# WEB версия калькулятора расчёта стоимости услуги облачного решения для клиента

GitHub: <a href="https://github.com/mufteev/ertelecom.git">https://github.com/mufteev/ertelecom.git</a>

Технологии на клиентской стороне приложения	2
Технологии на серверной стороне приложения	2
Структура БД	3
Структура клиентского приложения	4
Структура серверного приложения	6
Процесс развёртывания БД	7
Процесс развёртывания приложения	7

## Технологии на клиентской стороне приложения

- JS-фреймворк react;
- UI-фреймворк **bootstrap** + **react-bootstrap**;
- Маршрутизация react-router-dom;
- Хранилище состояний **redux**:
  - o **react-redux** − ядро;
  - о reduxjs/toolkit позволяет более просто описать модульное хранилище;
- Прочие решения:
  - react-select выпадающий список с полем поиска для получения информации с сервера;
  - о **http-proxy-middleware** позволяет настроить перенаправление запросов, например начинающихся с /арі/\* на серверную часть.

## Технологии на серверной стороне приложения

- СУБД PostgreSQL;
- Веб-фреймворк **express**;
- Провайдер СУБД PostgreSQL pg;
- Прочие решения:
  - о **dotenv** загрузчик переменных сред настроек;
  - pdfmake библиотека для генерации PDF файлов декларативно задавая его форму в формате JSON;
  - о **log4js** библиотека позволяющая настраивать уровни логирования и потоки вывода информации о логах.

## Структура БД

БД состоит из 4 схем:

- **info** содержит справочники для логики калькулятора;
- info\_price содержит таблицы с возможными конфигурациями, удельными стоимостями, а также представления (View) для вычисления стоимостей;
- **history** содержит информацию о клиентах и рассчитанных менеджером предложениях, представление рассчитывающее окончательную сумму на основе хранящихся данных, а также хранимую процедуру «save», которая позволяет внести информацию о расчётах единой транзакцией.
- **public** содержит информацию о хранимых функциях добавленных расширением «uuid-ossp», а также функцию «get\_unit\_cost\_value» которая по акрониму получает какую либо удельную стоимость.

Диаграмма связей между объектами БД представлена во вложении «db.pdf».

## Структура клиентского приложения

Приложение состоит из двух основных разделов калькулятора, позволяющего рассчитать стоимость лицензии облачного решения и списка истории, в котором отображаются ранее произведённые расчёты с возможностью выгрузить их в формате PDF.

Переключение между разделами происходит по переходам к соответствующим вкладкам «Калькулятор» и «История», либо переходя по адресам «/» и «/history», соответственно.

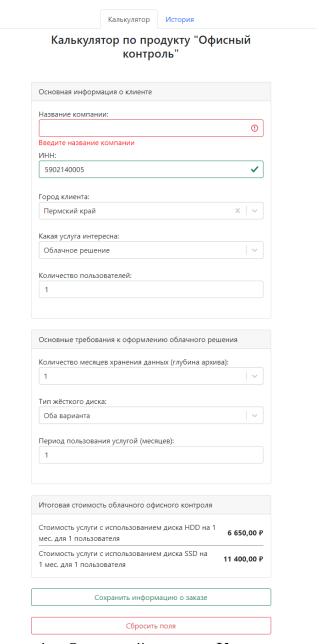
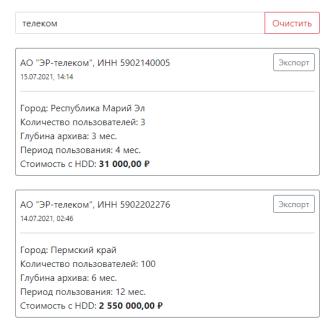


Рисунок 1 – Основной раздел «Калькулятор»

Калькулятор	История
, ,	

#### Ранее произведённые расчёты



### Рисунок 2 – История сохранённых расчётов

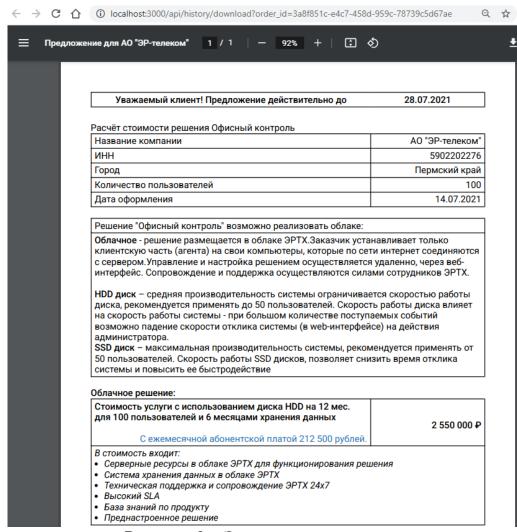


Рисунок 3 – Экспорт предложения

## Структура серверного приложения

Сервер аналогично разбит на 2 модуля с соответствующим АРІ:

- /api/calc маршрут для доступа к ресурсам калькулятора:
  - о GET: /get\_directory\_info получение данных из справочников для калькулятора;
  - GET: /get\_cities\_search?s=<text> фильтрация списка городов;
  - POST: /calculating вычисление стоимость услуг на основании представленных данных в теле запроса:

```
{
  users_count : int,
  period_service : int,
  archive_depth_id : uuid,
  type_storage_id : uuid
}
```

- /api/history маршрут для доступа к ресурсам истории:
  - POST: /search поиск по истории (пустой запрос выдаёт всю историю):

```
{
   query: string
}
```

- GET: /download?order\_id=<uuid> по идентификатору выгружает сохранённую запись оформленную в формате PDF;
- POST: /save сохраняет запись о клиенте и прочие необходимые параметры.

```
{
  company_name
                      : string
  users_count
                      : int
  period_service
                     : int
  company tin
                     : string
  city_id
                      : uuid
  type_storage_id
                     : uuid
  archive depth id
                     : uuid
  type_provision_id
                      : uuid
}
```

# Процесс развёртывания БД

- Установить СУБД PostgreSQL v13;
- Создать БД, например, с именем «ertelecom»;
- Перед восстановлением БД из файла выполнить добавление расширения для работы с UUID командой в консоли:

```
psql -d postgres -U postgres -c 'CREATE EXTENSION \"uuid-ossp\"'
```

– Затем выполнить восстановление БД из файла во вложении <u>db.sql</u>:

```
psql -d ertelecom -U postgres -q -f db.sql
```

где параметр -U <Имя пользователя>, -d <Имя\_БД>, a -f <Путь до бэкапа>

Важно, необходимо привести в соответствие данные о подключении для серверной стороны в файле <u>config.env</u>.

## Процесс развёртывания приложения

- Установить node.js (v14+);
- Клонировать исходный код из репозитория:

```
git clone https://github.com/mufteev/ertelecom.git
```

- Восстановление модулей серверного и клиентского приложения:

```
cd ertelecom
npm i
cd ./client
npm i
```

- Запуск сервера:

```
cd ertelecom
npm start
```

Запуск клиента:

```
cd ./client
npm start
```