

Работа с PostgreSQL на C++

Муравьев Кирилл,
242

<https://github.com/studokim/yachts/>

Обзор библиотек



Библиотеки

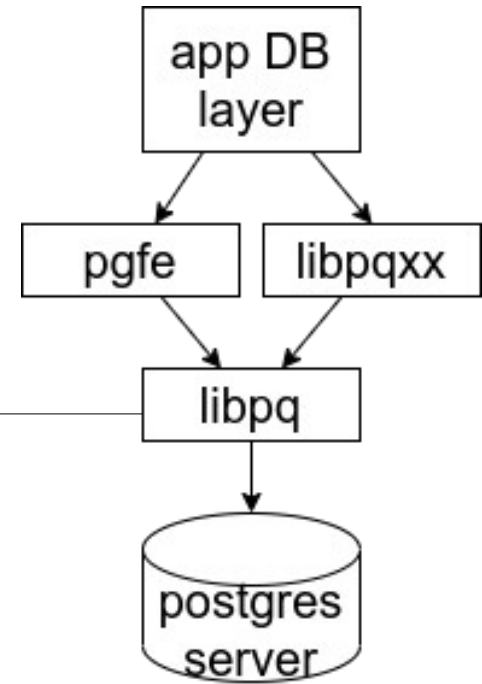
- libpq — C, официальная

- libpqxx — C++, “официальная”

- libpq++ — мертва

- dmitigr/pgfe — C++

- github.com/SOCI/soci — C++, cross-database



dmitigr/pgfe

- Ориентирована на простоту использования и надёжность;
- Слегка уступает в производительности libpq;
- Может быть использована header-only;
- Простой thread-safe пул соединений;
- Удобный вызов функций и процедур.

dmitigr/pgfe: выполнение запроса

```
void foo(Connection& conn)
{
    conn.execute("begin");
    conn.execute([](auto&& row)
    {
        using dmitigr::pgfe::to;           // see "Data conversions" section for details
        auto val = to<std::string>(row[0]); // converts the retrieved value to std::string
        std::cout << val << "\n";         // prints "Hi!\n"
    }, "select 'Hi!'");
    conn.execute("rollback");
}
```

dmitigr/pgfe vs libpqxx

- libpqxx — 30k упоминаний
- dmitigr/pgfe — 200 упоминаний

libpqxx

Основные используемые классы: **connection**, **transaction** и **result**.

- 1) Подключиться к БД, используя **pqxx::connection**.
- 2) Создать **transaction** для этого **connection**.
- 3) Вызвать функции **transaction's exec**, **query_value** или **stream** для выполнения SQL-запроса. Запрос представляет собой строку.
- 4) Вызвать метод **transaction.commit()**, чтобы сохранить результат. Иначе транзакция откатится при уничтожении соответствующего объекта.

libpqxx

- **pqxx::result** ведёт себя как контейнер строк: **pqxx::row**.
- **row** ведёт себя как контейнер полей: **pqxx::field**
- Данные поля **field's** хранятся как строка, в формате, определяемом PostgreSQL. Эти данные можно конвертировать, используя методы **as()** и **to()**.
- После закрытия транзакции объект **connection** готов к запуску следующей.

libpqxx: выполнение запроса

```
pqxx::connection conn("host=localhost port=5432 user=yachtsapp password=pwd  
dbname=yachts connect_timeout=10");
```

```
pqxx::connection conn("postgresql://yachtsapp:pwd@localhost:5432/yachts?  
connect_timeout=10");
```

```
void foo(Connection& conn)  
{  
    pqxx::work W{conn};  
    pqxx::result R{W.exec("SELECT name FROM employee")};  
    for (auto row: R)  
        std::cout << row[0].c_str() << '\n';  
    W.commit();  
}
```

Подготовка Postgres



Создание и удаление пользователя

```
1 CREATE USER yachtsapp WITH PASSWORD 'pwd';
2
3 GRANT CONNECT ON DATABASE yachts TO yachtsApp;
4 GRANT USAGE ON SCHEMA public TO yachtsApp;
5 GRANT SELECT, INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA public TO yachtsApp;
6 GRANT USAGE ON ALL SEQUENCES IN SCHEMA public TO yachtsApp;

39 REVOKE CONNECT ON DATABASE yachts FROM yachtsApp;
40 REVOKE ALL ON SCHEMA public FROM yachtsApp;
41 DROP ROLE IF EXISTS yachtsApp;
```

Настройки авторизации

add user to /etc/postgresql/12/main/pg_hba.conf

```
# Database administrative login by Unix domain socket
local    all             postgres                                peer

# TYPE      DATABASE     USER        ADDRESS            METHOD
host      yachts     yachtsapp    127.0.0.1/32       password
host      yachts     yachtsapp    172.0.0.2/8        password

# "local" is for Unix domain socket connections only
local     all             all                                peer
# IPv4 local connections:
host      all             all          127.0.0.1/32       md5
```

sudo systemctl restart postgresql

Применение Irqxx



Установка и подключение к проекту

`sudo apt install libpqxx-dev`

Cmake:

- `target_link_libraries(yachts pq)`
- `target_link_libraries(yachts pqxx)`

G++:

- `g++ main.cpp -lpqxx -lpq`

Database class

```
std::string Select(const std::string& query)
{
    if (!IsCorrect(query, keyword: "select"))
        return "ERROR: Wrong query!";
    pqxx::nontransaction work{ &: *conn};
    // now it looks like
    // select json_agg(t) from (select * from vclass) as t
    pqxx::result result{ .m_data: work.exec(JsonizeQuery(query))};
    return result[0][0].c_str();
}
```

```
std::string CallFunction(const std::string& query)
{
    if (!IsCorrect(query, keyword: "select"))
        return "ERROR: Wrong query!";
    try {
        pqxx::work work{ &: *conn};
        pqxx::result result{ .m_data: work.exec(JsonizeQuery(query))};
        work.commit();
        std::string res = result[0][0].c_str();
        std::string notice = Strip( message: handler->Message());
        handler->Reset();
        if (!notice.empty())
            // now it looks like
            // [{"keyword" : "result"}]
            return JsonizeResult( keyword: "Result", notice);
        if (!IsEmpty(res))
            return res;
        return JsonizeResult( keyword: "Result", result: "ok");
    }
    catch(std::exception& e) {
        std::string error = Strip( message: e.what());
        return JsonizeResult( keyword: "Error", error);
    }
}
```

Что делать, если функция возвращает Notice или Exception?

- В случае **exception** или неправильного запроса – **различные std::exception**.
- По умолчанию `lprqxx` игнорирует **notices**.
- Для обработки нужен кастомный **ErrorHandler**, наследуемый от **pqxx::errorhandler**.
- **Connection** принадлежит **handler**'у, а не наоборот.

ExceptionHandler

```
void Connect(bool force = false)
{
    if (force || conn == nullptr) {
        conn = new pqxx::connection(conn_string);
        handler = new PGEHandler(&*conn);
        std::cout << conn->get_errorhandlers().size() << std::endl;
    }
    std::cout << "Connected to " << conn->dbname() << '.' << std::endl;
}
```

```
public:
    PGEHandler() = delete;

    explicit PGEHandler(pqxx::connection& c) :
        pqxx::errorhandler(&c), message() {}

    bool operator()(char const msg[]) noexcept override
    {
        message = std::string{msg};
        return true;
    }

    std::string const& Message()
    {
        return message;
    }

    void Reset()
    {
        message.clear();
    }

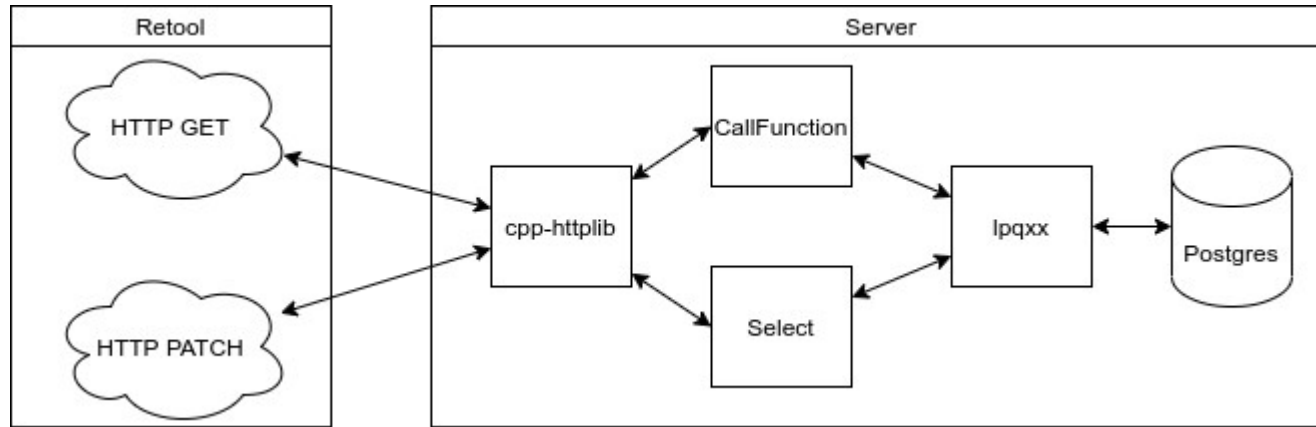
private:
    std::string message;
```

Строим REST API с PostgreSQL

Обзор библиотек



Структура



Библиотеки

[https://github.com/Stiffstream/**restinio**](https://github.com/Stiffstream/restinio)

[https://github.com/boostorg/**beast**](https://github.com/boostorg/beast)

[https://github.com/yhirose/**cpp-httpplib**](https://github.com/yhirose/cpp-httpplib)

[https://github.com/facebook/**proxygen**](https://github.com/facebook/proxygen)

cpp-httplib

- Single-file
- Header-only
- multi-threaded 'blocking'
- has built-in mappings
- И сервер, и клиент
- Множество примеров в Readme и репозитории

Extension	MIME Type	Extension	MIME Type
css	text/css	mpga	audio/mpeg
csv	text/csv	weba	audio/webm
txt	text/plain	wav	audio/wave
vtt	text/vtt	otf	font/otf
html, htm	text/html	ttf	font/ttf
apng	image/apng	woff	font/woff
avif	image/avif	woff2	font/woff2
bmp	image/bmp	7z	application/x-7z-compressed
gif	image/gif	atom	application/atom+xml
png	image/png	pdf	application/pdf
svg	image/svg+xml	mjs, js	application/javascript
webp	image/webp	json	application/json

API class

```
void Start(bool force = false)
{
    std::cout << "Server at " << host << ":" << port << " started." << std::endl;
    db->Connect( force: true);
    if (force || server == nullptr) {
        server = new Server();

        RegisterWelcomeHandler();
        RegisterGetCollectionHandlers();
        // We can go without insertionHandlers and HttpPost
        // if there are appropriate functions in the database.
        RegisterFunctionsHandlers();

        server->listen(host, port);
    }
}
```

```

void RegisterGetCollectionHandlers()
{
    server->Get( pattern: "/classes", handler: [this](const Request &req, Response &res) {...});

    server->Get( pattern: "/clients", handler: [this](const Request &req, Response &res) {...});

    server->Get( pattern: "/inspections", handler: [this](const Request &req, Response &res) {...});

    server->Get( pattern: "/invoices", handler: [this](const Request &req, Response &res) {...});

    server->Get( pattern: "/rents", handler: [this](const Request &req, Response &res) {
        auto result = db->SelectAll( tableName: "vrent", orderByColumn: 5, desc: true);
        res.set_content(result, content_type: "application/json");
    });

    server->Get( pattern: "/yachts", handler: [this](const Request &req, Response &res) {
        auto result = db->Select( query: "select vyacht.*, vclass.dailycost, vinspection.date as inspected,"
            "vinspection.statusok as ok, checktakenRent(vyacht.yachtid) as taken\n"
            " from vyacht join vclass on vclass.classid = vyacht.classid\n"
            " join vinspection on vinspection.yachtid = vyacht.yachtid\n"
            " where vinspection.date = (select max(date) from vinspection\n"
            " where vinspection.yachtid = vyacht.yachtid)");
        res.set_content(result, content_type: "application/json");
    });
}

```



```

void RegisterFunctionsHandlers()
{
    server->Get( pattern: "/discount", handler: [this](const Request &req, Response &res) {
        auto docId = req.get_param_value( key: "documentid");
        auto query = std::string( s: "select calculateddiscountfunction(\").append(docId).append( s: "\') as Discount");
        auto result = db->CallFunction(query);
        res.set_content(result, content_type: "application/json");
    });

    server->Get( pattern: "/revenue", handler: [this](const Request &req, Response &res) {...});

    server->Patch( pattern: "/prices", handler: [this](const Request &req, Response &res) {
        auto percent = req.get_param_value( key: "percent");
        auto query = std::string( s: "select updatepricesfunction(").append(percent).append( s: ") as Result");
        auto result = db->CallFunction(query);
        res.set_content(result, content_type: "application/json");
    });

    server->Patch( pattern: "/pay", handler: [this](const Request &req, Response &res) {...});
}

```

main.cpp

```
try
{
    HTTPServer* srv = new HTTPServer( host: "192.168.0.1", port: "5431", db: new PGClient( user: "yachtsapp", password: "pwd"));

    std::thread th_srv(&HTTPServer::Start, srv, true);

    std::this_thread::sleep_for( rtime: std::chrono::minutes ( rep: 20));

    srv->Stop();
    th_srv.join();
}
catch (std::exception const &e)
{
    std::cerr << e.what() << '\n';
    return 1;
}
return 0;
```

NoCode-frontend

так теперь модно

Доступные бесплатные решения

<https://retool.com/>

<https://appgyver.com/>


Retool

- Ориентирован на внутреннее использование;
- Можно развернуть **локально** в docker;
- Умеет подключаться примерно ко всем видам API;
- Позволяет использовать несколько “ресурсов” в проекте.

Retool: БД

DATABASES

 Postgres

 MySQL

 Microsoft SQL

 MongoDB

 Cassandra

 CosmosDB

 Amazon Redshift

 Amazon Athena

 BigQuery

 Elastic Search

 CouchDB

 RethinkDB

 Snowflake

 Denodo

 Redis

 DynamoDB

 Oracle DB

 Vertica

 Presto

 Cloud Datastore

 SAP Hana

Retool: API

APIS



REST API



GraphQL



Stripe



Twilio



GitHub



Lambda



Slack



Salesforce



Amazon S3



Google Cloud Storage



Google Sheets



SendGrid



Firebase



Basecamp



Close.io



Open API



SMTP



Asana



BigID



Datadog



Jira



gRPC

Can't find your database type? [Let us know.](#)

Подготовка Postgres



Настройки авторизации

Вторая строка /etc/postgresql/12/main/pg_hba.conf

(Docker автоматически разворачивается на IP 172.x.x.x)

```
# Database administrative login by Unix domain socket
local    all             postgres                                peer

# TYPE      DATABASE      USER      ADDRESS              METHOD
host      yachts     yachtsapp  127.0.0.1/32         password
host      yachts     yachtsapp  172.0.0.2/8          password

# "local" is for Unix domain socket connections only
local     all             all                                peer
# IPv4 local connections:
host      all             all             127.0.0.1/32         md5
```

sudo systemctl restart postgresql

Настройка Docker (опционально)

```
# show all working containers
```

```
sudo docker ps
```

```
# docker exec -it <containerid> <entrypoint>
```

```
sudo docker exec -it 2755fcc5401f bash
```

```
# ensure docker can reach my real ip
```

```
ping 192.168.43.115
```

Настройки сети

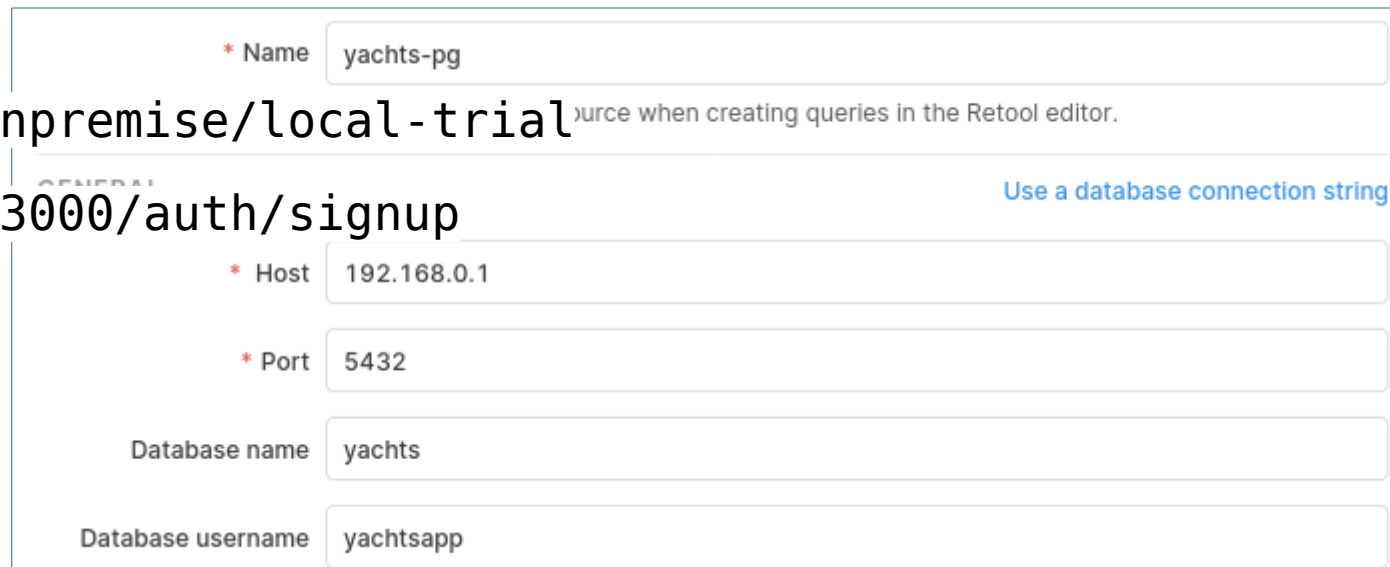
- **Фиксируем** ip, на который будет ссылаться контейнер

```
sudo docker network create -d bridge --subnet  
192.168.0.0/16 --gateway 192.168.0.1 mynet
```

- Запускаем Retool

```
~/retool/retool-onpremise/local-trial
```

```
http://localhost:3000/auth/signup
```





The screenshot shows the Retool configuration interface. It includes a 'Name' field with the value 'yachts-pg'. Below it is a 'GENERAL' section with a link 'Use a database connection string'. The 'Host' field is set to '192.168.0.1', the 'Port' field is set to '5432', the 'Database name' field is set to 'yachts', and the 'Database username' field is set to 'yachtsapp'.


* Name	yachts-pg
Use a database connection string	
* Host	192.168.0.1
* Port	5432
Database name	yachts
Database username	yachtsapp


Запрос к Postgres


+ New ▾


Showing all 5 

 getClientInvoices

 getYachts


 welcome

 getDiscount

 patchPay

General Response Advanced

getClientInvoices

Resource  yachts-pg (postgresql) Edit resource ▾

SQL mode ▾ Run query automatically when inputs change ▾

```
1 select * from vclientinvoice
2 where documentid::text like {{filterSQL_DocumentId.value+ "%"}}
3 AND name::text like {{"%" +filterSQL_Name.value+ "%"}}
4 AND phonenumber::text like {{"%" +filterSQL_PhoneNumber.value+ "%"}}
5 AND NOT (paiddat IS NULL AND {{filterSQL_Paid.value}})
6 order by issuedate desc
7 limit 100
```

Запрос к REST API

Showing all 5

getClientInvoices

getYachts

welcome

getDiscount

patchPay

Resource

yachts-cpp (restapi)

Edit resource

Run query only when manually triggered

Action type

PATCH

http://192.168.0.1:5431/

pay?documentid={{valuePayDocumentId.value}}&amount={{valuePayAmount.value}}&method={{valuePayMethod.value ? 'card' : 'cash'}}

(PATCH) Will only run in response to a user interaction (i.e. button click).

URL parameters

documentid	{{valuePayDocumentId.value}}
amount	{{valuePayAmount.value}}