




 [canove](#) / [whaticket-community](#) Público

A very simple Ticket System based on WhatsApp messages, that allow multi-users in same WhatsApp account.

 MIT license 1k stars  586 forks  Activity Star Notifications Código Problemas 31 Requisições pull 2 Ações Projetos Segurança Perc mestre ▾

ir para o arquivo



canove ...

✖ em 17 de maio ⌚

[View code](#)

Donate

PayPal

license scan

passing

 quality gate

failed

 maintainability

A

 chat

84 online

 forum

online

## WhaTicket

**NOTA** : A nova versão do whatsapp-web.js requer o Node 14. Atualize suas instalações para continuar usando.

Um Sistema de Tickets *muito simples* baseado em mensagens do WhatsApp.

O backend usa [o whatsapp-web.js](#) para receber e enviar mensagens do WhatsApp, criar tickets a partir deles e armazenar tudo em um banco de dados MySQL.

*Frontend é um aplicativo de bate -papo multiusuário completo* inicializado com react-create-app e Material UI, que se comunica com back-end usando REST API e Websockets. Ele permite que você interaja com contatos, tickets, envie e receba mensagens do WhatsApp.

**NOTA** : Não posso garantir que você não será bloqueado usando este método, embora tenha funcionado para mim. O WhatsApp não permite bots ou clientes não oficiais em sua plataforma, então isso não deve ser considerado totalmente seguro.

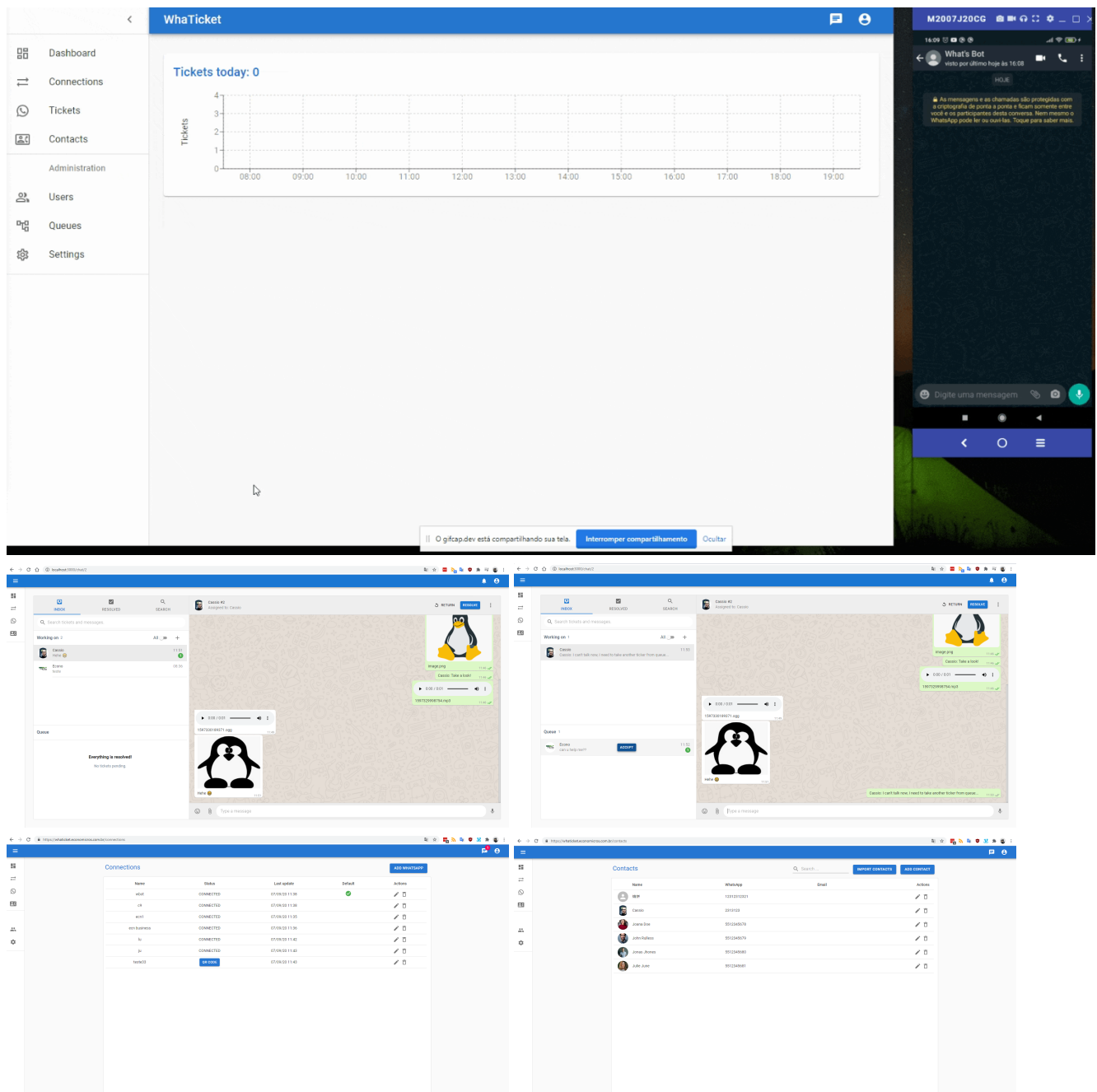
## Como funciona?

A cada nova mensagem recebida em um WhatsApp associado, um novo Ticket é criado. Em seguida, esse ticket pode ser acessado em uma *fila* na página *Tickets*, onde você pode atribuir o ticket a si mesmo, aceitando-o, respondendo a mensagem do ticket e, eventualmente, *resolvendo*-o.

As mensagens subsequentes do mesmo contato serão relacionadas ao primeiro ticket **aberto/pendente** encontrado.

Caso um contato envie uma nova mensagem em menos de 2 horas de intervalo, e não haja ticket deste contato com status **pendente/aberto**, o ticket **fechado** mais recente será reaberto, ao invés de criar um novo.

## Capturas de tela



## Características

- Ter vários usuários conversando no mesmo número do WhatsApp ✓
- Conecte-se a várias contas do WhatsApp e receba todas as mensagens em um só lugar ✓ NEW
- Crie e converse com novos contatos sem tocar no celular ✓
- Enviar e receber mensagem ✓
- Enviar mídia (imagens/áudio/documentos) ✓
- Receber mídia (imagens/áudio/vídeo/documentos) ✓

## Instalação e Uso (Linux Ubuntu - Desenvolvimento)

Crie um banco de dados Mysql usando a janela de encaixe: *Observação* : altere MYSQL\_DATABASE, MYSQL\_PASSWORD, MYSQL\_USER e MYSQL\_ROOT\_PASSWORD.

```
docker run --name whaticketdb -e MYSQL_ROOT_PASSWORD=strongpassword -e MYSQL_DATABASE=whaticket

# Or run using `docker-compose` as below
# Before copy .env.example to .env first and set the variables in the file.
docker-compose up -d mysql

# To administer this mysql database easily using phpmyadmin.
# It will run by default on port 9000, but can be changed in .env using `PMA_PORT`
docker-compose -f docker-compose.phpmyadmin.yaml up -d
```

Instale as dependências do marionetista:

```
sudo apt-get install -y libxshmfence-dev libgbm-dev wget unzip fontconfig locales gconf-service
```

Clonar este repositório

```
git clone https://github.com/canove/whaticket/ whaticket
```

Vá para a pasta back-end e crie o arquivo .env:

```
cp .env.example .env
nano .env
```

Preencha .env o arquivo com as variáveis de ambiente:

```
NODE_ENV=DEVELOPMENT      #it helps on debugging
BACKEND_URL=http://localhost
FRONTEND_URL=https://localhost:3000
PROXY_PORT=8080
PORT=8080

DB_HOST=                  #DB host IP, usually localhost
DB_DIALECT=
DB_USER=
DB_PASS=
DB_NAME=

JWT_SECRET=3123123213123
JWT_REFRESH_SECRET=75756756756
```

Instale dependências de back-end, crie aplicativos, execute migrações e sementes:

```
npm install
npm run build
```

```
npx sequelize db:migrate  
npx sequelize db:seed:all
```

Iniciar back-end:

```
npm start
```



Abra um segundo terminal, vá para a pasta frontend e crie o arquivo .env:

```
nano .env  
REACT_APP_BACKEND_URL = http://localhost:8080/ # Your previous configured backend app URL.
```



Inicie o aplicativo front-end:

```
npm start
```



- Vá para [http://your\\_server\\_ip:3000/signup](http://your_server_ip:3000/signup)
- Crie um usuário e faça o login com ele.
- Na barra lateral, vá para a página *Conexões* e crie sua primeira conexão do WhatsApp.
- Aguarde o botão QR CODE aparecer, clique nele e leia o código qr.
- Feito. Todas as mensagens recebidas pelo seu número do WhatsApp sincronizado aparecerão na lista de tickets.

## Implantação de produção básica

### Usando Ubuntu 20.04 VPS

Todas as instruções abaixo assumem que você NÃO está executando como root, pois dará um erro no marionetista. Então, vamos começar criando um novo usuário e concedendo privilégios sudo a ele:

```
adduser deploy  
usermod -aG sudo deploy
```



Agora podemos fazer o login com este novo usuário:

```
su deploy
```



Você precisará de dois subdomínios encaminhados para o ip do seu VPS para seguir estas instruções. Usaremos `myapp.mydomain.com` para front-end e `api.mydomain.com` back-end no exemplo a seguir.

Atualize todos os pacotes do sistema:

```
sudo apt update && sudo apt upgrade
```



Instale o nó e confirme se o comando do nó está disponível:

```
curl -fsSL https://deb.nodesource.com/setup_14.x | sudo -E bash -
sudo apt-get install -y nodejs
node -v
npm -v
```



Instale o docker e adicione seu usuário ao grupo docker:

```
sudo apt install apt-transport-https ca-certificates curl software-properties-common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic s
sudo apt update
sudo apt install docker-ce
sudo systemctl status docker
sudo usermod -aG docker ${USER}
su - ${USER}
```



Crie um banco de dados Mysql usando a janela de encaixe: *Observação* : altere MYSQL\_DATABASE, MYSQL\_PASSWORD, MYSQL\_USER e MYSQL\_ROOT\_PASSWORD.

```
docker run --name whaticketdb -e MYSQL_ROOT_PASSWORD=strongpassword -e MYSQL_DATABASE=whati

# Or run using `docker-compose` as below
# Before copy .env.example to .env first and set the variables in the file.
docker-compose up -d mysql

# To administer this mysql database easily using phpmyadmin.
# It will run by default on port 9000, but can be changed in .env using `PMA_PORT`
docker-compose -f docker-compose.phpmyadmin.yaml up -d
```



Clone este repositório:

```
cd ~
git clone https://github.com/canove/whaticket whaticket
```



Crie um arquivo .env de back-end e preencha com os detalhes:

### ☰ README.md

```
nano whaticket/backend/.env
```

```
NODE_ENV=
BACKEND_URL=https://api.mydomain.com #USE HTTPS HERE, WE WILL ADD SSL LATTER
FRONTEND_URL=https://myapp.mydomain.com #USE HTTPS HERE, WE WILL ADD SSL LATTER, CORS REL
PROXY_PORT=443 #USE NGINX REVERSE PROXY PORT HERE, WE WILL CONFI
PORT=8080

DB_HOST=localhost
DB_DIALECT=
DB_USER=
DB_PASS=
```



```
DB_NAME=
```

```
JWT_SECRET=3123123213123
```

```
JWT_REFRESH_SECRET=75756756756
```

Instale as dependências do marionetista:

```
sudo apt-get install -y libxshmfence-dev libgbm-dev wget unzip fontconfig locales gconf-ser
```

Instale dependências de back-end, crie aplicativos, execute migrações e sementes:

```
cd whaticket/backend
npm install
npm run build
npx sequelize db:migrate
npx sequelize db:seed:all
```

Comece com `npm start`, você deve ver: `Server started on port...` no console. Bata `CTRL + c` para sair.

Instale **pm2** com **sudo** e inicie o back-end com ele:

```
sudo npm install -g pm2
pm2 start dist/server.js --name whaticket-backend
```

Faça pm2 iniciar automaticamente após a reinicialização:

```
pm2 startup ubuntu -u `YOUR_USERNAME`
```

Copie a última linha resultante do comando anterior e execute-o, é algo como:

```
sudo env PATH=$PATH:/usr/bin pm2 startup ubuntu -u YOUR_USERNAME --hp /home/YOUR_USERNAME
```

Vá para a pasta frontend e instale as dependências:

```
cd ../frontend
npm install
```

Crie um arquivo `.env` de front-end e preencha APENAS com seu endereço de back-end, deve ficar assim:

```
REACT_APP_BACKEND_URL = https://api.mydomain.com/
```

Criar aplicativo de front-end:

```
npm run build
```

Inicie o frontend com pm2 e salve a lista de processos pm2 para iniciar automaticamente após a reinicialização:

```
pm2 start server.js --name whaticket-frontend
pm2 save
```



To check if it's running, run `pm2 list`, it should look like:

```
deploy@ubuntu-whats:~$ pm2 list
```



id	name	namespace	version	mode	pid	uptime	.
1	whaticket-frontend	default	0.1.0	fork	179249	12D	0
6	whaticket-backend	default	1.0.0	fork	179253	12D	15

Install nginx:

```
sudo apt install nginx
```



Remove nginx default site:

```
sudo rm /etc/nginx/sites-enabled/default
```



Create a new nginx site to frontend app:

```
sudo nano /etc/nginx/sites-available/whaticket-frontend
```



Edit and fill it with this information, changing `server_name` to yours equivalent to `myapp.mydomain.com` :

```
server {
    server_name myapp.mydomain.com;

    location / {
        proxy_pass http://127.0.0.1:3333;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_cache_bypass $http_upgrade;
    }
}
```



Create another one to backend api, changing `server_name` to yours equivalent to `api.mydomain.com`, and `proxy_pass` to your localhost backend node server URL:

```
sudo cp /etc/nginx/sites-available/whaticket-frontend /etc/nginx/sites-available/whaticket-  
sudo nano /etc/nginx/sites-available/whaticket-backend
```

```
server {  
    server_name api.mydomain.com;  
  
    location / {  
        proxy_pass http://127.0.0.1:8080;  
        .....  
    }  
}
```

Create a symbolic links to enable nginx sites:

```
sudo ln -s /etc/nginx/sites-available/whaticket-frontend /etc/nginx/sites-enabled  
sudo ln -s /etc/nginx/sites-available/whaticket-backend /etc/nginx/sites-enabled
```

By default, nginx limit body size to 1MB, which isn't enough for some media uploads. Lets change it to 20MB, adding a new line to config file:

```
sudo nano /etc/nginx/nginx.conf  
...  
http {  
    ...  
    client_max_body_size 20M; # HANDLE BIGGER UPLOADS  
}
```

Test nginx configuration and restart server:

```
sudo nginx -t  
sudo service nginx restart
```

Now, enable SSL (https) on your sites to use all app features like notifications and sending audio messages. An easy way to this is using Certbot:

Install certbot:

```
sudo snap install --classic certbot  
sudo apt update
```

Enable SSL on nginx (Fill / Accept all information required):

```
sudo certbot --nginx
```

## Using docker and docker-compose

To run WhaTicket using docker you must perform the following steps:



```
cp .env.example .env
```



Now it will be necessary to configure the .env using its information, the variables are the same as those mentioned in the deployment using ubuntu, with the exception of mysql settings that were not in the .env.

```
# MYSQL
MYSQL_ENGINE=                                # default: mariadb
MYSQL_VERSION=                               # default: 10.6
MYSQL_ROOT_PASSWORD=strongpassword           # change it please
MYSQL_DATABASE=whaticket
MYSQL_PORT=3306                               # default: 3306; Use this port to expose mysql serv
TZ=America/Fortaleza                         # default: America/Fortaleza; Timezone for mysql

# BACKEND
BACKEND_PORT=                                # default: 8080; but access by host not use this po
BACKEND_SERVER_NAME=api.mydomain.com
BACKEND_URL=https://api.mydomain.com
PROXY_PORT=443
JWT_SECRET=3123123213123                     # change it please
JWT_REFRESH_SECRET=75756756756               # change it please

# FRONTEND
FRONTEND_PORT=80                             # default: 3000; Use port 80 to expose in productio
FRONTEND_SSL_PORT=443                        # default: 3001; Use port 443 to expose in producti
FRONTEND_SERVER_NAME=myapp.mydomain.com
FRONTEND_URL=https://myapp.mydomain.com

# BROWSERLESS
MAX_CONCURRENT_SESSIONS=                     # default: 1; Use only if using browserless
```



After defining the variables, run the following command:

```
docker-compose up -d --build
```



On the first run it will be necessary to seed the database tables using the following command:

```
docker-compose exec backend npx sequelize db:seed:all
```



## SSL Certificate

To deploy the ssl certificate, add it to the ssl/certs folder. Inside it there should be a backend and a frontend folder, and each of them should contain the files fullchain.pem and privkey.pem, as in the structure below:

```
.
├── certs
│   ├── backend
│   │   ├── fullchain.pem
│   │   └── privkey.pem
│   └── frontend
│       └── fullchain.pem
```



```
|      └─ privkey.pem
└─ www
```

To generate the certificate files use `certbot` which can be installed using `snap`, I used the following command:

Note: The frontend container that runs `nginx` is already prepared to receive the request made by `certbot` to validate the certificate.

```
# BACKEND
certbot certonly --cert-name backend --webroot --webroot-path ./ssl/www/ -d api.mydomain.co

# FRONTEND
certbot certonly --cert-name frontend --webroot --webroot-path ./ssl/www/ -d myapp.mydomain
```



## Access Data

User: [admin@whaticket.com](mailto:admin@whaticket.com) Password: admin

## Upgrading

WhaTicket is a working in progress and we are adding new features frequently. To update your old installation and get all the new features, you can use a bash script like this:

**Note:** Always check the `.env.example` and adjust your `.env` file before upgrading, since some new variable may be added.

```
nano updateWhaticket
```



```
#!/bin/bash
echo "Updating Whaticket, please wait."
```



```
cd ~
cd whaticket
git pull
cd backend
npm install
rm -rf dist
npm run build
npx sequelize db:migrate
npx sequelize db:seed
cd ../frontend
npm install
rm -rf build
npm run build
pm2 restart all

echo "Update finished. Enjoy!"
```

Make it executable and run it:

```
chmod +x updateWhaticket
./updateWhaticket
```



## Contributing

This project helps you and you want to help keep it going? Buy me a coffee:



Para doações em BRL, utilize o Paypal:



Qualquer ajuda e sugestões serão apreciadas.

## Isenção de responsabilidade

Comecei a aprender Javascript há alguns meses e este é meu primeiro projeto. Pode ter problemas de segurança e muitos bugs. Eu recomendo usá-lo apenas na rede local.

Este projeto não é afiliado, associado, autorizado, endossado ou de qualquer forma oficialmente conectado com o WhatsApp ou qualquer uma de suas subsidiárias ou afiliadas. O site oficial do WhatsApp pode ser encontrado em <https://whatsapp.com>. "WhatsApp", bem como nomes, marcas, emblemas e imagens relacionados são marcas registradas de seus respectivos proprietários.

## Lançamentos

Nenhum lançamento publicado

## Pacotes 2

 **whaticket-community-back-end**

 **whaticket-community-frontend**

## Contribuintes 26



[+ 15 colaboradores](#)

## Línguas

● JavaScript 66,1%   ● TypeScript 32,9%   ● Outro 1,0%