ООО «КВАНТУМ АРТ»

115184, Москва, Озерковский переулок, д. 12

тел. (495) 783-65-74

Программный продукт «QP8.WidgetPlatform»

Руководство администратора

Москва
2024

Назначение документа

Настоящий документ содержит руководство администратора по программному продукту «QP8.WidgetPlatform». Цель документа – предоставить администратору сведения о продукте, достаточные для его установки и эксплуатации.

Целевая аудитория

Документ предназначен для администраторов, обладающих следующими компетенциями:

- администрирование операционных систем семейств Microsoft Windows и Microsoft Windows Server,
- администрирование СУБД PostgreSQL/Postgres Pro или Microsoft SQL Server,
- администрирование систем для управления данными,
- знание веб-технологий (HTTP, DNS).

История изменений

Версия	Дата	Автор	Описание
0.9.1	06.06.2024	Селю П.Н.	Добавлено описание настройки доступа к MinIO
0.9.0	13.06.2023	Селю П.Н.	Добавлено описание установки под Linux
0.8.3	26.08.2020	Грицай С.И.	Базовая версия документа

Оглавление

1. Термины и определения

1.1. Общие термины и определения

Термин или определение	Описание
Информационная Система (далее **«Система»**)	Автоматизированный программно-аппаратный комплекс, предназначенный для хранения, обработки и выдачи данных
«QP8.CMS» или **«QP8.CMS с поддержкой PostgreSQL»** (далее **«QP»**)	Программный продукт, обладающий широким спектром возможностей для разработки программной части Систем различной сложности
Модульное приложение (также **«Приложение»**, **«Виджет»**)	Обладающий ГПИ инструмент, содержащий набор функциональных возможностей для взаимодействия пользователей с какой-либо Системой (текущей или сторонней)
QP8.WidgetPlatform (также **«Виджетная платформа»**)	Продукт, расширяющий 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»).
Инструмент	Часть Системы, обладающая определёнными функциональными возможностями
Development-окружение	Среда, в которой осуществляется разработка и отладка Систем
Stage-окружение	Среда, максимально приближенная к production-окружению, в которой персоналом организации-разработчика осуществляется тестирование Систем
Production-окружение	Среда, используемая для размещения Систем, готовых к эксплуатации неограниченным кругом пользователей
Графический пользовательский интерфейс (далее **«ГПИ»**)	Метод взаимодействия пользователя с Системой, при котором все ключевые способы управления Системой выполнены с использованием различных графических элементов
Обработчик	Программное средство, используемое на серверной части Системы для обработки запросов пользователей к веб-сайту Системы
Active Directory (далее **«AD»**)	Служба каталогов для операционных систем Microsoft Windows Server. Базируется на протоколе LDAP
Пагинация	Нумерация страниц
Entity Framework (далее **«EF»**)	Технология для доступа к данным с использованием объектно-реляционного сопоставления (ORM, от англ. «Object-Relational Mapping»)
NuGet	Средство для управления пакетами, используемое при разработке программных продуктов на платформе Microsoft.
MinIO	Открытый сервер хранилища объектов, совместимый с сервисом облачных хранилищ S3.

1.2. Термины и определения для QP

Термин или определение	Описание
DNS	Доменное имя, используемое в Системе для работы с веб-сайтом

Бэкенд	Копия QP. Бэкенд обладает ГПИ для работы с содержимым БД Системы
Виртуальный путь	URI до объекта
Код клиента (Customer code)	Уникальный параметр, определяющий БД Системы, с которой взаимодействует бэкенд QP
Контент	Раздел сайта
Поле	Атрибут контента. С использованием полей формируется структура данных для контента
Пользовательское действие	Дополнительная функциональная возможность для бэкенда, добавленная Разработчиком в Систему
Реплейс	Уникальное кодовое имя для статьи, с использованием которого можно вызвать содержимое этой статьи в других статьях текущего сайта
Сайт	Набор данных в бэкенде. Допускается создание нескольких сайтов. Содержимое каждого сайта определяется созданными в нём контентами
Статья	Элемент контента. Статья содержит данные, заданные в поля контента
Физический путь	Путь до объекта в файловой системе

1.3. Определения для ролей пользователей в Системе

Роль	Определение
Пользователь	Персона, осуществляющая взаимодействие с Системой посредством интерфейсов, предоставляемых Системой
Администратор	Пользователь с правами на внесение любых изменений в Систему, которые можно выполнить с использованием бэкенда QP либо отдельной административной панели управления
Контент-менеджер	Пользователь с ограниченными правами на изменение содержимого Системы с использованием бэкенда QP либо отдельной административной панели управления
Разработчик	Пользователь с правами на внесение любых изменений в Систему (в том числе в содержимое скриптов, структуру БД)

1.4. Аббревиатуры

Аббревиатура	Значение
БД	База данных
СУБД	Система управления базами данных
AD	Active Directory
ID (Identifier)	Идентификатор объекта
HTML (HyperText Markup Language)	Язык разметки документов
JSON (JavaScript Object Notation)	Текстовый формат обмена данными
XML (eXtensible Markup Language)	Расширяемый язык разметки документов
PNG (Portable Network Graphics)	Растровый формат хранения для графических данных

API (Application programming interface, интерфейс программирования приложений)	Набор правил по использованию функциональных возможностей Системы, предоставляемый разработчикам для организации взаимодействия сторонних программных продуктов с Системой
LINQ (Language-Integrated Query)	Компонент .NET Framework для работы с данными из БД как с объектами. Запросы к СУБД формируются с использованием языков программирования .NET
LINQ to SQL	Решение для LINQ, позволяющее в качестве источника данных использовать СУБД Microsoft SQL Server
DPC (Digital Product Catalog)	Программный продукт, разработанный на базе QP; ориентирован на удобство работы со структурой данных для различных продуктов Заказчика.
POCO (Plain old CLR object)	Подход к разработке программного обеспечения, предполагающий использование максимально простых классов для работы с объектами

2. 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.

3. Общие сведения о продукте

3.1. Получение дистрибутива продукта

Продукт «QP8.WidgetPlatform» доступен по адресу http://downloads.quantumart.ru/Widgets. Продукт QP8.WidgetPlatform состоит из:

- административного модуля управления структурой сайта **WP.Admin**;
- API для режима OnScreen **WP.OnScreen**;
- набора библиотек QA.DotNetCore.Engine.*, доступных для скачивания из nuget-репозитория;
- демо-сайта **WP.DemositeRus.**

В рамках скачивания продукта доступны:

- дистрибутив QP8.WidgetPlatform для установки на Windows;
- архив с бинарными файлами виджетной платформы и демо-сайта для Linux-платформ;
- бекап базы данных демосайта и скрипт базы данных current.sql;
- архив с манифестами для docker-compose и k8s и образцом конфигурации для nginx – для виджетной платформы;
- архив с медиаресурсами для демосайта.

3.2. Системные требования

3.2.1. Аппаратное обеспечение

Параметр	**Минимальная конфигурация**	**Рекомендуемая конфигурация**
Процессор	Intel Pentium IV 1.8 ГГц	Intel Xeon 2.4 ГГц x2
Память	2 ГБ	8 ГБ
Дисковое пространство	2 ГБ	100 ГБ и больше (в зависимости от применения)

3.2.2. Программное обеспечение

Операционная система

Окружение	**Описание**
Production	Требуется серверная ОС: • Microsoft Windows Server 2012 R2 или выше; • Linux (в.т.ч. Ubuntu, Debian, CentOS, RedOS, AstraLinux).
Stage	Достаточно ОС для настольных ПК: • Microsoft Windows 10 ver. 1809 или выше. Рекомендуется ОС из списка для production-окружения.
Development	

Поддерживаются 64-битные версии ОС.

Для рабочего места Разработчика достаточно любой из перечисленных ОС для настольных ПК.

СУБД

Поддерживается любая из списка:

- Microsoft SQL Server 2012 или выше,
- PostgreSQL (или Postgres Pro) 12 или выше с установленным contrib.

ПО для установки на Linux (с использованием Docker)

- docker
- docker-compose

ПО для установки на Linux (без использования Docker)

- Для работы приложения:
  - aspnetcore-runtime-6.0 (см инструкцию),
  - nginx.
- Для сборки из исходников:
  - .NET Core SDK 6.0.408 и выше,
  - NodeJs 16.

ПО для установки на Linux (Kubernetes)

- Кластер Kubernetes с версией компонентов 1.20 или выше
- kubectl 1.20 или выше

ПО для установки на Windows

- IIS 8.5
- Powershell 5.1 или выше с модулем WebAdministration (для IIS)
- Для установки на SQL Server:
  - PS-модуль SqlServer или SqlPs
- Для установки на PostgreSQL:
  - Postgres CLI (psql, pg_restore)
- ASP.NET Core Runtime 6.0.16 (Hosting Bundle)

Примечание: Для работы Postgres CLI Может потребоваться дополнительная установка компонента (https://support.microsoft.com/ru-ru/help/2977003/the-latest-supported-visual-c-downloads), если он не был установлен в системе ранее.

Программное обеспечение для работы с ГПИ продукта

Работа с ГПИ ведётся с использованием веб-браузера. Поддерживаемые веб-браузеры:

- Google Chrome (или веб-браузер на основе Chromium),
- Microsoft Internet Explorer (не ниже 11.0),
- Microsoft Edge,
- Mozilla Firefox.

Примечание: рекомендуется использовать актуальную версию веб-браузера.

ПО для разработчика

- Необходимые SDK и фреймворки:
  - .NET Core SDK 6.0.408 и выше,
  - NodeJs 16.
- Поддерживаемые IDE:
  - Microsoft Visual Studio 2022 (v.17.4 или выше),
  - Rider 2022.1 или выше.

3.3. Документация по продукту

Пакет документов по продукту содержит:

Название	Описание
Руководство администратора	Установка, обновление, удаление продукта и эксплуатация Системы на основе продукта
Руководство разработчика	Разработка Системы на основе продукта
Руководство редактора	Работа с данными в Системе, созданной на основе продукта, в роли Контент-менеджера

3.4. Предварительные требования

Требуется актуальная версия QP8.CMS, детали по установке доступны здесь.

4. Архитектурная схема

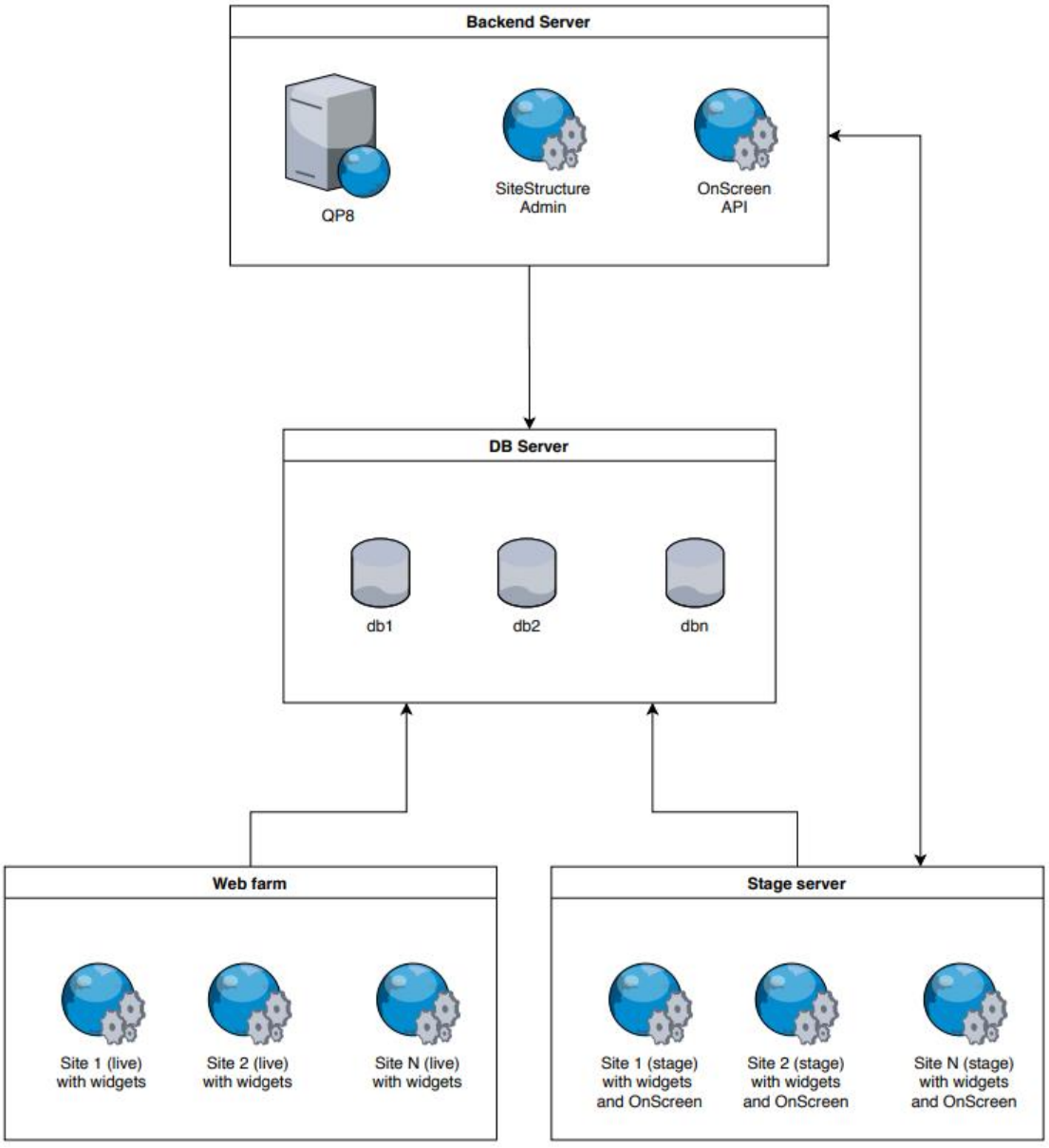

Рисунок 1. Общая схема архитектуры Системы

4.1. Backend Server

- QP8 – базовая конфигурация QP
- SiteStructure Admin – административный модуль управления структурой сайта и виджетами, который разворачивается на одном сервере с QP8 и подключается в виде пользовательского действия. Вносимые изменения сохраняются в БД.
- OnScreen API – веб-приложение, предоставляющее API для реализации функциональности OnScreen в Stage-режиме

4.2. DB Server

Сервер БД. Один экземпляр SiteStructure Admin и OnScreen API может работать с несколькими БД.

4.3. Web farm

Live-версии сайтов, построенных на QP8 с использованием виджетной платформы. Обычно развертываются в виде веб-фермы. Используют подключение к БД, сервер статики (не показан на схеме). Прямая связь с QP8 или административным модулем структуры сайта отсутствует.

4.4. Stage server

Stage-версии сайтов, построенных на QP8 с использованием виджетной платформы. Обычно развертываются на отдельном stage-сервере. Используют подключение к БД, сервер статики (не показан на схеме). На Stage-версии может быть настроен режим OnScreen, который взаимодействует с OnScreen API. Прямая связь с QP8 или административным модулем структуры сайта отсутствует.

5. Установка продукта на Windows

5.1. Необходимые права

- Права локального администратора (для копирования файлов на локальном хосте, настройки IIS, создания и запуска Windows служб).
- В версии для SQL Server:
  - права копирования файлов на сетевую шару SQL Server;
  - права администратора на SQL Server (для восстановления базы из бэкапа и запуска скриптов);
  - если не задаются имя пользователя и пароль для SQL Server, предполагается наличие Windows-аутентификации, настроенной для пользователя, запускающего скрипт инсталляции.
- В версии для PostgreSQL:
  - права администратора на PostgreSQL (для восстановления базы из бэкапа и запуска скриптов).

5.2. Автоматическая установка

1. Скачать и распаковать дистрибутив (перед распаковкой необходимо разблокировать архив (см. рис. 1), чтобы в дальнейшем извлекаемые из него файлы были также разблокированы):

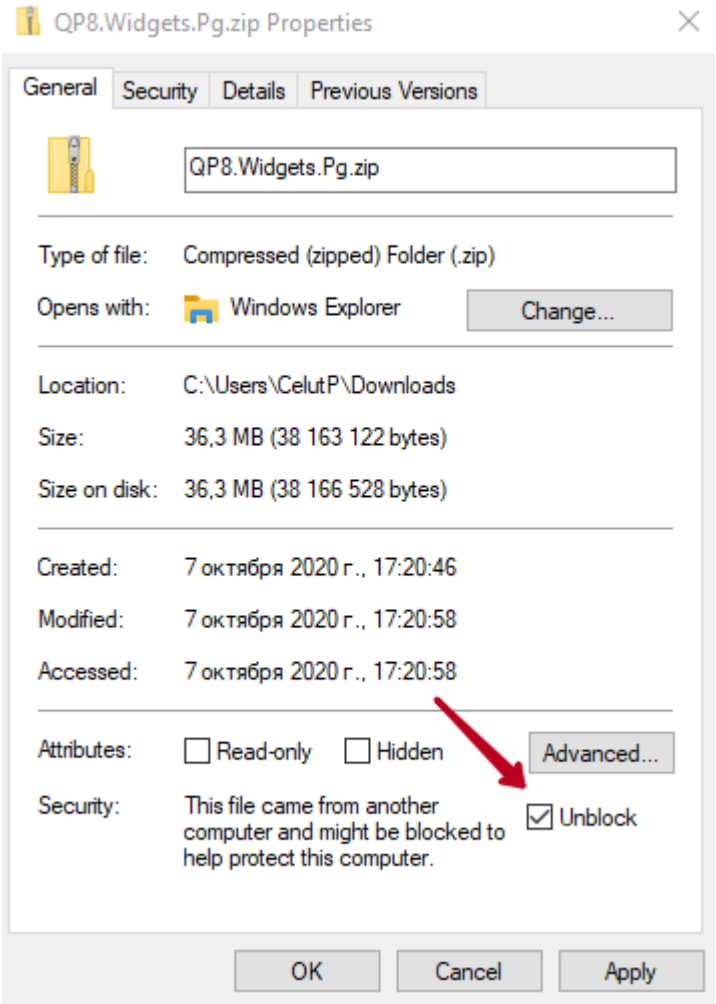

Рисунок 1. Разблокировка архива с дистрибутивом продукта

2. Запустить Install.bat (для SQL Server) или install_pg.bat (для PostgreSQL) от имени Администратора.
3. Откроется консоль, в которой будут запрошены параметры:
    - `CustomerCode` - уникальное наименование клиента/проекта;
    - `Db server` - имя хоста SQL Server или PostgreSQL, где будет установлена база каталога;
    - `Db login` - логин администратора (только для PostgreSQL);
    - `Db password` - пароль администратора (только для PostgreSQL).

Примечание: Параметр CustomerCode также используется как имя создаваемой базы данных. При повторном запуске нужно указать новый customer code, если требуется сохранить текущие изменения в базе данных, иначе она будет пересоздана в процессе инсталляции

4. На время установки откроется консоль PowerShell. Все действия установки сохранятся в файл Install.log в папке C:\QA\Logs. Проверить файл логов на наличие ошибок. Если были ошибки, то устранить их причину и повторно запустить Install.bat.
5. Зайти в бекенд каталога по URL http://<хост установки>:<порт установки QP>, для авторизации выбрать установленный customer code сайта, логин и пароль по умолчанию – admin/admin.

Примечание: В качестве имени хоста вместо localhost нужно использовать название компьютера, которое можно узнать, например, через команду hostname, иначе не будут сохраняться cookies и часть функциональности будет недоступна.

Примечание: порт установки QP по умолчанию – 89. В иных случаях актуальное значение можно посмотреть в IIS Manager.

5.3. Ручная установка

Служит альтернативой Install.bat (install_pg.bat). Позволяет использовать дополнительные параметры для управления процессом установки.

1. Запустить консоль PowerShell от имени администратора.
2. Перейти в каталог `.\Install\.`
3. Команда `Get-Help .\Install.ps1 –detailed` вернет подробное описание скрипта установки, его параметров и примеры использования.
4. Запустить `.\Install.ps1` с нужными параметрами.

Скрипт установки работает по следующему сценарию:

1. Проводится валидация параметров на существующий путь к файлу/каталогу:
    - `sourceBackupPath`;
    - `currentSqlPath`.
2. Проверка окружения:
    - установлен ли .NET Core Runtime нужной версии;
    - доступен ли SQL Server или PostgreSQL по указанному хосту.
3. Опционально (в зависимости от параметра `cleanUp`):
    - удаляются все компоненты продукта;
    - удаляются их файлы;
    - удаляется customer code каталога из QP.

Таким образом, при повторном запуске, можно обновлять продукт, делать новый биндинг портов и т.п.

4. Проверяется доступность портов, передаваемых в параметрах:
   - `administrationPort;`
   - `onScreenPort;`
   - `demositeStagePort;`
   - `demositeLivePort.`

Они будут зарезервированы за компонентами продукта и должны быть доступны на момент установки.

5. Устанавливаются компоненты продукта:
   - QA.Engine.Administration: Административный модуль управления структурой сайта;
   - QA.Engine.OnScreenAdmin: API для режима OnScreen;
   - QA.Engine.DemoSite.Live: Live-версия демо-сайта;
   - QA.Engine.DemoSite.Stage: Stage-версия демо-сайта.
6. Создается новый customer code для демо-сайта.
7. Копируется бэкап на сервер БД (только для SQL Server).
8. Восстанавливается база данных из бэкапа.
9. База данных обновляется до актуального состояния.
10. В базе обновляются настройки, зависящие от параметров скрипта.
11. Регистрируется в QP customer code демо-сайта.

6. Установка продукта на Linux

6.1. Установка БД на Linux

1. Скачать БД демосайта для QP8.WidgetPlatform, а также пример БД для QP8.CMS (для получения актуального скрипта current.sql) и распаковать.
2. Создать новую пустую БД в PostgreSQL, например, с именем *demosite_rus.*
3. Восстановить БД с помощью утилиты `pg_restore` из файла дампа, распакованного на шаге 1, в пустую БД *demosite_rus*, созданную на шаге 2. Пример команды (необходимо использовать реальные логин, пароль и имя сервера вместо *dbuser*, *dbpass* и *dbserver*, вместо DUMP_PATH указывается путь к распакованному файлу дампа):

```
pg_restore -Fc -d 'postgresql://dbuser:dbpass@dbserver/demosite_rus' -j 4 'DUMP_PATH'
```

Замечание: рекомендуется использовать pg_restore версии 12.*, можно использовать версию старше.

4. Выполнить файл `current.sql`, распакованный на шаге 1, на БД *demosite_rus,* предварительно переключив текущий сеанс на схему *qp* с помощью команды:

```
SET search_path TO qp;
```

5. Создать пользователя, например, с именем *demo* и дать ему права на базу данных. Список прав:

```
GRANT CONNECT,TEMP ON DATABASE demosite_rus TO demo;
GRANT USAGE,CREATE ON SCHEMA qp TO demo;
GRANT SELECT,INSERT,UPDATE,DELETE,REFERENCES,TRIGGER ON ALL TABLES IN SCHEMA qp TO demo;
GRANT USAGE,SELECT,UPDATE ON ALL SEQUENCES IN SCHEMA qp TO demo;
GRANT EXECUTE ON ALL ROUTINES IN SCHEMA qp TO demo;
ALTER ROLE demo SET search_path TO qp,public;
ALTER DEFAULT PRIVILEGES IN SCHEMA qp GRANT SELECT,INSERT,UPDATE,DELETE,REFERENCES,TRIGGER ON TABLES TO demo;
ALTER DEFAULT PRIVILEGES IN SCHEMA qp GRANT USAGE,SELECT,UPDATE ON SEQUENCES TO demo;
ALTER DEFAULT PRIVILEGES IN SCHEMA qp GRANT EXECUTE ON ROUTINES TO demo;
call qp_change_contents_ownership('demo')
```

Замечание: если в дальнейшем под созданным пользователем не будет выполняться обновление БД при переходе на новую версию, а вместо этого будет использована административная учётная запись, то права REFERENCES и TRIGGER для таблиц и UPDATE для последовательностей можно не задавать.

6. Если необходимо использовать разных пользователей БД для QP и для сайта, то вот скорректированный набор прав для пользователя сайта (пользователь *demosite*):

```
GRANT CONNECT,TEMP ON DATABASE demosite_rus TO demosite;
GRANT USAGE ON SCHEMA qp TO demosite;
GRANT SELECT,INSERT,UPDATE,DELETE ON ALL TABLES IN SCHEMA qp TO demosite;
GRANT USAGE,SELECT ON ALL SEQUENCES IN SCHEMA qp TO demosite;
GRANT EXECUTE ON ALL ROUTINES IN SCHEMA qp TO demosite;
```

```
ALTER ROLE demosite SET search_path TO qp,public;
ALTER DEFAULT PRIVILEGES IN SCHEMA qp GRANT SELECT,INSERT,UPDATE,DELETE ON TABLES TO demosite;
ALTER DEFAULT PRIVILEGES IN SCHEMA qp GRANT USAGE,SELECT ON SEQUENCES TO demosite;
ALTER DEFAULT PRIVILEGES IN SCHEMA qp GRANT EXECUTE ON ROUTINES TO demosite;
```

7. Добавить в конфигурационный файл QP `config.xml` новую запись `customer`, указав актуальную строку подключения к БД *demosite_rus*. В качестве образца можно использовать уже существующую запись `customer`. При этом если необходимо запускать приложение под стандартным пользователем *postgres,* нужно добавить в строку подключения дополнительный параметр для поиска схемы `;Search Path=qp,public;`
8. Настроить установленную базу на ранее развёрнутый QP.Storage, выполнив скрипт (*demosite_rus* можно заменить на желаемое имя папки, остальные настройки соответствуют тому, что было развёрнуто по умолчанию):

```
update site set upload_dir = '/qplibrary/demosite_rus/', upload_url = '/demosite_rus/',
upload_url_prefix = 'http://localhost:5000'
```

Внимание: при использовании для QP.Storage внешнего DNS, который настраивался в рамках установки QP, необходимо использовать этот внешний DNS вместо `localhost:5000`. Вариант с localhost подходит только если все сервисы разворачиваются на одном компьютере и пользователь будет открывать приложение через браузер также на этом компьютере.

9. Скачать файл с медиа-ресурсами, распаковать его в папку `/qplibrary/demosite_rus` (если ранее было выбрано другое имя вместо *demosite_rus*, то использовать его). Проверить, что у пользователя `QP` есть права на модификацию содержимого папки.

Внимание. Если настройки в БД не будут соответствовать физическому расположению файлов, то медиа-ресурсы не будут отображаться на сайте и будет невозможно управлять ими через интерфейс QP.

6.2. Установка продукта на Linux (с использованием Docker)

6.2.1. Базовая последовательность действий

1. Скачать архив widget-config.tar и распаковать его содержимое в `/etc/widget-config`.
2. Перейти в папку `/etc/widget-config/compose`.
3. В файле docker-compose.yml задать внешний URL приложения WP.OnScreen в параметре `OnScreen__AdminSiteBaseUrl` вместо `http://localhost:6000`, если этот этот внешний URL известен на текущий момент, и если планируется, что пользователь будет открывать приложение через браузер с другого компьютера.
4. В файле docker-compose.yml задать строку подключения к ранее развёрнутой базе демосайта в параметре `ConnectionStrings__DatabaseQPPostgre` (можно скопировать из конфигурационного файла QP).
5. Выполнить команду: `sudo docker-compose up -d`.
6. В результате запустятся три контейнера:
   1) `qp-admin-wp` - веб-приложение управления виджетной платформой (интегрируется в QP);
   2) `qp-onscreen-wp` - подсистема, предоставляющее API для реализации функциональности OnScreen в Stage-режиме;
   3) `qp-demosite-rus-wp` - сайт, предназначенный для демонстрации функциональности виджетной платформы и OnScreen.

При необходимости внесения изменений в базовую конфигурацию в следующем разделе представлено подробное описание манифеста `docker-compose.yaml`.

6.2.2. Описание манифеста

В каждом сервисе присутствуют одни и те же секции конфигурации, а именно:

- `container_name` - понятное имя контейнера, по нему потом можно будет обращаться к контейнеру, к его логам и так далее, рекомендуем использовать его, куда проще запомнить, чем каждый раз меняемые идентификаторы;
- `image` - образ контейнера (далее для каждого контейнера будет отдельное описание этого пункта);
- `restart` - настройка перезапуска, по умолчанию стоит "всегда", но можно отключить автоматический перезапуск при падениях контейнера и других сценариях. Подробнее в документации к docker;
- `ports` - маппинг портов хоста на внутренний порт контейнера, задаётся в формате `host_port:app_in_container_port`, где:
  - `host_port` - порт сервера на котором поднимается контейнер,
  - `app_in_container_port` - порт который использует приложение внутри контейнера для своей работы.

Внимание: app_in_container_port менять запрещено, в случае изменения работа ПО не гарантируется! host_port для разных контейнеров в рамках одного хоста не должны пересекаться.

- `environment` - список параметров окружения (конфигурация приложения в контейнере) (далее для каждого контейнера будет отдельное описание этого пункта);
- `depends_on` - список контейнеров от которых зависит работа текущего контейнера. **Внимание:** depends on определяет порядок запуска. НЕ МЕНЯТЬ! В случае изменения работа ПО не гарантируется.

Настройки сервиса qp-admin-wp:

- `image` репозиторий, имя и версия docker-образа, со списком доступных образов можно ознакомиться по ссылке;
- `environment` – переменные окружения для контейнера:
  - `ConfigurationServiceUrl` – адрес сервиса конфигурации (может быть внутренним),
  - `ConfigurationServiceToken` – токен сервиса конфигурации.

Внимание: Настройки сервиса конфигурации должны быть такими же как и у основного приложения QP, которое устанавливалось в составе продукта QP8.CMS.

Настройки сервиса `qp-onscreen-wp`:

- `image` репозиторий, имя и версия docker-образа, со списком доступных образов можно ознакомиться по ссылке;
- `environment` – переменные окружения для контейнера:
  - `ConfigurationServiceUrl` – адрес сервиса конфигурации (может быть внутренним),
  - `ConfigurationServiceToken` – токен сервиса конфигурации.

Внимание: Настройки сервиса конфигурации должны быть такими же как и у основного приложения QP, которое устанавливалось в составе продукта QP8.CMS.

Настройки сервиса `qp-demosite-rus-wp:`

- `image` репозиторий, имя и версия docker-образа, со списком доступных образов можно ознакомиться по ссылке;
- `environment` – переменные окружения для контейнера:
  - `ConnectionStrings__DatabaseQPPostgre` - строка подключения к базе данных согласно стандарту, типовой пример строки подключения:
    `Server=SERVER;Database=DB-NAME;User Id=USER;Password=PASSWORD`
    где:
    SERVER — адрес сервера, где развернут Postgres, *DB-NAME* — имя базы данных, *USER* — пользователь, у которого есть права в указанной БД на чтение и запись, *PASSWORD* - пароль от пользователя;
  - `QpSettings__IsStage` - режим работы сайта (функциональность OnScreen доступна только в режиме Stage), данный параметр следует задать как `true`;
  - `QpSettings__CustomerCode` - customer code сайта (уникальное наименование клиента/проекта);
  - `OnScreen__AdminSiteBaseUrl` - URL модуля OnScreen, должен быть внешний адрес или DNS-имя, `localhost:6000` можно оставить только если пользователь будет запускать приложение из браузера с того же компьютера, где установлено приложение;
  - `NewsNotificationServiceConfig__NotificationServiceIsActive` - опция позволяющая включать и выключать сервис рассылки (для упрощения демонстрационных действий, данный параметр следует задать как `false`);

- `CaptchaSettings__IsActive` - опция позволяющая включать и выключать функционал капчи (для упрощения демонстрационных действий, данный параметр следует задать как `false`);
- `Search__BaseUrl` — опция необходимая для работы поиска (отдельный модуль, нет в комплекте данной поставки), в рамках данной инструкции следует указать `http://localhost`.

6.3. Установка продукта на Linux (с использованием Kubernetes)

1. Скачать архив widget-config.tar и распаковать его содержимое в `/etc/widget-config.`
2. Перейти в папку `/etc/widget-config/k8s.`
3. В файле *widget.yml* можно настроить:
   - внешний DNS приложения WP.OnScreen в параметре `OnScreen__AdminSiteBaseUrl` вместо `wp-onscreen.qp`;
   - строку подключения к ранее развёрнутой базе демосайта в параметре `ConnectionStrings__DatabaseQPPostgre` (можно скопировать из конфигурационного файла QP).
4. Разворачиваем сервисы командой `kubectl apply –f widget.yml.`
5. При необходимости можно настроить ингрессы к `wp-admin`, `wp-onscreen` и `wp-demosite-rus` в файле *ing.yaml*, задав свои DNS вместо `wp-admin.test`, `wp-onsreen.test` и `wp-demosite-rus.test` и выполнив команду `kubectl apply -f ing.yaml`.

6.4. Установка продукта на Linux (без использования Docker)

При установке из скомпилированного кода:

- Скачать архив widget-config.tar и распаковать его содержимое в `/home/qp/widget-config.`
- Выкачиваем архив, содержащий бинарные файлы виджетной платформы и демосайта.
- Распаковываем полученный архив, в домашний каталог пользователя, созданного в рамках установки QP8.CMS (по умолчанию - `/home/qp`). Должна получиться такая структура из трех каталогов:
  - `WP.Admin`;
  - `WP.OnScreen`;
  - `WP.DemositeRus`.

6.4.1. Установка WP.Admin

- При сборке из исходников:
  - вытягиваем исходники из репозитория на github;
  - в папке `QA.Engine.Administration.Core` выполняем установку npm-пакетов командой `npm ci`;
  - собираем фронт командой `npm run build`;
  - в папке проекта `QA.Engine.Administration.WebApp.Core` (там должен быть файл `QA.Engine.Administration.WebApp.Core.csproj`) выполняем команду на публикацию:

```
dotnet publish "QA.Engine.Administration.WebApp.Core.csproj" -c Release -o
bin/release/publish/ -r linux-x64 --self-contained=false
```

  - в папке `/home/qp` создаём подпапку `WP.Admin` и копируем туда всё содержимое каталога `bin/release/publish/`, в который осуществлялась публикация. Нужно проверить, что у пользователя `qp` есть права на чтение и исполнение содержимого папки `WP.Admin`.
- Переходим в папку `WP.Admin.`
- В файле `appsettings.json` в корневую секцию добавляем параметр `ConfigurationServiceUrl` и в качестве значения указываем адрес сервиса конфигурации QP который мы развернули ранее. Значение должно совпадать с тем, которое уже прописано в параметре `QpConfigUrl` конфигурационного файла `/home/qp/QP/appsettings.json` приложения QP.
- В файле `appsettings.json` в корневую секцию добавляем параметр `ConfigurationServiceToken` и в качестве значения указываем JWT-токен. Значение должно совпадать с тем, которое уже прописано в параметре `QpConfigToken` конфигурационного файла `/home/qp/QP/appsettings.json` приложения QP.
- В файле `NLogClient.config` находим `internalLogFile` и `<variable name="logDirectory" value=` и прописываем там путь `/var/log/wp-admin/`.

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

- В файле `NLogClient.config` в секции `rules` во всех логгерах, в которых в параметре `writeTo` задано значение `console`, заменяем его на `fileStructured`.
- Создаём на сервере директорию `/var/log/wp-admin/` и выдаём `qp` пользователю права на владение директорией.

- Создаём файл `wp-admin.service` в папке `/usr/lib/systemd/system/` и заполняем его следующим содержимым:

```
[Unit]
Description=WP Admin
After=qp.service
StartLimitIntervalSec=0

[Service]
Type=simple
Restart=on-failure
RestartSec=5
User=qp
WorkingDirectory=/home/qp/WP.Admin/
ExecStart=/bin/dotnet QA.Engine.Administration.WebApp.Core.dll --urls http://*:5500

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

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

- Прописываем сервис в автозапуск, выполнив команду:

```
systemctl enable wp-admin.service
```

- Запускаем сервис, выполнив команду:

```
systemctl start wp-admin.service
```

6.4.2. Установка WP.OnScreen

- При сборке из исходников:
  - вытягиваем исходники из репозитория на GitHub;
  - в папке *QA.Engine.OnScreenAdmin.Web* выполняем установку npm-пакетов командой `npm ci`;
  - собираем фронт командой `npm run build:prod`;
  - в папке проекта `QA.Engine.OnScreenAdmin.Web` (там должен быть файл `QA.Engine.OnScreenAdmin.Web.csproj`) выполняем команду на публикацию:

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

  - в папке `/home/qp` создаём подпапку `WP.OnScreen` и копируем туда всё содержимое каталога `bin/release/publish/`, в который осуществлялась публикация. Нужно проверить, что у пользователя `qp` есть права на чтение и исполнение содержимого папки `WP.OnScreen`.
- Переходим в папку `WP.OnScreen.`
- В файле `appsettings.json` в секцию `ConfigurationService` добавляем параметр `Url` и в качестве значения указываем адрес сервиса конфигурации QP который мы развернули

ранее. Значение должно совпадать с тем, которое уже прописано в параметре `QpConfigUrl` конфигурационного файла `/home/qp/QP/appsettings.json` приложения QP.

- В файле `appsettings.json` в секцию `ConfigurationService` добавляем параметр `Token` и в качестве значения указываем JWT-токен. Значение должно совпадать с тем, которое уже прописано в параметре `QpConfigToken` конфигурационного файла `/home/qp/QP/appsettings.json` приложения QP.
- В файле `NLog.config` находим `internalLogFile` и `<variable name="logDirectory" value=` и прописываем там путь `/var/log/wp-onscreen/`.

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

- В файле `NLog.config` в секции `rules` во всех логгерах, в которых в параметре `writeTo` задано значение `console`, заменяем его на `fileStructured`.
- Создаём на сервере директорию `/var/log/wp-onscreen/` и выдаём `qp` пользователю права на владение директорией.
- Создаём файл `wp-onscreen.service` в папке `/usr/lib/systemd/system/` и заполняем его следующим содержимым:

```
[Unit]
Description=WP OnScreen
After=qp.service
StartLimitIntervalSec=0

[Service]
Type=simple
Restart=on-failure
RestartSec=5
User=qp
WorkingDirectory=/home/qp/WP.OnScreen/
ExecStart=/bin/dotnet QA.Engine.OnScreenAdmin.Web.dll --urls http://*:6000

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

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

- Прописываем сервис в автозапуск, выполнив команду:

```
systemctl enable wp-onscreen.service
```

- Запускаем сервис, выполнив команду:

```
systemctl start wp-onscreen.service
```

6.4.3. Установка WP.DemoSiteRus

- При сборке из исходников:
  - вытягиваем исходники из репозитория на GitHub;
  - в папке проекта `Demosite` (там должен быть файл `Demosite.csproj`) выполняем команду на публикацию:

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

  - в папке `/home/qp` создаём подпапку `WP.DemoSiteRus` и копируем туда всё содержимое каталога `bin/release/publish/`, в который осуществлялась публикация. Нужно проверить, что у пользователя `qp` есть права на чтение и исполнение содержимого папки `WP.DemoSiteRus`.
- Переходим в папку `WP.DemoSiteRus.`
- В файле `appsettings.json` в секции `ConnectionStrings` задаём в значении параметра `DatabaseQPPostgre` строку подключения к ранее развёрнутой базе данных демо-сайта (см. Установка БД на Linux). Значение можно скопировать из конфигурационного файла QP `/home/qp/qpconfig/config.xml`.
- В файле `appsettings.json` в секции `OnScreen` задаём в значении параметра `AdminSiteBaseUrl` **внешний** URL приложения WP.OnScreen, развёрнутого ранее. Если все сервисы разворачиваются на одном компьютере и пользователь будет открывать приложение через браузер также на этом компьютере, то можно оставить значение `http://localhost:6000`.
- В файле `NLog.config` находим `internalLogFile` и `<variable name="defaultLogDir" value=` и прописываем там путь `/var/log/wp-demosite-rus/`.

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

- Создаём на сервере директорию `/var/log/wp-demosite-rus/` и выдаём `qp` пользователю права на владение директорией.
- Создаём файл `wp-demosite-rus.service` в папке `/usr/lib/systemd/system/` и заполняем его следующим содержимым:

```
[Unit]
Description=WP DemoSiteRus
After=qp.service
StartLimitIntervalSec=0

[Service]
Type=simple
Restart=on-failure
RestartSec=5
User=qp
WorkingDirectory=/home/qp/WP.DemoSiteRus/
ExecStart=/bin/dotnet Demosite.dll --urls http://*:6100
```

```
[Install]

WantedBy=multi-user.target
```

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

- Прописываем сервис в автозапуск, выполнив команду:

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

- Запускаем сервис, выполнив команду:

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

6.5. Настройка nginx

Для корректной работы `WP.Admin` и `WP.OnScreen` необходимо решить вопрос безопасности, связанный с CORS ограничениями. Для их обхода следует делать запросы на тот же домен, на котором располагается QP, осуществить это можно с помощью nginx-а.

Необходимые секции представлены в файле `/widget-config/nginx/wp-nginx.conf`. Детально ознакомится с настройкой nginx, а так же общей механикой – можно по этой ссылке.

6.5.1. 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-config/nginx/wp-nginx.conf`;
4) задать свои DNS вместо `wp-admin.test`, `wp-onsreen.test` и `wp-demosite-rus.test`
5) выполнить команду `docker-compose up –d`.

6.5.2. 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-config/nginx/wp-nginx.conf`;
3) задать свои DNS вместо `wp-admin.test`, `wp-onsreen.test` и `wp-demosite-rus.test`;
4) после чего проверить корректность конфигурации командой `nginx -t`;
5) если всё корректно, то перечитать конфигурацию nginx командой `nginx -s reload`.

6.6. Дополнительная настройка в интерфейсе QP

6.6.1. Правка адреса WP.Admin

В дереве веб-интерфейса QP разворачиваем `qa_demosite_rus`, `Custom Actions`, кликаем узел дерева `Manage Pages`.

В секции `Basic Parameters` в поле `URL` прописываем внешний адрес приложения **WP.Admin**. Это тот самый внешний адрес, который задавался в настройках nginx, либо в ингрессе вместо `wp-admin.test`. В случаях intranet-разворачивания можно задать значение `http://<host_name>:5500`, где *host_name* - имя комьютера. Если все сервисы разворачиваются на одном компьютере и пользователь будет открывать приложение через браузер также на этом компьютере, то можно задать значение `http://localhost:5500`.

6.6.2. Правка адреса демо-сайта с функциональностью OnScreen

В дереве веб-интерфейса QP разворачиваем `qa_demosite_rus`, `Custom Actions`, кликаем узел дерева `OnScreen stage`.

В секции `Basic Parameters` в поле `URL` прописываем внешний адрес приложения **WP.DemoSiteRus**. Это тот самый внешний адрес, который задавался в настройках nginx, либо в ингрессе вместо `wp-demosite-rus.test`. В случаях intranet-разворачивания можно задать значение `http://<host_name>:6100`, где *host_name* - имя комьютера. Если все сервисы разворачиваются на одном компьютере и пользователь будет открывать приложение через браузер также на этом компьютере, то можно задать значение `http://localhost:6100`.

Внимание. Функциональность *OnScreen* будет работать только на сайте с включенной опцией `IsStage=true`, если одновременно используются *live*- и *stage*-версии сайта, необходимо указывать адрес stage-версии.

7. Детальная информация о конфигурации продукта

7.1. Конфигурационный файл WP.Admin

Пример синтаксиса:

```
{
  "Logging": {
    "IncludeScopes": false,
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "System": "Warning"
    }
  },
  "ServiceName": "QA.Engine.Administration.WebApp.Core",
  "CustomerCodeParamName": "customerCode",
  "SiteIdParamName": "site_id",
  "BackendSidParamName": "backend_sid",
  "HostIdParamName": "hostUID",
  "IndexOrderStep": 1,
  "UseHierarchyRegionFilter": true,
  "CustomAction": {
    "Alias": "onscreen_custom_action",
    "ItemIdParamName": "pageId",
    "CultureParamName": "culture",
    "RegionParamName": "region"
  },
  "StartPageDiscriminator": "start_page"
}
```

Параметры:

Название	Тип	Описание
Logging	Объект	Раздел настроек логирования
IncludeScopes	Булевый	Использовать ли области видимости для логирования (системный параметр)
LogLevel	Объект	Уровни логирования для различных модулей
Default	Строка	Минимальный уровень логирования по умолчанию
Microsoft	Строка	Минимальный уровень логирования для модулей Microsoft
System	Строка	Минимальный уровень логирования для модулей System
ServiceName	Строка	Название сервиса для записи в лог

CustomerCodeParamName	Строка	Имя параметра для передачи customer code из QP
SiteIdParamName	Строка	Имя параметра для передачи ID сайта из QP
BackendSidParamName	Строка	Имя параметра для передачи ключа BackendSID из QP (используется для аутентификации)
HostIdParamName	Строка	Имя параметра для передачи идентификатора приложения из QP (используется для JS-интеграции)
UseHierarchyRegionFilter	Булевый	Использовать фильтрацию по регионам
CustomAction	Объект	Настройки интеграции с админкой OnScreen
Alias	Строка	Alias пользовательского действия OnScreen
ItemIdParamName	Строка	Название параметра для передачи ID элемента
CultureParamName	Строка	Название параметра для передачи культуры
RegionParamName	Строка	Название параметра для передачи региона
ConfigurationServiceUrl	Строка	ConfigurationServiceUrl – URL сервиса конфигурации, ConfigurationServiceToken - токен доступа к сервису конфигурации. Если оба параметра заданы, то для получения доступных кастомер кодов используется http-запрос к сервису конфигурации, иначе список берется из локального QP8
ConfigurationServiceToken	Строка	
S3	Объект	Настройки доступа к MinIO
Endpoint	Строка	Адрес MinIO
AccessKey	Строка	Ключ доступа к MinIO (логин)
SecretKey	Строка	Секрет доступа к MinIO (пароль)
Bucket	Строка	Имя бакета

7.1.1. Настройка доступа к MinIO

Настройка доступа к MinIO выполняется на уровне приложения в файле `appsettings.json`, в отдельной секции `S3`:

```
"S3": {
   "Endpoint": "minio:9000",
   "AccessKey": "minioLogin",
   "SecretKey": "minioPassword",
   "Bucket": "qp8"
}
```

Следующая функциональность WP.Admin теперь поддерживает MinIO:

- версионирование файловых полей при обновлении.

7.2. Конфигурационный файл WP.DemoSiteRus

Пример синтаксиса:

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Debug",
      "System": "Information",
      "Microsoft": "Information"
    }
  },
  "ConnectionStrings": {
    "DatabaseQP": "Application Name=DemoSite;Initial Catalog=__db.name__;Data
Source=__db.server__;User
ID=__db.user__;Password=__db.password__;MultipleActiveResultSets=True;",
    "DatabaseQPPostgre": "Server=__postgre.db.server__;Database=__postgre.db.name__;User
Id=__postgre.db.user__;Password=__postgre.db.password__"
  },
  "QpSettings": {
    "SiteId": 52,
    "IsStage": "__settings.isStage__",
    "DatabaseType": "__qp.databasetype__",
    "CustomerCode": "__qp.customercode__"
  },
  "OnScreen": {
    "AdminSiteBaseUrl": "__settings.onscreenUrl__"
  }
}
```

Параметры:

Название	Тип	Описание
Logging	Объект	Раздел настроек логирования
LogLevel	Объект	Уровни логирования для различных модулей
Default	Строка	Минимальный уровень логирования по умолчанию
Microsoft	Строка	Минимальный уровень логирования для модулей Microsoft
System	Строка	Минимальный уровень логирования для модулей System
ConnectionStrings	Объект	Строки подключения к БД
DatabaseQP	Строка	Строка подключения к SQL Server. Выбор осуществляется DatabaseType
DatabaseQPPostgre	Строка	Строка подключения к PostgreSQL. Выбор осуществляется DatabaseType
QpSettings	Объект	Задает настройки QP

SiteId	Число	Значение Id сайта
IsStage	Строка	Режим работы сайта
DatabaseType	Строка	Тип БД – postgres или sqlserver. Производит выбор строки подключения: DatabaseQPPostgre или DatabaseQP
CustomerCode	Строка	Кастомер-код
OnScreen	Объект	Задает параметры режима OnScreen
AdminSiteBaseUrl	Строка	URL административного модуля OnScreen

7.3. Конфигурационный файл WP.OnScreen

Пример синтаксиса:

```
{
  "Logging": {
    "IncludeScopes": false,
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "System": "Warning"
    }
  },
  "QpAuthSettings": {
    "WorkAsAdministrator": false,
    "ApplicationNameInQp": "onscreen-api"
  },
  "ConfigurationService": {
    "Url": "__configurationservice.url__",
    "Token": "__configurationservice.token__"
  }
}
```

Параметры:

Название	Тип	Описание
Logging	Объект	Раздел настроек логирования
IncludeScopes	Булевый	Использовать ли области видимости для логирования (системный параметр)
LogLevel	Объект	Уровни логирования для различных модулей
Default	Строка	Минимальный уровень логирования по умолчанию
Microsoft	Строка	Минимальный уровень логирования для модулей Microsoft
System	Строка	Минимальный уровень логирования для модулей System
QpAuthSettings	Объект	Задает настройки аутентификации

WorkAsAdministrator	Булевый	Предоставлять ли пользователю права администратора на действия в QP или использовать встроенные права
ApplicationNameInQp	Строка	Имя приложения для аутентификации в QP
ConfigurationService	Объект	Настройки сервиса конфигурации
Url	Строка	Если оба параметра заданы, то для получения доступных кастомер кодов используется http-запрос к сервису конфигурации, иначе список берется из локального QP8
Token	Строка	