведьмак';

else return 'Ведьмак умер'; end if;

end;

$$

language plpgsql;

SELECT battle('Sasha','Monster2');

--проверить здоровье (тригер)

create or replace function zeroHealthWitcher() returns trigger as

$$

begin

if new.health <= 0

then

delete from board where witchername = new.witchername;

delete from witcher where witchername = new.witchername;

return null;

end if;

return new;

end;

$$

language plpgsql;

create trigger zeroHealthWitcher

before insert or update

on witcher FOR EACH ROW

execute procedure zeroHealthWitcher();

create or replace function zeroHealthMonster() returns trigger as

$$

begin

if new.health <= 0

then

delete from board where board.monstername = new.monstername;

return null;

end if;

return new;

end;

$$

language plpgsql;

create trigger zeroHealthMonster

before insert or update

on monster FOR EACH ROW

execute procedure zeroHealthMonster();

**4 Этап**

На данном этапе необходимо было реализовать информационную систему на основе разработанной базы данных. Технологией для Backend я выбрал Spring Boot, а для front - React.

Пример, одной из бизнес функций, **сражение**.

На сервере

@GetMapping("/battle")

public ResponseEntity<String> battle(@RequestParam("witcherName") String witcherName, @RequestParam("monsterName") String otherName){

if (witcherName.isEmpty()) return new ResponseEntity<>(null, HttpStatus.BAD\_REQUEST);

if (otherName.isEmpty()) return new ResponseEntity<>(null, HttpStatus.BAD\_REQUEST);

return new ResponseEntity<>(witcherService.battle(witcherName, otherName), HttpStatus.OK);

}

@Override

public String battle(String witcherName, String monsterName){

SimpleJdbcCall battleFunc = new SimpleJdbcCall(jdbcTemplate).withFunctionName("battle");

SqlParameterSource in = new MapSqlParameterSource()

.addValue("witcherNameIn", witcherName)

.addValue("monsterNameIn", monsterName);

return battleFunc.executeFunction(String.class, in);

}

На фронте

handleOnSubmit(e){

if (this.state.witcherName !== "" && this.state.monsterName !== ""){

fetch("http://localhost:8080/battle?witcherName=" + this.state.witcherName

+ "&monsterName=" + this.state.monsterName)

.then(res => res.text())

.then(response => {

this.setState({deal : response})

});

}

e.preventDefault();

}

render() {

return(

<div>

<form onSubmit={this.handleOnSubmit}>

<select onClick={this.changeWitcher} onChange={this.changeWitcher}>

<option value="" disabled selected>Ведьмак</option>

{this.state.witcher.map((elem, index) => {

return(<option value={elem.witcherName}>{elem.witcherName}</option> );

})}

</select>

<nobr> будет биться с </nobr>

<select onClick={this.changeMonster} onChange={this.changeMonster}>

<option value="" disabled selected>Монстр</option>

{this.state.monster.map((elem, index) => {

return(<option value={elem.monsterName}>{elem.monsterName}</option> );

})}

</select>

<input className="button" type="submit" value="БОЙ"/>

</form>

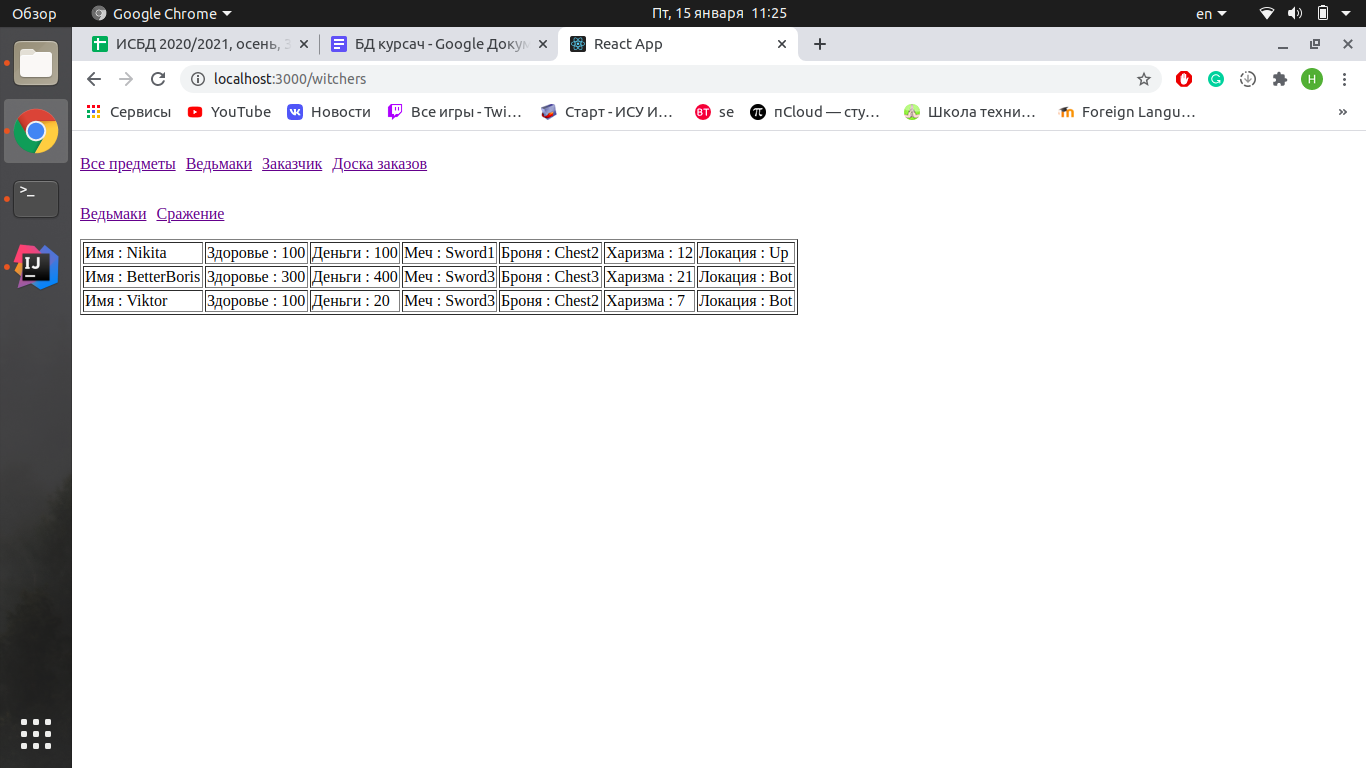
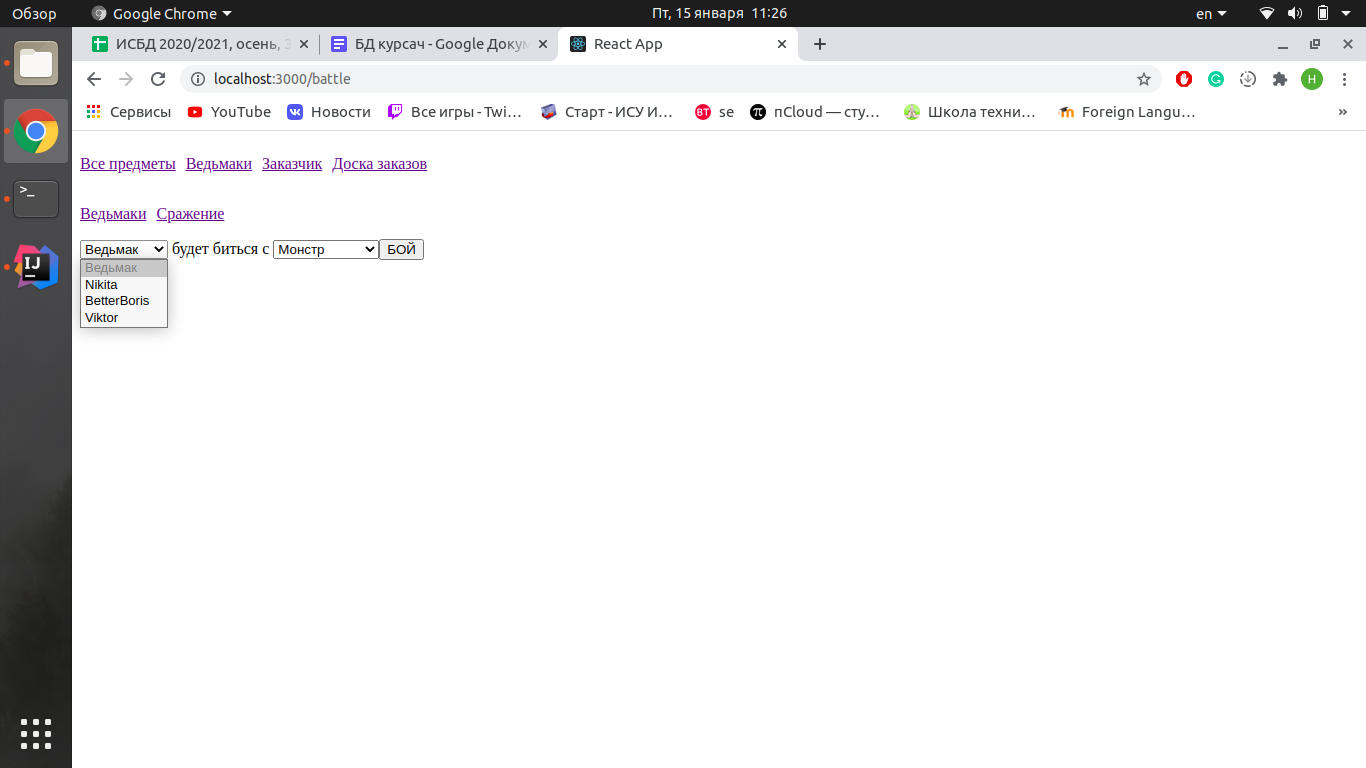
{this.state.deal !== "" && <p> и {this.state.deal}</p>}

</div>

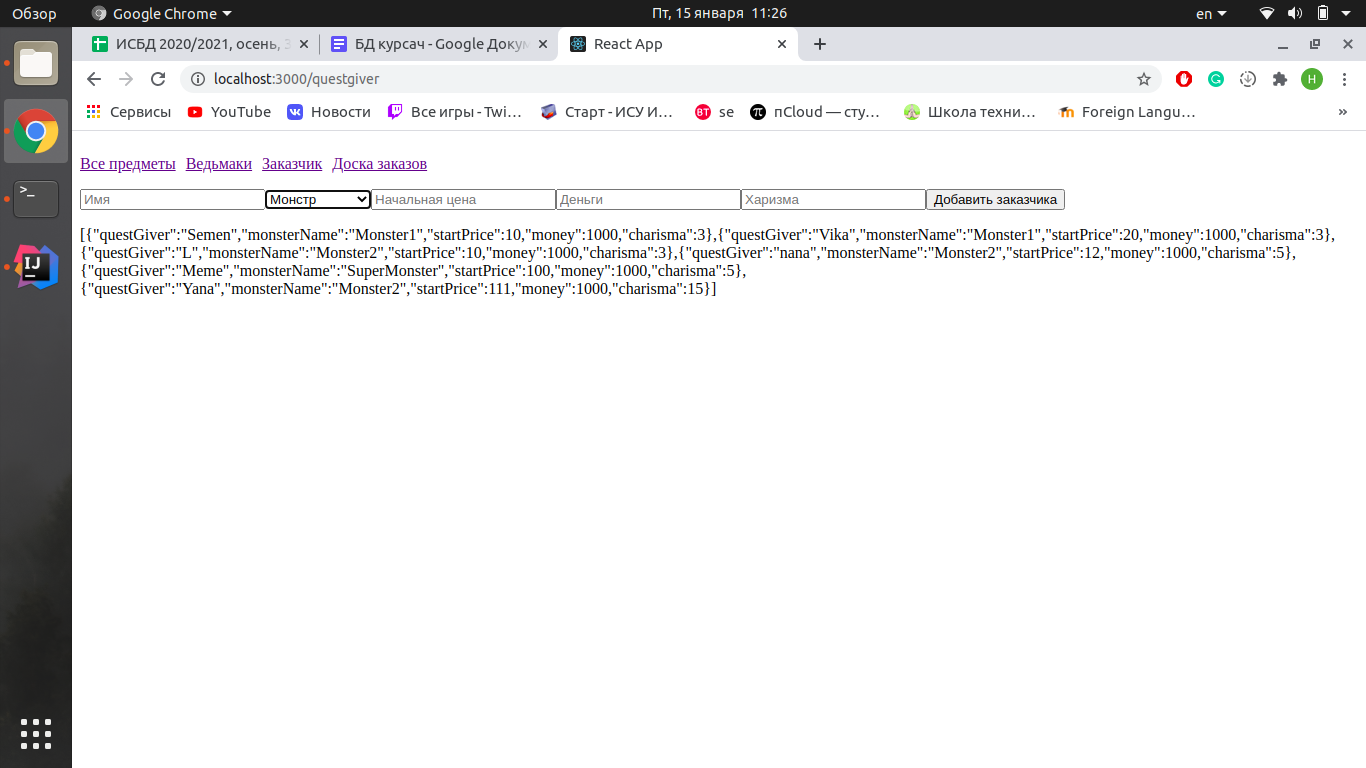
);

}

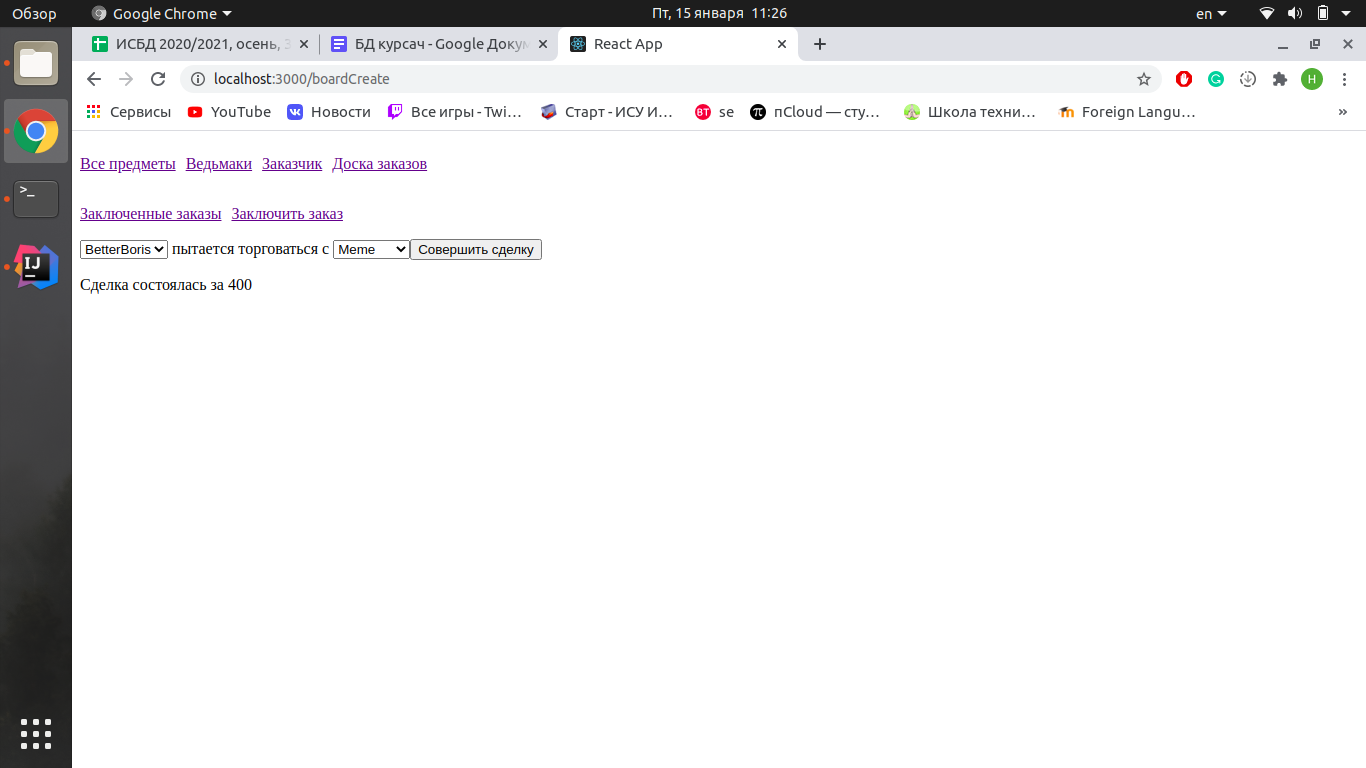
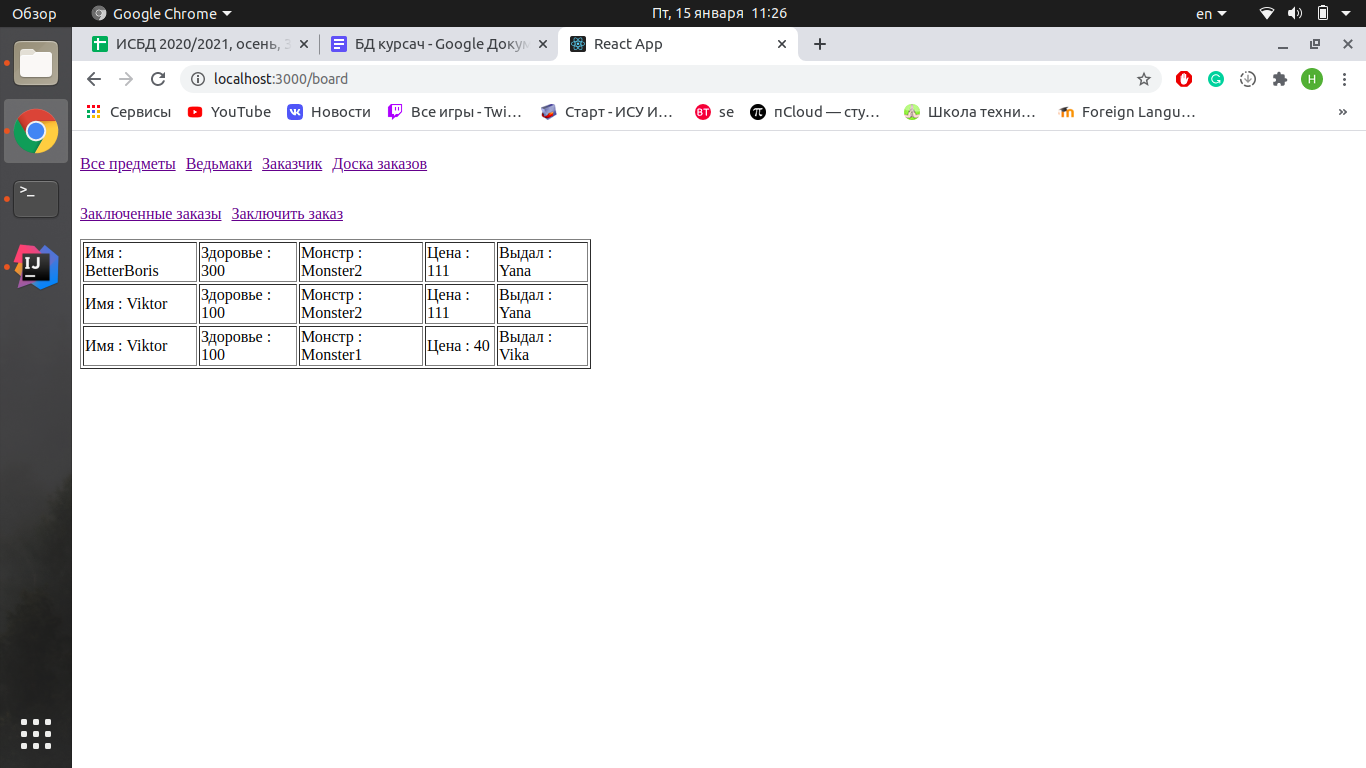
}

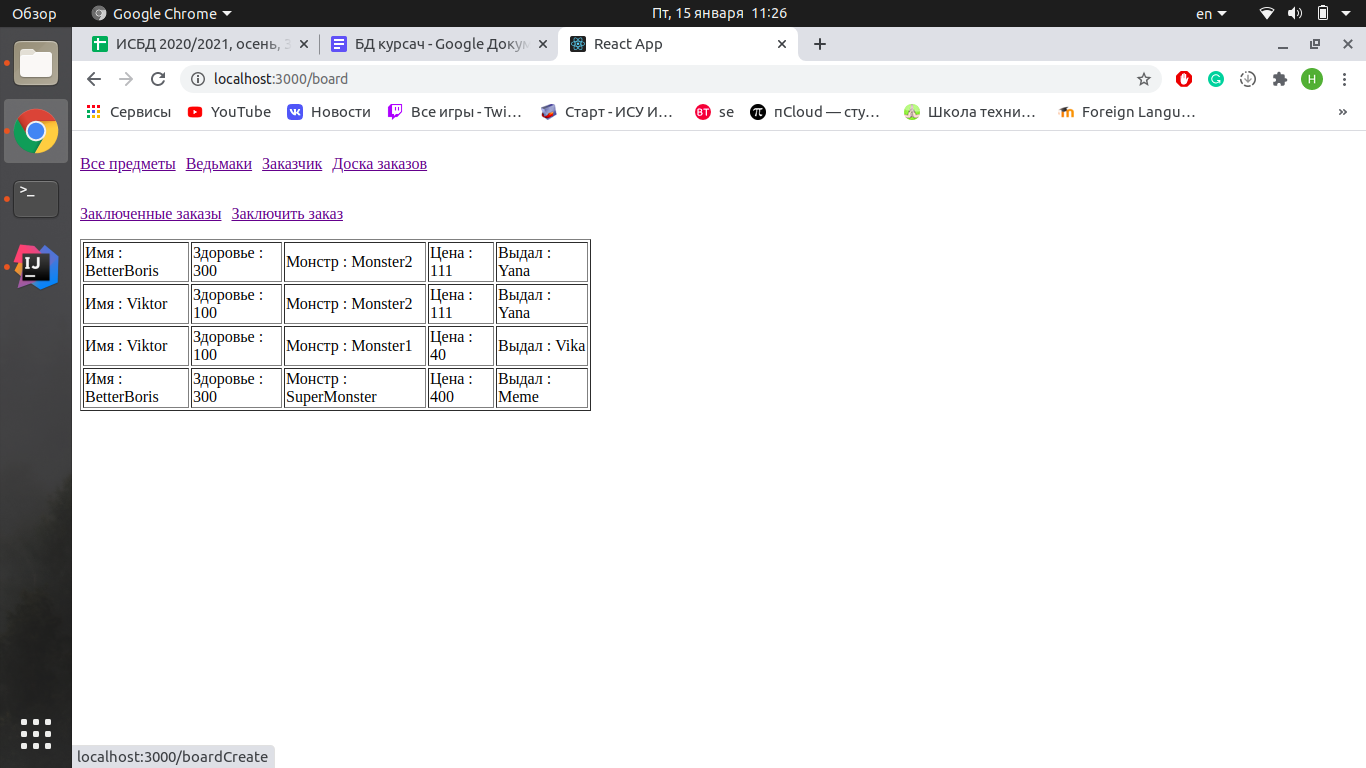


Есть возможность добавить заказчика



Есть возможность ведьмаку заключить сделку с заказчиком и принять заказ





**Вывод**: В процессе написания курсового проекта я вспомнил базовые вещи, касающиеся БД и SQL, создал собственные триггеры и функции, воспользовался функциями взаимодействия JDBC и PostgreSQL. База данных это невероятно удобная структура значительно упрощающая работу с данными и их контроль. Надеюсь, что полученные навыки пригодятся мне в жизни.