

Национальный исследовательский Университет ИТМО
Мегафакультет компьютерных технологий и управления
Факультет программной инженерии и компьютерной техники

Информационные системы и базы данных

Курсовой проект

Работу

выполнили:

Н. В. Кулаков,

Н. К. Нестеров

Группа: Р33312

Преподаватель:

Д. М. Шешуков

Санкт-Петербург
2023

Содержание

1. Этап 1	3
1.1. Описание предметной области	3
1.2. Описание бизнес процессов	3
1.3. Список сущностей и их классификации	3
2. Этап 2	4
2.1. Инфологическая модель	4
2.2. Даталогическая модель	5
3. Этап 3	5
3.1. Запросы	5
3.1.1. Для администраторов системы	5
3.1.2. Для поставщика	5
3.1.3. Для владельца кафе	5
3.1.4. Для пользователя	6
3.2. Создание объектов	6
3.2.1. Таблицы	6
3.2.2. Функции	8
3.2.3. Процедуры	8
3.3. Удаление объектов	10
3.3.1. Таблицы	10
3.3.2. Процедуры	10
3.3.3. Функции	10
3.4. Начальные значения	10
3.4.1. Создание	10
3.4.2. Удаление	11
3.5. Примеры запросов выше	11
3.5.1. Администратор	11
3.5.2. Поставщик	12
3.5.3. Владельцы	12
3.5.4. Пользователи	13
3.6. Индексы	14
3.6.1. Описание	14
3.6.2. Реализация	14
4. Этап 4	15
4.1. Скриншоты	15
4.2. Ссылки	19

1. Этап 1

1.1. Описание предметной области

В мире существует множество кофеен, которые объединяются в целые сети. Владелец таких заведений нужно иметь возможность мониторить их состояние и управлять ими, а покупателям (которые знают SQL, конечно же) всегда удобнее, когда у кофейни доступно актуальное меню в виде БД!

Без такой системы владельцы тратят миллионы на товары, которые в конце концов будут выброшены, так как успели просрочиться. А покупателям приходится неделями стоять в очереди за любимым блюдом или напитком, в ожидании того, что кофейня закупит нужные ингредиенты.

1.2. Описание бизнес процессов

Владелец может для каждой конкретной кофейни записывать доступное количество ингредиентов, изменять его при использовании ингредиентов, смотреть цены на разные товары у поставщиков, пополнять запасы.

Владелец может создавать меню с различными блюдами, привязанными к конкретному заведению и менять их в зависимости от необходимости. Эти меню будут видны и покупателям.

Продавцы продуктов могут выставлять свои товары, которые будут доступны для покупки в любом заведении, подключенном к этой системе.

1.3. Список сущностей и их классификации

Стержневые сущности:

- Поставщик товаров (название, описание)
- Кафе (адрес, комментарий)
- Меню (название, комментарий)
- Рецепт (название, описание, информация о питательных свойствах)
- Ингредиент (название, комментарий, единицы измерения)

Ассоциативные сущности:

- Цена ингредиента (количество, цена за единицу) - Поставщик М-М Ингредиент
- Ингредиент рецепта (количество, обязательность) - Рецепт М-М Ингредиент
- Запас продукта (количество, срок годности, дата покупки) - Кафе М-М Ингредиент
- Меню кафе - Кафе М-М Меню
- Пункт меню (цена) - Меню М-М Рецепт
- Заказ (цена, кафе) - Цена М-М Кафе

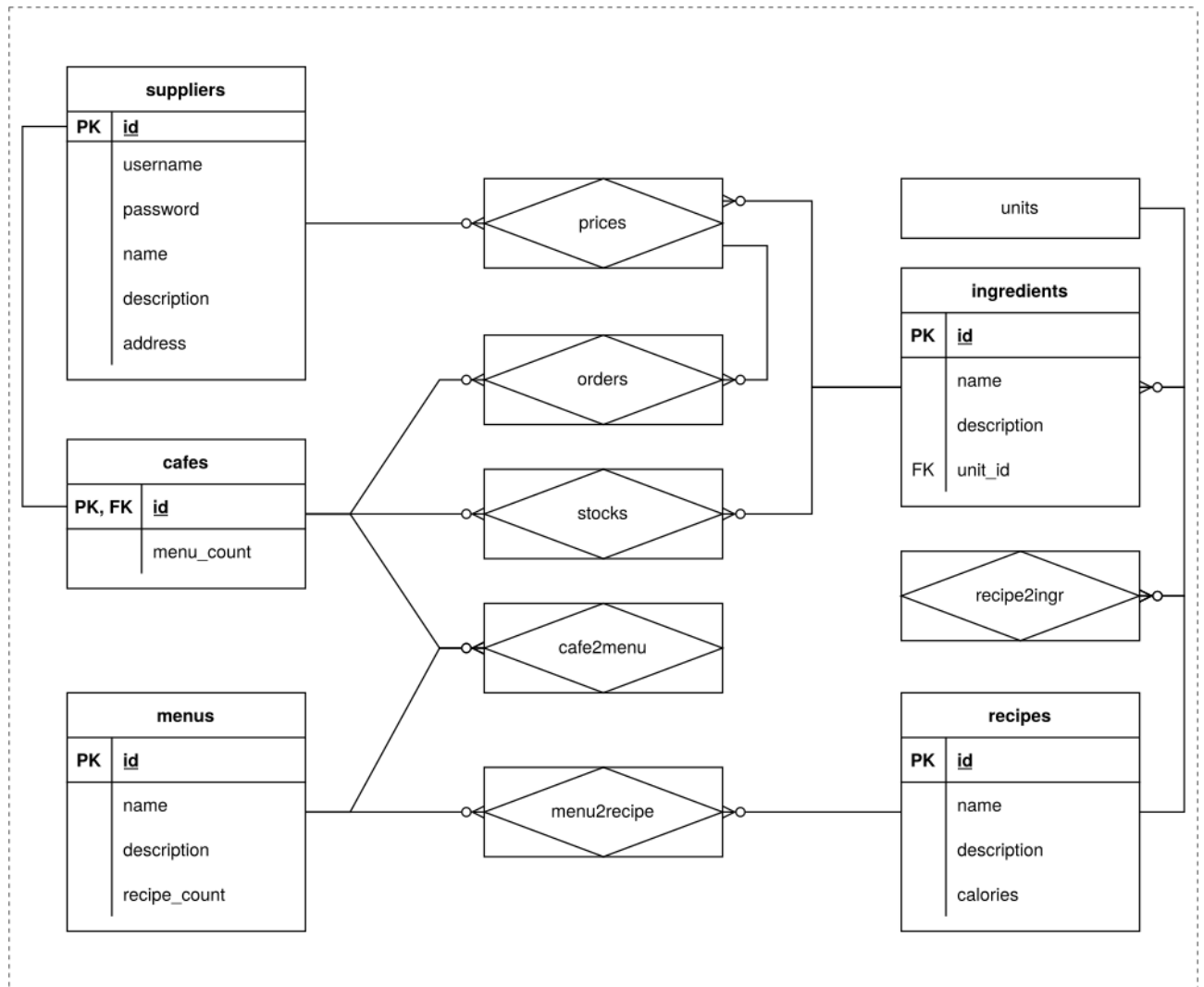
Характеристические сущности:

- Единицы измерения (название) - к ингредиентам
- Статус заказа

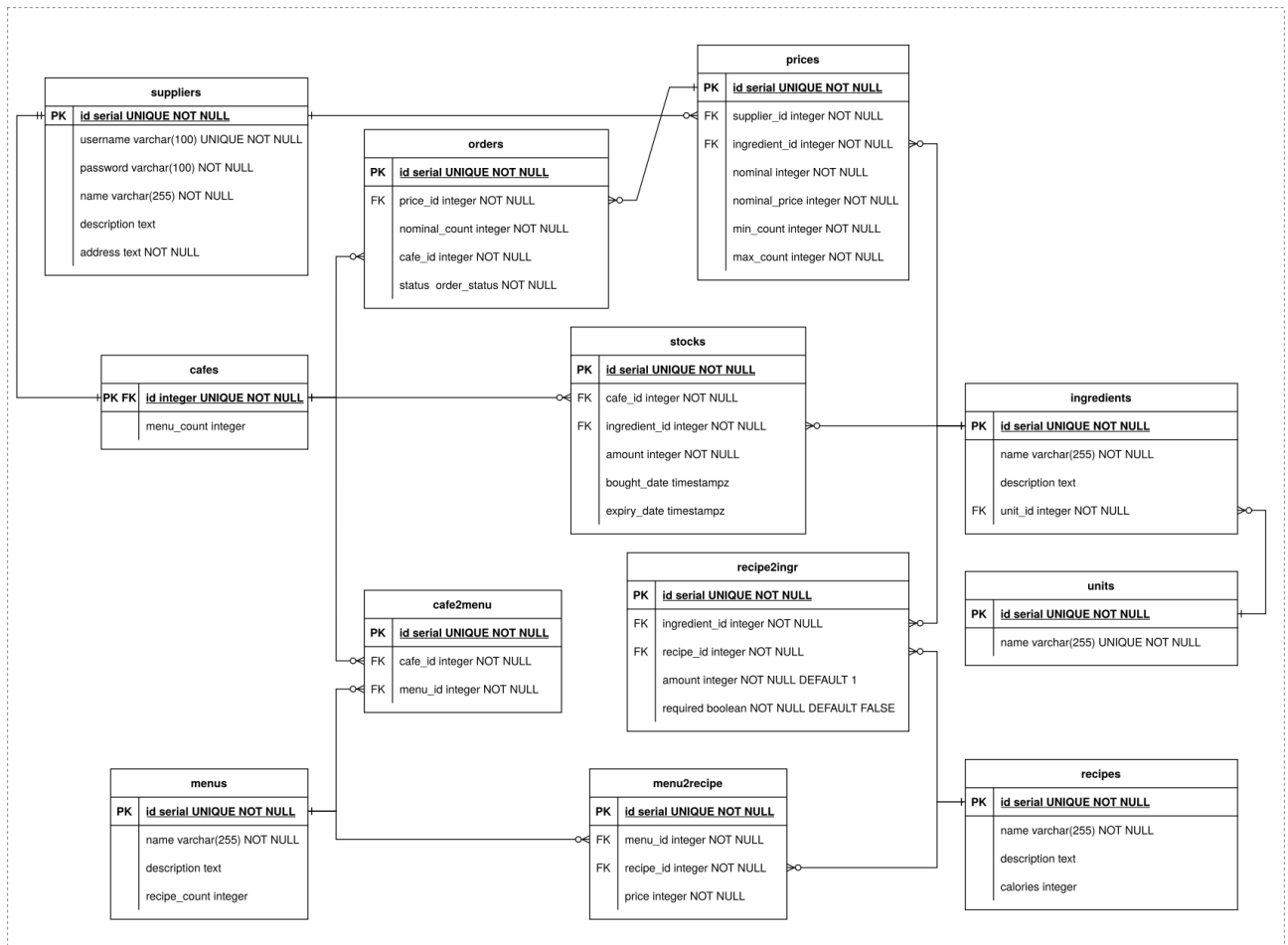
Характеристическая сущность (характеристика) — связь вида "многие-к-одной" или "одна-к-одной" между двумя сущностями (частный случай ассоциации). Цель характеристики - описание или уточнение некоторой другой сущности.

2. Этап 2

2.1. Инфологическая модель



2.2. Даталогическая модель



3. Этап 3

3.1. Запросы

3.1.1. Для администраторов системы

- Редактирование доступных единиц измерения
- Редактирование доступных ингредиентов

3.1.2. Для поставщика

- Создание и удаление аккаунта поставщика
- Размещение цен на товары

3.1.3. Для владельца кафе

- Добавление, изменение, удаление кафе
- Добавление, удаление, изменение меню
- Добавление, удаление меню в кафе
- Добавление, изменение, удаление рецептов
- Добавление, удаление рецептов в меню
- Поиск товаров у поставщиков
- Покупка товара у поставщиков

- Получение информации о купленных товаров, т.е. кассового чека, и обновление склада при изменении статуса чека на получено
- Редактирование и удаление записей об инвенторе

3.1.4. Для пользователя

- Просмотр меню кафе
- Просмотр рецептов блюд

3.2. Создание объектов

Идет в правильном порядке запуска скриптов.

3.2.1. Таблицы

```
-- Characteristic
-- NOTE: mb replace with enum
create table units(
  id serial primary key,
  name varchar(255) not null unique
);

create type order_status as enum();

-- Core
create table suppliers(
  id serial primary key,
  username varchar(100) unique not null,
  password varchar(100) not null,
  name varchar(255) not null,
  description text,
  address text not null
);

create table cafes(
  id integer not null references suppliers
  on update cascade on delete restrict,
  menu_count integer not null
);

create table menus(
  id serial primary key,
  name varchar(255) not null,
  description text,
  recipe_count integer not null
);

create table recipes(
  id serial primary key,
  name varchar(255) not null,
  description text,
  calories integer
  check(calories >= 0)
);

create table ingredients(
  id serial primary key,
  name varchar(255) not null,
```

```

description text,
unit_id integer not null references units
    on update cascade on delete cascade,
unique(name, unit_id)
);

-- Associative
create table prices(
    id serial primary key,
    supplier_id integer not null references suppliers
        on update cascade on delete cascade,
    ingredient_id integer not null references ingredients
        on update cascade on delete cascade,
    nominal integer default 1 not null
        check(nominal > 0),
    nominal_price integer not null
        check(nominal_price >= 0),
    min_count integer not null
        check(min_count > 0),
    max_count integer,
    constraint price_nominal_count check(min_count <= max_count or max_count is null)
);

create table orders(
    id serial primary key,
    price_id integer not null,
    nominal_count integer not null,
    cafe_id integer not null,
    status order_status not null
);

create table recipe2ingr(
    id serial primary key,
    recipe_id integer not null references recipes
        on update cascade on delete cascade,
    ingredient_id integer not null references ingredients
        on update cascade on delete cascade,
    amount integer default 1 not null
        check(amount >= 0),
    required boolean default false not null
);

create table stocks(
    id serial primary key,
    cafe_id integer not null references cafes
        on update cascade on delete cascade,
    ingredient_id integer not null references ingredients
        on update cascade on delete cascade,
    amount integer not null
        check(amount >= 0),
    bought_date timestamp with time zone,
    expiry_date timestamp with time zone
);

create table cafe2menu(
    id serial primary key,
    cafe_id integer not null references cafes,
    -- on update cascade on delete cascade,
    menu_id integer not null references menus
    -- on update cascade on delete cascade

```

```
);

create table menu2recipe(
  id serial primary key,
  menu_id integer not null references menus
    on update cascade on delete cascade,
  recipe_id integer not null references recipes
    on update cascade on delete cascade,
  price integer not null
    check(price >= 0)
);
```

3.2.2. Функции

```
create or replace function get_unit_id_from_name(unit_name varchar(255))
returns integer
as $$
begin
  return (select id from units where units.name = unit_name limit 1);
end
$$
language plpgsql;

create or replace function get_amount_from_nominal(nominal integer, nominal_count integer)
returns integer
as $$
begin
  return nominal * nominal_count;
end
$$
language plpgsql;
```

3.2.3. Процедуры

```
-- NOTE: if cafe_menu relations contain menu_id that are not present in table menu
create or replace function tf_cafe_menu_count()
returns trigger
language plpgsql
as $$
begin
  select count(id) into NEW.menu_count from cafe2menu where cafe2menu.menu_id = NEW.id;
  return NEW;
end
$$;

create or replace trigger tr_cafe_menu_count
before insert on cafes
for each row execute procedure tf_cafe_menu_count();

create or replace function tf_cafe_update_count()
returns trigger
language plpgsql
as $$
begin
  if (TG_OP = 'INSERT') then
    update cafes set menu_count = menu_count + 1 where id = NEW.cafe_id;
  elseif (TG_OP = 'DELETE') then
    update cafes set menu_count = menu_count - 1 where id = OLD.cafe_id;
  return OLD;
end
$;
```



```

    end if;
    return NEW;
end
$$;

create or replace trigger tr_cafe_update_count
before insert or delete on cafe2menu
for each row execute procedure tf_cafe_update_count();

-- NOTE: if menu_item relations contain item_id that are not present in table menu
create or replace function tf_menu_recipe_count()
returns trigger
language plpgsql
as $$
begin
    select count(id) into NEW.recipe_count from menu2recipe where menu2recipe.menu_id = NEW
        .id;
    return NEW;
end
$$;

create or replace trigger tr_menu_recipe_count
before insert on menus
for each row execute procedure tf_menu_recipe_count();

create or replace function tf_menu_update_count()
returns trigger
language plpgsql
as $$
begin
    if (TG_OP = 'INSERT') then
        update menus set recipe_count = recipe_count + 1 where id = NEW.menu_id;
    elseif (TG_OP = 'DELETE') then
        update menus set recipe_count = recipe_count - 1 where id = OLD.menu_id;
        return OLD;
    end if;
    return NEW;
end
$$;

create or replace trigger tr_menu_update_count
before insert or delete on menu2recipe
for each row execute procedure tf_menu_update_count();

-- Order trigger on table orders update
create or replace function tf_order_status_updated()
returns trigger
as $$
declare
    v_price record;
begin
    select * into v_price from prices where id = OLD.price_id limit 1;
    if (OLD.status = 'paid' and NEW.status = 'received') then
        insert into stocks(cafes_id, ingredient_id, amount, bought_date)
        values(
            NEW.cafes_id,
            v_price.ingredient_id,
            get_amount_from_nominal(v_price.nominal, NEW.nominal_count),

```

```

        clock_timestamp()
    );
    elsif (OLD.status = 'received' and NEW.status <> 'received') then
        raise notice 'Cannot update status of %, it is final', v_price.id;
        return OLD;
    end if;
    return NEW;
end
$$
language plpgsql;

create or replace trigger tr_insert_order_in_stock
after update of status on orders
for each row
execute procedure tf_order_status_updated();

```

3.3. Удаление объектов

3.3.1. Таблицы

```

drop table if exists
units,
suppliers,
cafes,
menus,
recipes,
ingredients,
prices,
orders,
recipe2ingr,
stocks,
cafe2menu,
menu2recipe
cascade;

drop type order_status;

```

3.3.2. Процедуры

```

-- Trigger functions
drop function if exists
tf_cafe_menu_count,
tf_cafe_update_count,
tf_menu_recipe_count,
tf_menu_update_count,
tf_order_status_updated
cascade;

```

3.3.3. Функции

```

drop function if exists
get_unit_id_from_name,
get_amount_from_nominal
restrict;

```

3.4. Начальные значения

3.4.1. Создание

```

-- Add order status
alter type order_status add value 'paid';

```

```

alter type order_status add value 'received';

-- Add initial units
insert into units(name) values('milliliter');
insert into units(name) values('unit'); -- колво- штук

insert into units(name) values('gram');

-- ounce
insert into units(name) values('ounce');

create or replace function ounce_to_gram(cnt double precision)
returns double precision
as $$
begin
    return cnt * 28.35;
end
$$
language plpgsql;

create or replace function gram_to_ounce(cnt double precision)
returns double precision
as $$
begin
    return cnt / 28.35;
end
$$
language plpgsql;

```

3.4.2. Удаление

```

drop function if exists
    ounce_to_gram,
    gram_to_ounce
cascade;

```

3.5. Примеры запросов выше

3.5.1. Администратор

```

-- Units
-- pounds
insert into units(name) values('pounds');

create or replace function pound_to_ounce(cnt double precision)
returns double precision
as $$
begin
    return cnt * 16;
end
$$
language plpgsql;

create or replace function ounce_to_pound(cnt double precision)
returns double precision
as $$
begin
    return cnt / 16;
end
$$
language plpgsql;

```

```

update units set name = 'pound' where name = 'pounds';

delete from units where name = 'pounds';
drop function if exists
    pound_to_ounce,
    ounce_to_pound
restrict;

-- Ingredients
insert into ingredients(name, description, unit_id)
values('cucumber', 'small cucumber', get_unit_id_from_name('gram'));

update ingredients set description = null where name = 'cucumber';
delete from ingredients
where name = 'cucumber' and unit_id = get_unit_id_from_name('gram');

```

3.5.2. Поставщик

```

-- Supplier account
insert into suppliers(name, description, address)
values('groceries', null, 'kronva');

update suppliers set description = 'groc' where name = 'groceries';
delete from suppliers where name = 'groceries';

-- Prices
insert into prices(supplier_id, ingredient_id, nominal, nominal_price, min_count,
    max_count)
values(
    (select id from suppliers where name = 'groceries' limit 1),
    (select id from ingredients where name = 'cucumber' and unit_id = get_unit_id_from_name(
        'gram')),
    1000,
    100,
    1,
    null
);
update prices set nominal_price = 120
where
    ingredient_id = (select id from ingredients where name = 'cucumber' and
        unit_id = get_unit_id_from_name('gram'));

delete from prices
where
    ingredient_id = (select id from ingredients where name = 'cucumber' and
        unit_id = get_unit_id_from_name('gram')) and
    supplier_id = (select id from suppliers where name = 'groceries' limit 1);

```

3.5.3. Владельцы

```

-- Cafe
insert into cafes(name, description, address)
values('first', null, 'belorusskaya');

update cafes set address = 'belorusskaya 6' where name = 'first';
delete from cafes where name = 'first';

-- Menu
insert into menus(name) values('first');

```

```

update menus set description = 'vegan' where name = 'first';
delete from menus where description = 'vegan';

-- Menu in cafe
insert into cafe2menu(cafes_id, menu_id)
values(
  (select id from cafes where name = 'first'),
  (select id from menus where name = 'first')
);
delete from cafe2menu
where
  cafes_id = (select id from cafes where name = 'first') and
  menu_id = (select id from menus where name = 'first');

-- Recipes
insert into recipes(name) values('makaroni');
-- update
delete from recipes where name = 'makaroni';

-- Search suppliers requests
select prices.id, nominal, nominal_price, min_count, max_count, name, description,
  unit_id from prices
join ingredients on ingredients.id = prices.ingredient_id
where
  supplier_id = (select id from suppliers where name = 'groceries');

-- Order from supplier
insert into orders(price_id, nominal_count, cafes_id, status)
values(
  (select id from prices limit 1),
  10,
  (select id from cafes limit 1),
  'paid'
);
update orders set status = 'received';

-- Stock
update stocks set expiry_date = clock_timestamp();
delete from stocks;

```

3.5.4. Пользователи

```

-- Show recipes in menu in cafe
select recipes.name, recipes.calories, menu2recipe.price
from
  cafe2menu
  join menus on menus.id = cafe2menu.menu_id
  join menu2recipe on menu2recipe.menu_id = menus.id
  join recipes on recipes.id = menu2recipe.recipe_id
where
  cafes_id = 1;

-- Show ingredients of recipes
select ingredients.name, ingredients.description, recipe2ingr.amount, recipe2ingr.
  required
from
  recipes
  join recipe2ingr on recipe2ingr.recipe_id = recipes.id
  join ingredients on ingredients.id = recipe2ingr.ingredient_id
where
  recipes.name = 'makaroni';

```

3.6. Индексы

3.6.1. Описание

По умолчанию для primary key PostgreSQL использует btree. На основании полученная даталогической модели и написанным выше запросам были выбраны следующие индексы, исключая индексы по умолчанию:

- prices(nominal_price) - btree. Будет производиться сортировка чисел.
- stocks(expiry_date) - btree. Осуществление сортировки и фильтрации по времени окончания срока хранения.
- cafes(name) - gin. Поиск кафе по паттерну.
- menus(name) - gin. Поиск позиции в меню по паттерну.
- recipes(name) - gin. Поиск рецепта по паттерну.
- ingredients(name) - gin. Поиск ингредиента по паттерну.

Однако поскольку вылетает ошибка при попытке создания gin и мы не имеем достаточными правами доступа к db, чтобы ее исправить, то обойдемся hash. Описание ошибки:

```
psql:create-indexes.sql:5: ERROR: data type character varying has no default operator class for access method "gin"ПОДСКАЗКА
: You must specify an operator class for the index or define a default operator class for the data type.
```

3.6.2. Реализация

```
create index idx_prices_nominal_price on prices using btree(nominal_price);
create index idx_stocks_expiry_date on stocks using btree(expiry_date);
create index idx_supplier_name on suppliers using hash(name);
create index idx_supplier_username on suppliers using hash(username);
create index idx_menu_name on menus using hash(name);
create index idx_recipes_name on recipes using hash(name);
create index idx_ingredients_name on ingredients using hash(name);
```

4. Этап 4

4.1. Скриншоты

PUT	/users/me/data/	Edit User Data	✓	🔒
units ^				
GET	/units/	Get All Units	✓	
POST	/units/	Create Unit	✓	
ingredients ^				
GET	/ingredients/	Get All Ingredients	✓	
POST	/ingredients/	Create Ingredient	✓	
GET	/ingredients/{ingredient_id}/	Get Ingredient	✓	
PUT	/ingredients/{ingredient_id}/	Edit Ingredient	✓	
recipes ^				
GET	/recipes/	Get All Recipes	✓	
POST	/recipes/	Create Recipe	✓	
GET	/recipes/{recipe_id}/	Get Recipe	✓	
PUT	/recipes/{recipe_id}/	Edit Recipe	✓	
GET	/recipes/{recipe_id}/items/	Get Ingredients From Recipe	✓	
POST	/recipes/items/	Add Ingredient To Recipe	✓	
DELETE	/recipes/items/	Remove Ingredient From Recipe	✓	
menus ^				
GET	/menus/	Get All Menus	✓	🔒
POST	/menus/	Create Menu	✓	🔒
GET	/menus/{menu_id}/	Get Menu	✓	
PUT	/menus/{menu_id}/	Edit Menu	✓	
DELETE	/menus/{menu_id}/	Delete Menu	✓	
GET	/menus/{menu_id}/items/	Get Ingredients From Recipe	✓	

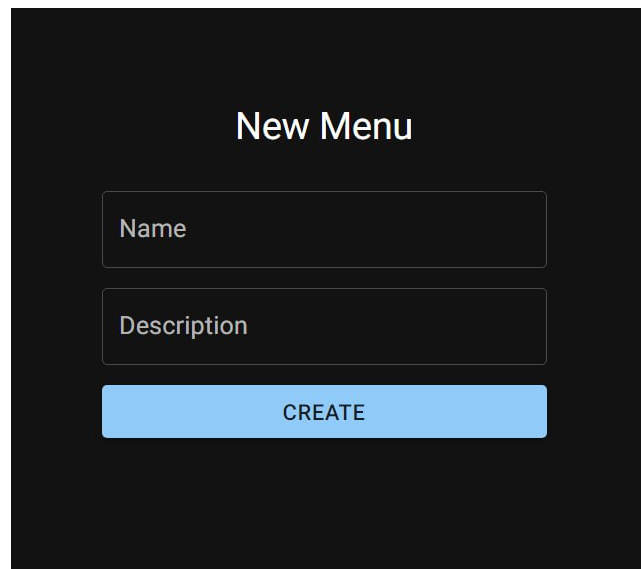
Рисунок 4.1. Вырезка из описания Backend API

The image shows a user authentication interface on a dark background. On the left, the 'New Account' form contains fields for Username (filled with 'testest'), Password (filled with '....'), Name (filled with 'tew'), Description (filled with 'rwe'), and Address. Below these fields are two radio buttons labeled 'CAFE' and 'SUPPLIER', and a 'SIGN UP' button. On the right, the 'Log In' form contains fields for Username and Password, both with red error outlines, and a 'SIGN IN' button.

Рисунок 4.2. Страница аутентификации пользователей

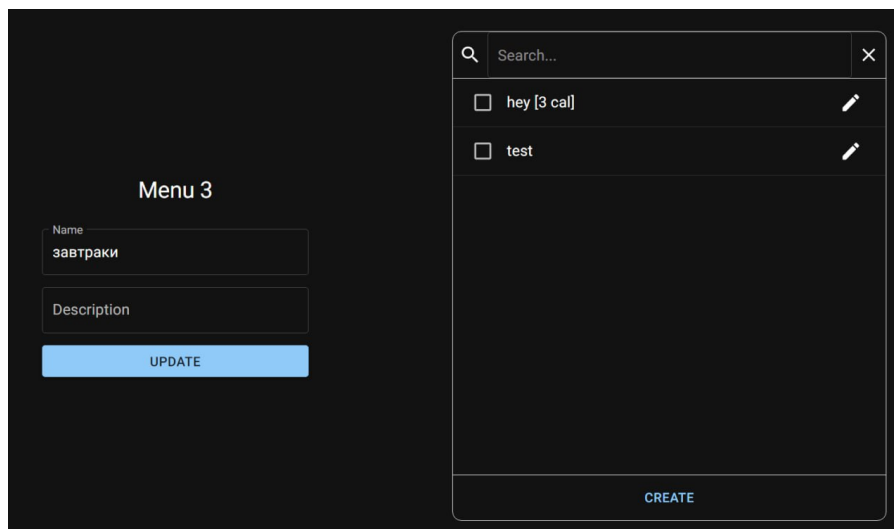
The image shows a cafe management interface. On the left, the 'Cafe' form has fields for Name (filled with 'tew'), Description (filled with 'rwe'), and Address, with an 'UPDATE' button below. On the right, a search modal is open, featuring a search bar with a magnifying glass icon and a close button (X). The modal is currently empty, and a 'CREATE' button is visible at the bottom right.

Рисунок 4.3. Страница владельца кафе



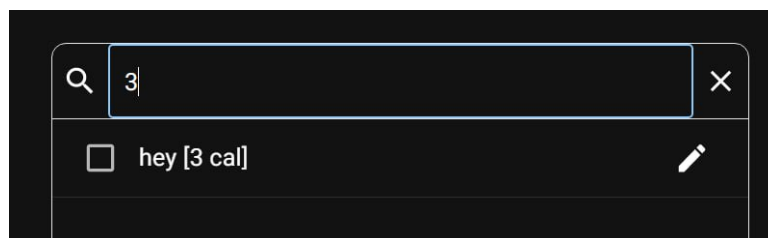
A dark-themed form titled "New Menu". It contains two input fields: "Name" and "Description". Below these fields is a blue button labeled "CREATE".

Рисунок 4.4. Создание нового меню



A dark-themed interface for "Menu 3". On the left, there is a form with "Name" (containing "завтраки") and "Description" fields, and a blue "UPDATE" button. On the right, there is a list of recipes. The list has a search bar at the top with a magnifying glass icon and a close button. Below the search bar, there are two items: "hey [3 cal]" and "test", each with a checkbox and an edit icon. At the bottom of the list is a blue "CREATE" button.

Рисунок 4.5. Добавление рецепта в меню



A close-up of the search bar from the previous figure. The search bar contains the text "з" and has a magnifying glass icon on the left and a close button on the right. Below the search bar, the first item "hey [3 cal]" is visible, with a checkbox and an edit icon.

Рисунок 4.6. Поиск рецептов

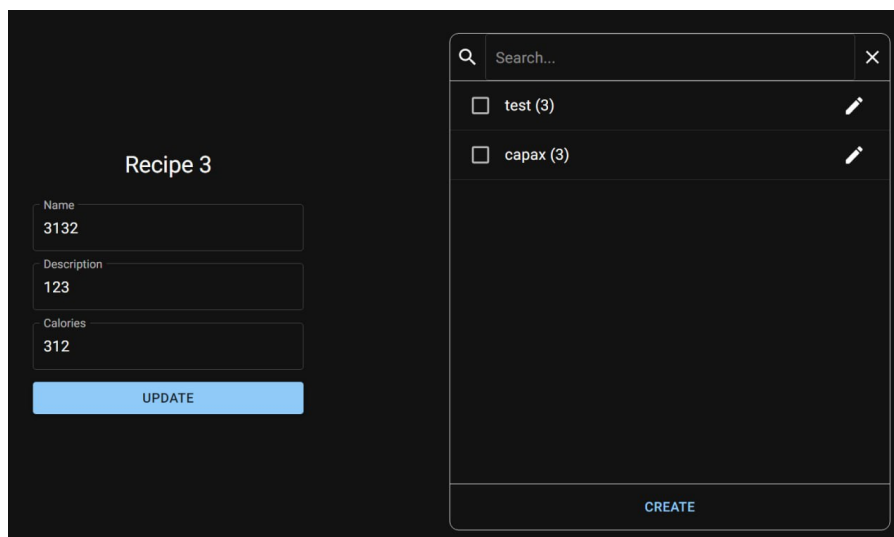


Рисунок 4.7. Обновление рецептов

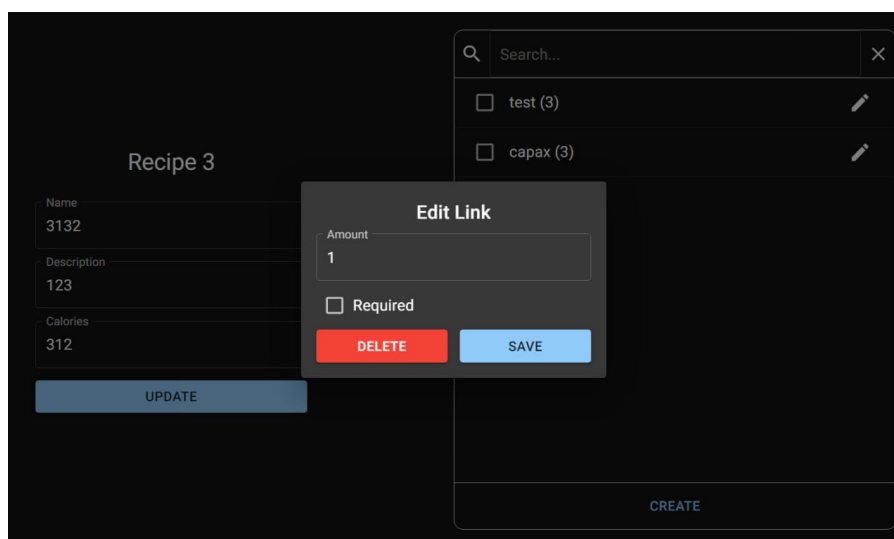


Рисунок 4.8. Редактирование связки ингредиент-рецепт

New Price

Nominal Amount
12

Nominal Cost
32

Min Count
1

Max Count
32

Ingredient
сахар

CREATE

Рисунок 4.9. Создание лота поставщика

Supplier

Name
test

Description

Address

UPDATE

Search... X

CREATE

Рисунок 4.10. Страница поставщика

4.2. Ссылки

Рабочий репозиторий: <https://github.com/BD-term-paper/term-web-application>.
Описание на умозрение пользователей.