ТЕХНИЧЕСКОЕ ЗАДАНИЕ НА ПРОЕКТИРОВАНИЕ DWH, СОЗДАНИЕ ПРОЦЕДУР И ФУНКЦИЙ ДЛЯ СЛУЖБЫ КУРЬЕРСКОЙ ДОСТАВКИ

ИСПОЛНИТЕЛЬ ПРОЕКТА: Кляпко Владислав Андреевич

ЦЕЛЬ:

Реализовать хранение и получение данных по курьерским заявкам о доставке корреспонденции контрагентам

СТАДИЯ ПРОЕКТА:

Реализованы frontend и backend

ЗАДАЧИ

1. Используя сервис **https://supabase.com** Необходимо поднять облачную базу данных PostgreSQL.

2. Для доступа к данным в базе данных должен быть создан пользователь.

Логин: netocourier

Пароль: *****

Права: полный доступ на схему **public**, к **information_schema** и **pg_catalog** права только на чтение, предусмотреть доступ к иным схемам, если они нужны.

3. Должны быть созданы отношения

Таблица №1 – Данные по заявкам курьера **(courier)**

№ п/п	Наименование поля в базе данных	Тип данных	Смысловое значение
1	2	3	4
1	id	uuid (PK)	id записи
2	from_place	varchar	откуда
3	where_place	varchar	куда
4	name	varchar	название документа
5	account_id	uuid (FK)	id контрагента
6	contact_id	uuid (FK)	id контакта
7	description	text	описание
8	user_id	uuid FK	id сотрудника отправителя
9	status	enum	статус заявки: - в очереди; - выполняется; - выполнено; - отменен *По умолчанию: «в очереди»*
10	created_date	date	дата создания заявки *По умолчанию: now()*

Таблица №2 – Список контрагентов (**account**)

№ п/п	Наименование поля в базе данных	Тип данных	Смысловое значение
1	2	3	4
1	id	uuid (PK)	id контрагента
2	name	varchar	название контрагента

Таблица №3 – Список контактов контрагентов (**contact**)

№ п/п	Наименование поля в базе данных	Тип данных	Смысловое значение
1	2	3	4
1	id	uuid (PK)	id записи
2	last_name	varchar	фамилия контакта
3	first_name	varchar	имя контакта
4	account_id	uuid (FK)	id контрагента

Таблица №4 – Сотрудники **(user)**

№ п/п	Наименование поля в базе данных	Тип данных	Смысловое значение
1	2	3	4
1	id	uuid (PK)	id записи
2	last_name	varchar	фамилия контакта
3	first_name	varchar	имя контакта
4	dismissed	boolean	уволен или нет (по умолчанию «нет»)

4. Для генерации uuid необходимо использовать функционал модуля uuid-ossp, который уже подключен в облачной базе.

5. Для формирования списка значений в атрибуте status необходимо использовать create type ... as enum.

6. Для возможности тестирования приложения необходимо реализовать процедуру **insert_test_data(value)**, которая принимает на вход целочисленное значение.

Данная процедура должна внести:

value * 1 строк случайных данных в отношение account.

value * 2 строк случайных данных в отношение contact.

value * 1 строк случайных данных в отношение user.

value * 5 строк случайных данных в отношение courier.

- Генерация id должна быть через uuid-ossp;

- Генерация символьных полей через конструкцию SELECT repeat(substring(«абвгдеёжзийклмнопрстуфхцчшщьыъэюя»,1,(random()*33)::integer),(random()*10)::integer).

Необходимо соблюдать длину типа varchar. Первый random получает случайный набор символов из строки, второй random дублирует количество символов полученных в substring.

- Генерация булева типа происходит через 0 и 1 с использованием оператора random.

- Генерацию даты и времени необходимо сформировать через select now() - interval «1 day» * round(random() * 1000) as timestamp.

7. Необходимо реализовать процедуру **erase_test_data()**, которая будет удалять тестовые данные из отношений.

8. На backend реализована функция по добавлению новой записи о заявке на курьера:

```
function add($params) --добавление новой заявки
    {
        $pdo = Di::pdo();
        $from = $params["from"];
        $where = $params["where"];
        $name = $params["name"];
        $account_id = $params["account_id"];
```

```
$contact_id = $params["contact_id"];
$description = $params["description"];
$user_id = $params["user_id"];
$stmt = $pdo->prepare(«CALL add_courier (?, ?, ?, ?, ?, ?, ?)»);
$stmt->bindParam(1, $from); --from_place
$stmt->bindParam(2, $where); --where_place
$stmt->bindParam(3, $name); --name
$stmt->bindParam(4, $account_id); --account_id
$stmt->bindParam(5, $contact_id); --contact_id
$stmt->bindParam(6, $description); --description
$stmt->bindParam(7, $user_id); --user_id
$stmt->execute();
}
```

Необходимо реализовать процедуру **add_courier(from_place, where_place, name, account_id, contact_id, description, user_id)**, которая принимает на вход вышеуказанные аргументы и вносит данные в таблицу courier.

9. На backend реализована функция по получению записей о заявках на курьера:

```
static function get() --получение списка заявок
{
$pdo = Di::pdo();
$stmt = $pdo->prepare(«SELECT * FROM get_courier()»);
$stmt->execute();
$data = $stmt->fetchAll();
return $data;
}
```

Необходимо реализовать функцию **get_courier()**, которая возвращает таблицу согласно структуре таблицы 5.

Таблица №5 – Результат функции get_courier()

№ п/п	Наименование поля в базе данных	Тип данных	Смысловое значение
1	2	3	4
1	id	uuid (PK)	id заявки
2	from_place	varchar	откуда
3	where_place	varchar	куда
4	name	varchar	название документа
5	account_id	uuid (FK)	id контрагента
6	account	varchar	название контрагента
7	contact_id	uuid (FK)	id контакта
8	contact	varchar	фамилия и имя контакта через пробел
9	description	text	описание
10	user_id	uuid FK	id сотрудника отправителя
11	user	varchar	фамилия и имя сотрудника через пробел
12	status	enum	статус заявки: - в очереди; - выполняется; - выполнено; - отменен
13	created_date	date	дата создания заявки

Сортировка результата должна быть сперва по статусу, затем по дате от большего к меньшему.

10. На backend реализована функция по изменению статуса заявки.

```
function change_status($params) --изменение статуса заявки
    {
        $pdo = Di::pdo();
        $status = $params["new_status"];
```

```
    $id = $params["id"];
    $stmt = $pdo->prepare(«CALL change_status(?, ?)»);
    $stmt->bindParam(1, $status); --новый статус
    $stmt->bindParam(2, $id); --идентификатор заявки
    $stmt->execute();
  }
```

Необходимо реализовать процедуру **change_status(status, id)**, которая будет изменять статус заявки. На вход процедура принимает новое значение статуса и значение идентификатора заявки.

11. На backend реализована функция получения списка сотрудников компании.

```
static function get_users() --получение списка пользователей
  {
    $pdo = Di::pdo();
    $stmt = $pdo->prepare(«SELECT * FROM get_users()»);
    $stmt->execute();
    $data = $stmt->fetchAll();
    $result = [];
    foreach ($data as $v) {
      $result[] = $v[«user»];
    }
    return $result;
  }
```

Необходимо реализовать функцию **get_users()**, которая возвращает таблицу согласно следующей структуры:

user --фамилия и имя сотрудника через пробел.

Сотрудник должен быть действующим! Сортировка должна быть по фамилии сотрудника.

12. На backend реализована функция получения списка контрагентов.

```
static function get_accounts() --получение списка контрагентов
  {
    $pdo = Di::pdo();
    $stmt = $pdo->prepare(«SELECT * FROM get_accounts()»);
    $stmt->execute();
    $data = $stmt->fetchAll();
    $result = [];
    foreach ($data as $v) {
      $result[] = $v[«account»];
    }
    return $result;
  }
```

Необходимо реализовать функцию **get_accounts()**, которая возвращает таблицу согласно следующей структуры:

account --название контрагента.

Сортировка должна быть по названию контрагента.

13. На backend реализована функция получения списка контактов.

```
function get_contacts($params) --получение списка контактов
  {
    $pdo = Di::pdo();
    $account_id = $params["account_id"];
    $stmt = $pdo->prepare(«SELECT * FROM get_contacts(?)»);
```

```
    $stmt->bindParam(1, $account_id); --идентификатор контрагента
    $stmt->execute();
    $data = $stmt->fetchAll();
    $result = [];
    foreach ($data as $v) {
        $result[] = $v[«contact»];
    }
    return $result;
}
```

Необходимо реализовать функцию **get_contacts(account_id)**, которая принимает на вход идентификатор контрагента и возвращает таблицу с контактами переданного контрагента согласно следующей структуры:

contact --фамилия и имя контакта через пробел.

Сортировка должна быть по фамилии контакта. Если в функцию вместо идентификатора контрагента передан null, Необходимо вернуть строку «Выберите контрагента».

14. На backend реализована функция по получению статистики о заявках на курьера:

```
static function get_stat() --получение статистики
{
    $pdo = Di::pdo();
    $stmt = $pdo->prepare(«SELECT * FROM courier_statistic»);
    $stmt->execute();
    $data = $stmt->fetchAll();
    return $data;
}
```

Необходимо реализовать представление **courier_statistic**, со структурой таблицы 6.

Таблица №6 – Представление **courier_statistic**

№ п/п	Наименование поля в базе данных	Смысловое значение
1	2	3
1	account_id	id контрагента
2	account	название контрагента
3	count_courier	количество заказов на курьера для каждого контрагента
4	count_complete	количество завершенных заказов для каждого контрагента
5	count_canceled	количество отмененных заказов для каждого контрагента
6	percent_relative_prev_month	процентное изменение количества заказов текущего месяца к предыдущему месяцу для каждого контрагента, если получаете деление на 0, то в результат вывести 0
7	count_where_place	количество мест доставки для каждого контрагента
8	count_contact	количество контактов по контрагенту, которым доставляются документы
9	cansel_user_array	массив с идентификаторами сотрудников, по которым были заказы со статусом "Отменен" для каждого контрагента