ООО «КВАНТУМ АРТ»

115184, Москва, Озерковский переулок, д. 12

тел. (495) 783-65-74

Модуль React для продукта QP8.WidgetPlatform

Руководство по установке (Linux)

Москва
2023

История изменений

Версия	Дата	Автор	Описание
1.1	16.10.2023	Молькова М.Е.	Обновление структуры документа
1.0	05.07.2023	Григорьева М.А.	Первичное техническое описание

Оглавление

1. Обозначения

Обозначение	Описание	Пример использования
Технические данные	Используется для выделения различных технических данных в тексте: URL, названия свойств и методов, имена файлов и т.п.	ГПИ Системы доступен по URL `https://www.domainname.zone/`.
Код	Пример кода.	`public DataTable Data { get; set; }`
Переменная	Используется для указания переменного значения.	Формат URL: *Базовый URI/Псевдоним объекта*
Требует дополнения	TBD (to be determined). Указывает, что необходима доработка текста – проверка корректности утверждения, детализация, правка после внесения изменений в документ и т.п.	Система работает с одной БД.
Примечание:	Дополнительные данные справочного характера.	**Примечание:** используется при генерации классов LINQ to SQL.
Внимание:	Важные данные, которые требуется обязательно учитывать.	**Внимание:** опция поддерживается только ASP-сборкой в целях совместимости.

2. Определения и термины, используемые в документе

2.1. Общие термины и определения

В таблице ниже приведено описание используемых терминов и определений.

Термин или определение	Описание
API	«Application Programming Interface» (интерфейс программирования приложений) – набор правил по использованию функциональных возможностей Системы, предоставляемый разработчикам для организации взаимодействия сторонних программных продуктов с Системой
QP8.CMS или **QP8.CMS с поддержкой PostgreSQL** (далее **«QP»**)	Программный продукт, обладающий широким спектром возможностей для разработки программной части Системы различной сложности
QP8.WidgetPlatform (также **«Виджетная платформа»**)	Расширяет возможности QP. Позволяет через бэкенд наполнять веб-страницы Системы самостоятельно разработанными модульными приложениями. Виджетная платформа и виджеты основаны на шаблоне архитектуры MVC (от англ. «Model-View-Controller», «Модель-Представление-Контроллер»)
WP.API	API виджетной платформы QP8.Widgets
URL	«Uniform Resource Locator» – система унифицированных адресов электронных ресурсов, или единообразный определитель местонахождения ресурса
БД	База данных
ГПИ	Графический пользовательский интерфейс
REST	«Representational State Transfer» – архитектурный стиль, определяющий правила взаимодействия между клиентом и сервером
HTTP	«HyperText Transfer Protocol» – протокол передачи гипертекста
Docker	Программная платформа для быстрой сборки, отладки и развертывания приложений с помощью контейнеров
Kubernetes	Инструмент оркестрации контейнеров с открытым исходным кодом, который позволяет беспрепятственно развертывать и масштабировать контейнерные приложения, управлять ими в различных производственных средах
nginx	Программное обеспечение с открытым исходным кодом, используемое в качестве почтового или обратного прокси-сервера
NPM	«Node Package Manager» – пакетный менеджер для JavaScript, работающий на Node.js
JSON	«JavaScript Object Notation» — текстовый формат хранения и передачи структурированных данных в формате «ключ:значение»
DNS	«Domain Name System» — система доменных имен

CORS	«Cross-Origin Resource Sharing» — механизм обеспечения безопасности и защиты пользователей, позволяющий определить список ресурсов, к которым страница может получить доступ

2.2. Термины QP

В таблице ниже приведены термины QP.

Термин или определение	Описание
GraphQL	Отдельный сервис для доступа к данным QP при разработке новых сайтов/продуктов, подключаемый к QP как плагин
Бэкенд	Копия QP; обладает ГПИ для работы с содержимым БД Системы
Контент	Раздел сайта, отвечающий за содержание пользовательской таблицы БД Системы и ее настройки
Поле	Атрибут контента. С использованием полей формируется структура данных для контента.
Сайт	Набор данных в бэкенде. Допускается создание нескольких сайтов. Содержимое каждого сайта определяется созданным в нём контентом.

3. Общие сведения

Модуль **React** для продукта **QP8.WidgetPlatform** (или QP8.WidgetPlatform.React) – это набор средств, позволяющих разработать сайт на технологии *React* с использованием возможностей виджетной платформы **QP8.WidgetPlatform**.

3.1. Состав модуля

Модуль состоит из:

- библиотек @quantumart/qp8-widget-platform-shell-core, @quantumart/qp8-widget-platform-shell, @quantumart/qp8-widget-platform-module, @quantumart/qp8-widget-platform-bridge (доступны в репозитории *npm.js*, включаются в состав сайтов, разрабатываемых на технологии *React*);
- сервиса API виджетной платформы;

Также в состав дистрибутива входит демо-сайт, показывающий возможности виджетной платформы на технологии *React*.

Внимание: Модуль React не является мультитенантным и устанавливается для конкретного *customer code*.

Внимание: Компонент WP.API может использоваться только либо для live, либо для stage (по умолчанию). Если необходимы оба режима, нужно установить ещё один экземпляр сервиса, поменяв название сервиса и порт.

Внимание: Компонент WP.API (сервис API виджетной платформы) входит также в состав модуля Angular. В случае одновременной установки обоих модулей можно использовать уже установленный компонент и закомментировать соответствующие разделы манифестов / не выполнять соответствующие шаги установки, либо установить ещё один экземпляр сервиса, поменяв название сервиса и порт.

4. Системные требования

Модуль **React** требует для своей работы установленный продукт **QP8.WidgetPlatform**.

Внимание: Для установки демо-сайта необходим предварительная установка модуля **QP.GraphQL.**

Внимание: При установке модуля **React** без использования Docker требуется предварительная установка Node.js 18 (или выше).

5. Установка QP8.WidgetPlatform.React (с использованием Docker)

1. Скачать архив widget-react-config.tar и распаковать его содержимое в `/etc/widget-react-config`.
2. Перейти в папку `/etc/widget-react-config`.
3. В файле `settings.json` задать:
   - внешний URL приложения **WP.API** в параметре `widgets.apiUrl` вместо `http://localhost:6200`, если этот этот внешний URL известен на текущий момент, и если планируется, что пользователь будет открывать приложение через браузер с другого компьютера;
   - внешний URL приложения **QP.GraphQL** в параметре `widgets.graphql.apiUrl` вместо `http://localhost:6300`, если этот этот внешний URL известен на текущий момент, и если планируется, что пользователь будет открывать приложение через браузер с другого компьютера.

Внимание: URL приложения QP.GraphQL должен заканчиваться на /graphql или /graphql/stage

4. Перейти в папку `/etc/widget-react-config/compose`.
5. В файле `.env` задать:
   - строку подключения к ранее развёрнутой базе демосайта в параметре `CONNECTION_STRING` (можно скопировать из конфигурационного файла QP);
   - внешний URL сервиса `react-module` в параметре `MODULE_CLIENT_URL` вместо `http://localhost:6500`, если этот этот внешний URL известен на текущий момент, и если планируется, что пользователь будет открывать приложение через браузер с другого компьютера;
6. Выполнить команду:

```
sudo docker-compose up -d
```

6. Установка QP8.WidgetPlatform.React (с использованием Kubernetes)

1. Скачать архив widget-react-config.tar и распаковать его содержимое в `/etc/widget-react-config`.
2. Перейти в папку `/etc/widget-react-config/k8s`.
3. В файле `widget.yml` можно настроить:
   - строку подключения к ранее развёрнутой базе демо-сайта в параметре `QpSettings__ConnectionString` (можно скопировать из конфигурационного файла QP);
   - внешний DNS приложения **WP.API** в параметре `widgets.apiUrl` внутри `settings.json`, настраиваемого в *configmap*, вместо `wp-demosite-rus-api.test`;
   - внешний DNS приложения **QP.GraphQL** в параметре `widgets.graphql.apiUrl` внутри `settings.json`, настраиваемого в *configmap*, вместо `graphql-demosite-rus-api.test`.

Внимание: URL приложения QP.GraphQL должен заканчиваться на /graphql или /graphql/stage

4. Развернуть сервисы командой:

```
kubectl apply –f widget.yml
```

5. При необходимости можно настроить *ингрессы* к сервисам `wp-demosite-rus-api`, `wp-demosite-rus-react-module` и `wp-demosite-rus-react-shell` в файле `ing.yaml`, задав свои *DNS* вместо `wp-demosite-rus-api.test`, `wp-demosite-rus-react-module.test` и `wp-demosite-rus-react-shell.test` соответственно и выполнив команду:

```
kubectl apply -f ing.yaml
```

7. Установка QP8.WidgetPlatform.React (без использования Docker)

При установке из скомпилированного кода:

1. Скачать архив, содержащий бинарные файлы API виджетной платформы (**WP.API**) и демо-сайта *React* (**WP.DemositeRus.React**).
2. Распаковать полученный архив в домашний каталог пользователя, созданного в рамках установки **QP8.CMS** (по умолчанию - `/home/qp`). Должна получиться следующая структура из двух каталогов:
   - `WP.API`;
   - `WP.DemositeRus.React.Shell`.
   - `WP.DemositeRus.React.Module`.

7.1. Установка WP.API

При сборке из исходников:

1. Вытянуть исходники из репозитория на github.
2. В папке проекта `QA.WidgetPlatform.Api` (где должен быть файл `QA.WidgetPlatform.Api.csproj`) выполнить команду на публикацию:

```
dotnet publish "QA.WidgetPlatform.Api.csproj" -c Release -o
bin/release/publish/ -r linux-x64 --self-contained=false
```

3. В папке `/home/qp` создать подпапку `WP.API` и скопировать туда всё содержимое каталога `bin/release/publish/`, в который осуществлялась публикация. Следует проверить, что у пользователя *QP* есть права на чтение и исполнение содержимого папки `WP.API`.

При всех вариантах установки:

1. Перейти в папку `WP.API`. В файле `appsettings.json` в секции `QpSettings` изменить значение параметра `ConnectionString` на строку подключения к ранее развёрнутой базе демо-сайта в параметре (можно скопировать из конфигурационного файла QP).
2. В файле `NLog.config` найти `internalLogFile` и `<variable name="logDirectory" value=` и прописать там путь `/var/log/wp-api/`.

Внимание: для `internalLogFile` должен сохранится конечный файл, т.е. правильно строка будет выглядеть так: `/var/log/wp-api/internal-nlog.txt`

3. В файле `NLog.config` в секции `rules` во всех логгерах, в которых в параметре `writeTo` задано значение `console`, заменить его на `fileStructured`.
4. Создать на сервере папку `/var/log/wp-api/` и выдать пользователю *QP* права на владение папкой.
5. Создать файл `wp-api.service` в папке `/usr/lib/systemd/system/` и заполнить его следующим содержимым:

```
[Unit]
Description=WP API
After=qp.service
```

```
StartLimitIntervalSec=0


[Service]
Type=simple
Restart=on-failure
RestartSec=5
User=qp
WorkingDirectory=/home/qp/WP.API/
ExecStart=/bin/dotnet QA.WidgetPlatform.Api.dll --urls http://*:6200


[Install]
WantedBy=multi-user.target
```

Внимание: если в конкретном дистрибутиве Linux отсутствует папка `/usr/lib/systemd/system/`, то все изменения следует вносить в папку `/lib/systemd/system/`.

6. Прописать сервис в автозапуск, выполнив команду:

```
systemctl enable wp-api.service
```

7. Запустить сервис, выполнив команду:

```
systemctl start wp-api.service
```

7.2. Установка WP.DemoSiteRus.React.Module

При сборке из исходников:

1. Вытянуть исходники из репозитория на github.
2. В папке `demosite-rus-module` выполнить установку *npm*-пакетов командой `npm ci`.
3. Собрать модуль в подпапку `dist`, выполнив команду:

```
npm run build:server:prod
```

4. В папке `/home/qp` создать подпапку `WP.DemoSiteRus.React.Module` и скопировать туда всё содержимое каталога `dist`, в который осуществлялась публикация. Следует проверить, что у пользователя *QP* есть права на чтение и исполнение содержимого папки `WP.DemoSiteRus.React.Module`.

При всех вариантах установки:

1. Установить компонент `http-server` командой:

```
npm install --global http-server
```

2. Создать файл `wp-demosite-rus-react-module.service` в папке `/usr/lib/systemd/system/` и заполнить его следующим содержимым:

```
[Unit]
```

```
Description=QP DemositeRus.React.Module
StartLimitIntervalSec=0

[Service]
Type=simple
Restart=on-failure
RestartSec=5
User=qp
Environment=NODE_ENV=production
WorkingDirectory=/home/qp/WP.DemositeRus.React.Module/
ExecStart=http-server –d false –p 6500

[Install]
WantedBy=multi-user.target
```

Внимание: если в конкретном дистрибутиве Linux отсутствует папка `/usr/lib/systemd/system/`, то все изменения следует вносить в папку `/lib/systemd/system/`.

3. Прописать сервис в автозапуск, выполнив команду:

```
systemctl enable wp-demosite-rus-react-module.service
```

4. Запустить сервис, выполнив команду:

```
systemctl start wp-demosite-rus-react-module.service
```

7.3. Установка WP.DemoSiteRus.React.Shell

1. Вытянуть исходники из репозитория на github.
2. В папке `demosite-rus-shell` выполнить установку *npm*-пакетов командой `npm ci`.
3. Собрать модуль в подпапку `dist`, выполнив команду:

```
npm run build:server:prod
```

4. В папке `/home/qp` создать подпапку `WP.DemoSiteRus.React.Shell` и скопировать туда всё содержимое каталога `dist`, в который осуществлялась публикация. Следует проверить, что у пользователя *QP* есть права на чтение и исполнение содержимого папки `WP.DemoSiteRus.React.Shell`.

При всех вариантах установки:

1. Создать файл `wp-demosite-rus-react-shell.service` в папке `/usr/lib/systemd/system/` и заполнить его следующим содержимым:

```
[Unit]
Description=QP DemositeRus.React.Shell
After=qp-graphql.service wp-api.service wp-demosite-rus-react-shell
StartLimitIntervalSec=0

[Service]
Type=simple
Restart=on-failure
RestartSec=5
```

```
User=qp
Environment=NODE_ENV=production
WorkingDirectory=/home/qp/WP.DemositeRus.React.Shell/
ExecStart=/bin/node server/main.js

[Install]
WantedBy=multi-user.target
```

Внимание: если в конкретном дистрибутиве Linux отсутствует папка `/usr/lib/systemd/system/`, то все изменения следует вносить в папку `/lib/systemd/system/`.

2. В файле `settings.json` задать:
   - внешний URL приложения **WP.API** в параметре `widgets.apiUrl` вместо `http://localhost:6200`, если этот этот внешний URL известен на текущий момент, и если планируется, что пользователь будет открывать приложение через браузер с другого компьютера;
   - внешний URL приложения **QP.GraphQL** в параметре `widgets.graphql.apiUrl` вместо `http://localhost:6300`, если этот этот внешний URL известен на текущий момент, и если планируется, что пользователь будет открывать приложение через браузер с другого компьютера.

Внимание: URL приложения QP.GraphQL должен заканчиваться на /graphql или /graphql/stage

3. Скопировать файл `settings.json` в папку `WP.DemoSiteRus.React.Shell/server`
4. Если внешний URL приложения **WP.DemositeRus.React.Shell** известен на текущий момент, и если планируется, что пользователь будет открывать приложение через браузер с другого компьютера, то можно выполнить команду в папке `WP.DemoSiteRus.React.Shell` (предварительно заменив `demosite-rus-react-shell.test` на реальный внешний URL):

```
sed -i -e 's/http:\/\/localhost:6500/http:\/\/demosite-rus-react-
shell.test/g' /static/client/*.*
```

5. Прописать сервис в автозапуск, выполнив команду:

```
systemctl enable wp-demosite-rus-react-shell.service
```

6. Запустить сервис, выполнив команду:

```
systemctl start wp-demosite-rus-react-shell.service
```

8. Настройка nginx

Для корректной работы **WP.API** и **QP.GraphQL** необходимо решить вопрос безопасности, связанный с *CORS*-ограничениями. Для их обхода следует формировать запросы на тот же домен, на котором располагается **WP.DemoSiteRus.React**. Осуществить это можно с помощью сервера *nginx*.

8.1. Nginx с использованием docker

Если *nginx* при установке **QP8.CMS** был запущен в *docker*, то следует:

1. Выполнить команду:

```
docker-compose down
```

2. Открыть на редактирование файл `nginx.conf`, предположительно расположенный по пути `/etc/qpconfig/nginx/nginx.conf`;
3. В конфигурационный файл добавить секции из файла `/etc/widget-react-config/nginx/wp-nginx.conf`;
4. Задать свои *DNS* вместо `wp-demosite-rus-api.test`, `graphql-demosite-rus-api.test`, `wp-demosite-rus-react-shell.test` и `demosite-rus-react-module.test`;
5. Выполнить команду:

```
docker-compose up –d
```

8.2. Nginx без использования docker

Если *nginx* при установке **QP8.CMS** был запущен не в *docker*, то следует:

1. Открыть на редактирование файл `nginx.conf`, предположительно расположенный по пути `/etc/nginx/nginx.conf`;
2. В конфигурационный файл добавить секции из файла `/home/qp/widget-react-config/nginx/wp-nginx.conf`;
3. Задать свои DNS вместо `wp-demosite-rus-api.test`, `graphql-demosite-rus-api.test`, `wp-demosite-rus-react-shell.test` и `demosite-rus-react-module.test`;
4. Проверить корректность конфигурации командой:

```
nginx -t
```

5. Если всё корректно, то прочитать обновленную конфигурацию *nginx* командой:

```
nginx -s reload
```