Go.Data
Руководство системного администратора

4 августа 2020 г.

V0.4

“Мне нужно планировать работу, осуществлять поддержку, настройку и отладку моей копии приложения Go.Data …

… что мне нужно сделать, чтобы настроить Go.Data для корректной работы моих пользователей или для подключения его к другим системам?”

Оглавление

Приветствие

Назначение

Добро пожаловать в руководство системного администратора программного обеспечения Go.Data.

Данное руководство содержит подробную информацию для системных администраторов, которым необходимо осуществлять поддержку, настройку или отладку их пользовательской копии (копий) приложения Go.Data, и является дополнением к руководству пользователя и руководству по внедрению.

В руководстве рассматриваются такие задачи, как настройка почтового сервера SMTP для отправки сообщений при восстановлении пароля Go.Data, получение представления о файлах конфигурации и журнала программного средства, а также средств безопасности данного программного продукта и т. д.

Таким образом, данный документ предназначен для системных администраторов или других лиц, осуществляющих управление внедрением программного обеспечения Go.Data, как правило, на сервере для сообщества пользователей. Соответственно, документ предполагает знакомство с данным программным средством и не охватывает его функциональные возможности (эту информацию можно найти в руководстве пользователя).

Это руководство, скорее всего, будет менее актуально для тех, кто устанавливает автономную копию Go.Data на своем локальном компьютере, но, тем не менее, может быть полезно для отладки программы.

Поэтому нет необходимости читать данный документ «от корки до корки». Его следует использоваться как справочное руководство при решении конкретных задач, связанных с системным администрированием.

Дополнительная поддержка

При отсутствии необходимой информации в данном документе обратитесь в техподдержку Go.Data: -

Электронный адрес службы поддержки: godata@who.int

Присоединяйтесь к онлайн-сообществу: https://community-godata.who.int

Основные сведения

Go.Data может устанавливаться на компьютеры под управлением Windows, Linux и Apple Mac, и включает в себя MongoDB, платформу NodeJS и браузер Electron для обеспечения работы веб-интерфейса в среде «форм». Однако браузер Electron использовать не обязательно. Пользователь может получить доступ к программному продукту, используя другой браузер по своему выбору, введя в адресную строку на компьютере, на котором он установлен, следующий адрес: -

http://localhost:8000

Порт 8000 назначен по умолчанию и может быть изменен при установке программного средства – например, если другая служба уже использует этот порт.

Программное средство использует Прикладной программный интерфейс (API), методы которого самодокументируются с использованием инструмента Loopback API. Данная документация доступна по адресу: -

http://localhost:8000/explorer

Go.Data также предлагает приложение для смартфонов под управлением Android и iOS, которое осуществляет связь с API путем обмена данными.

Фактически, процесс обмена информацией с клиентского веб-интерфейса и смартфонов осуществляется через один и тот же API.

Установка Go.Data

Подробную информацию по установке Go.Data см. в Руководстве по развертыванию. В данном разделе рассматриваются вопросы, связанные с системным администрированием и поиском и устранением неисправностей при развертывании приложения Go.Data.

Минимальные системные требования

Минимальные системные требования для установки Go.Data: -

- Телефоны под управлением Android:
    - Системные требования: четырехъядерный процессор (2 ГГц или выше), 2 Гб оперативной памяти (Самым «слабым» устройством, протестированным в компании Clarisoft был Samsung Galaxy S5: https://www.gsmarena.com/samsung_galaxy_s5-6033.php) ОС: Android 5.0 или выше (желательно иметь более свежие версии, имеющие обновления для системы безопасности)
    - В связи с расширением базы данных необходимо иметь не менее 1 Гб свободного пространства
- Телефоны под управлением iOS:
    - Системные требования: iPhone 5s или выше (желательно иметь iPhone 6 или более новую версию, имеющую экран большего размера) ОС: iOS 9.3 или выше (желательно 10 или выше)
    - В связи с расширением базы данных необходимо иметь не менее 1 Гб свободного пространства
- Windows (64-бит)
    - ОС: 7
    - свободное пространство: 3 Гб (может потребоваться больше при необходимости хранения большого объема данных)
    - ЦП: 2 ГГц (в случае процессоров с более низкой частотой приложение может существенно замедляться)
    - Оперативная память: 8 Гб
- Linux (64-бит / 32-GOST)
    - ОС: Ubuntu 12.04, Fedora 21, Debian 8…
    - свободное пространство: 3 Гб (может потребоваться больше при необходимости хранения большого количества данных)
    - ЦП: 2 ГГц (в случае процессоров с более низкой частотой приложение может существенно замедляться)
    - Оперативная память: 4 Гб
- Mac (64-бит)
    - ОС: X 10.10

- свободное пространство: 3 Гб (может потребоваться больше при необходимости хранения большого количества данных)
- ЦП: 2 ГГц (в случае процессоров с более низкой частотой приложение может существенно замедляться)
- Оперативная память: 8 Гб

Разрешения для установки

При установке Go.Data на компьютер с ОС Windows можно выбрать автономный или серверный тип программы. При «автономном» варианте установки Go.Data устанавливается без сервисов Windows, что означает, что компоненты программного продукта (API и Database) будут работать при работе самого приложения Go.Data.

В том случае, если выбран «серверный» вариант, две службы Windows будут работать в фоне, так что даже при закрытом интерфейсе Go.Data программный продукт продолжает работать.

Однако использование служб Windows требует от устанавливающего повышения прав и, даже после установки Go.Data как «сервера», при запуске программы потребуется щелчок правой кнопкой мыши и выбор опции «Запуск от имени Администратора».

Пользователям Linux рекомендуется запустить инсталлятор при помощи команды «**sudo**».

Повышенные привилегии (sudo) требуются в большинстве случаев при изменении служб/функций на системном уровне. Пример: изменение брандмауэра, запуск службы, и т.д.

Команда sudo необходима для предотвращения возможного появления всплывающих окон системы безопасности, блокирующих или прерывающих установку программы.

Однако команду sudo не следует запускать, если вы по какой-либо причине не доверяете устанавливаемому приложению.

Если язык системы отличается от английского

Некоторые пользователи испытывали проблемы с установкой Go.Data на компьютеры, язык системы которых отличается от английского. В этом случае возникают ошибки либо в процессе установки, либо при запуске базы данных, при котором процесс просто «зависает» на неопределенное время на сообщении о запуске.

В качестве обходного приема необходимо изменить язык системы на английский и переустановить Go.Data.

Внимание: Необходимо изменить язык системы, а **НЕ** язык клавиатуры, поскольку последнее не даст необходимого эффекта. В последующих выпусках программы эта проблема будет решена.

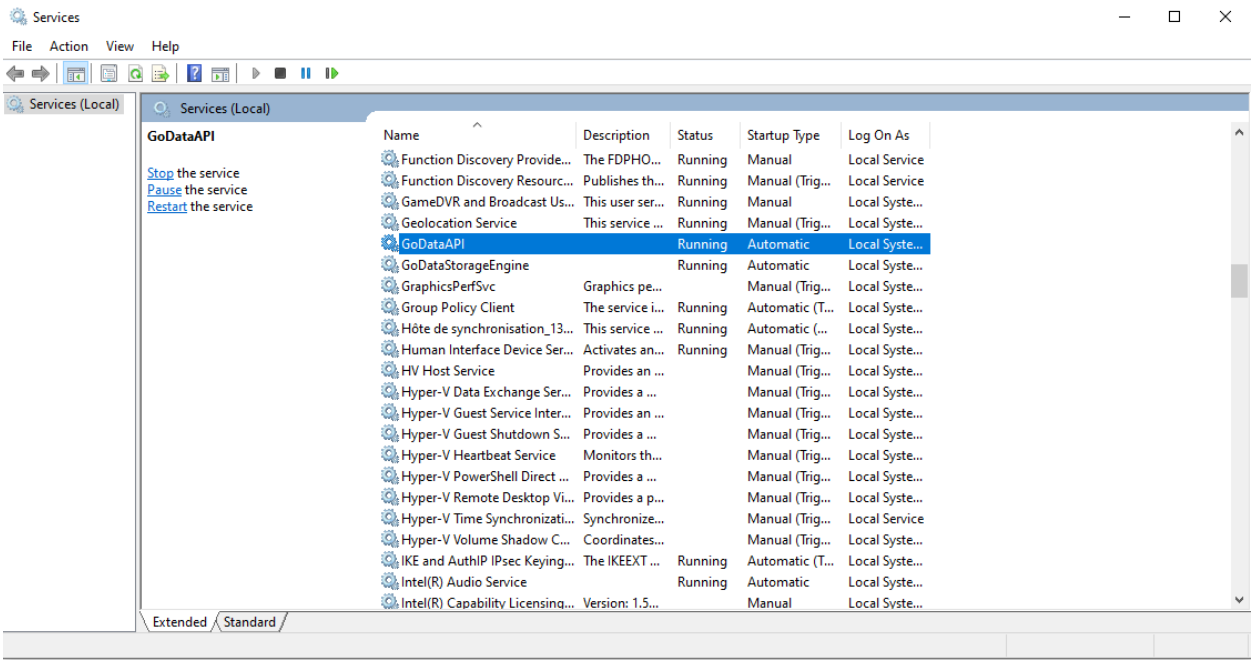
Поддержка 32-битных версий

Хотя ранние версии Go.Data были доступны как для 64-битных, так и 32-битных систем, более поздние версии Go.Data (V34 и выше) предназначены только для 64-битных

систем. Это решение было принято в связи с изначально низким спросом на установочные файлы для 32-битных систем.

Автоматический запуск Go.Data при загрузке системы

В операционной системе Windows программа Go.Data запускает свои процессы как службы Windows, которые могут запускаться автоматически при загрузке системы. Для выбора автоматического запуска программы перейдите в меню Windows, наберите «Службы» и откройте окно Служб, как показано ниже.



Две службы Go.Data носят наименования: -

- **GoDataAPI**
- **GoDataStorageEngine**

Во вкладке «Тип запуска» выберите «Автоматически». В этом окне можно изменить тип запуска, запустить, остановить, приостановить и повторно запустить любые службы.

В системе Linux существует несколько способов запуска Go.Data при загрузке системы, в зависимости от дистрибутива Linux и его версии. Подробную информацию о сценарии запуска программы как службы можно найти в руководстве к дистрибутиву Linux.

Основной способ создания служб в Linux заключается в использовании команды `systemctl`.

Создайте файл с именем godata.service в папке /etc/systemd/system.

```
cd /etc/systemd/system
touch godata.service
```

В файле godata.service пропишите следующую конфигурацию службы:

```
[Unit]
```

```
Description=GoData service
StartLimitIntervalSec=0
[Service]
Type=simple
Restart=always
RestartSec=1

# Fill here user account
User=user

# Fill here the path to Go.Data launch script
ExecStart=/home/user/Desktop/Go.Data/go-data-x64.sh
[Install]
WantedBy=multi-user.target
```

Представленная выше конфигурация может различаться в зависимости от дистрибутива Linux.
После сохранения файла запустите службу:

```
systemctl start godata
```

После успешного запуска службы настройте ее автоматический запуск при загрузке системы.

```
systemctl enable godata
```

Настройка сервера установки / обновления

Файлы установки

Хотя Go.Data при запуске проверяет наличие новой версии программы и выдает пользователю подсказку о возможности обновления, администратор может самостоятельно загрузить установочные файлы для каждой новой версии.

Сервер обновлений Go.Data будет непрерывно предоставлять новые сборки.

В будущем ВОЗ, скорее всего, сменит сервер обновлений.

Сообщения о выходе новых версий будут публиковаться на портале сообщества Go.Data по адресу https://community-godata.who.int с предоставлением соответствующих ссылок.

Структура каталога

Корневая папка сервера, используемая для загрузки новых сборок, первоначально имела следующую структуру:

```
├── x64                                    # Сборки для 64-битных архитектур
│   ├── go-data-linux-x64-{VERSION}.tar.gz     # Архив для Linux
│   ├── Go.Data Setup {VERSION}.exe            # Установщик для Windows
```

```
│   ├── Go.Data Setup {VERSION}.exe.blockmap   # Обновления для Windows
│   ├── Go.Data-{VERSION}-mac.zip              # Обновления для Mac
│   ├── Go.Data-{VERSION}.dmg                  # Установщик для Mac
│   ├── Go.Data-{VERSION}.dmg.blockmap         # Обновления для Mac
│   ├── latest-mac.yml                         # Обновления для Mac
│   └── latest.yml                             # Обновления для Windows
├── x86                                        # Сборки для 32-битных архитектур
│   ├── go-data-linux-x86-{VERSION}.tar.gz     # Архив для Linux
│   ├── Go.Data Setup {VERSION}.exe            # Установщик для Windows
│   ├── Go.Data Setup {VERSION}.exe.blockmap   # Обновления для Windows
│   ├── Go.Data-{VERSION}-mac.zip              # Обновления для Mac
│   ├── Go.Data-{VERSION}.dmg                  # Установщик для Mac
│   ├── Go.Data-{VERSION}.dmg.blockmap         # Обновления для Mac
│   ├── latest-mac.yml                         # Обновления для Mac
│   └── latest.yml                             # Обновления для Windows
└── .
```

Поскольку поддержка x86 была приостановлена начиная с версии V34, данная папка более недоступна.

Следует иметь в виду, что при выпуске новой версии файлы предыдущей сборки перемещаются в другую папку в структуре каталога или полностью удаляются. В случае, если требуется более ранняя версия Go.Data, отличная от текущей, отправьте запрос по адресу godata@who.int .

Для развертывания версии для Linux необходимо скопировать с сервера файл:

- go-data-linux-{VERSION}.tar.gz

Для развертывания версии для Windows скопируйте с сервера следующие файлы:

- Go.Data Setup {VERSION}.exe
- Go.Data Setup {VERSION}.exe.blockmap
- Latest.yml

Для развертывания версии для Mac необходимо скопировать с сервера следующие файлы:

- Go.Data-{VERSION}-mac.zip
- Go.Data-{VERSION}.dmg
- Go.Data-{VERSION}.dmg.blockmap
- latest-mac.yml

Установщик для мобильных телефонов

Последние версии приложения Go.Data для смартфонов под управлением Android и iOS доступны в Play Store и App Store, соответственно, а информацию о них можно найти на канале ВОЗ.

Обновление до последней версии Go.Data для смартфонов осуществляется так же, как и для других приложений, когда пользователь получает уведомление о возможности обновления.

Рекомендуется всегда использовать самую последнюю копию сервера и приложения Go.Data для смартфонов, чтобы воспользоваться актуальными исправлениями и улучшениями функционала приложения.

Если требуется исходный файл .apk для установки Go.Data на устройства под управлением Android, не имеющие доступа к Play Store, отправьте запрос по адресу godata@who.int .

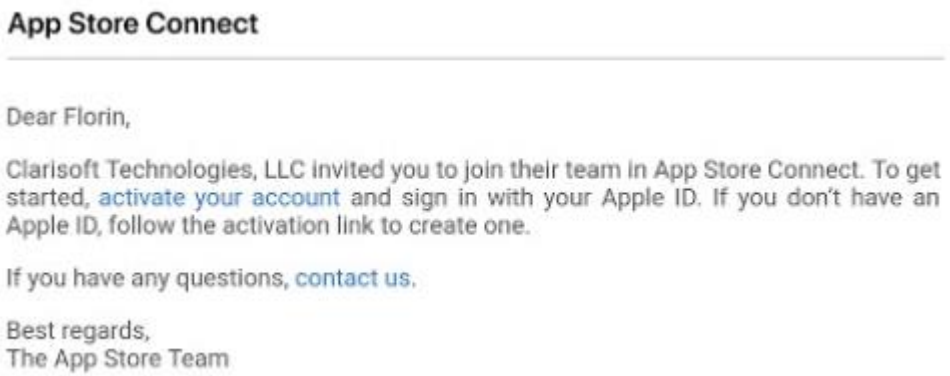
Тестирование сборок для мобильных телефонов

Если требуется установить предварительную (бета) сборку мобильного приложения Go.Data (версию, которая пока отсутствует в магазине приложений для мобильных телефонов), используйте следующую процедуру установки на iPhone и устройства под управлением Android: -

iOS

Для тестирования бета-версии Go.Data на устройстве от Apple должно использоваться приложение TestFlight. Вам необходимо будет получить приглашение от разработчика и иметь устройство, на котором будет выполняться тестирование.

Если у вас еще нет учетной записи AppleID, вам будет предложено завести ее при получении приглашения.



App Store Connect

Dear Florin,

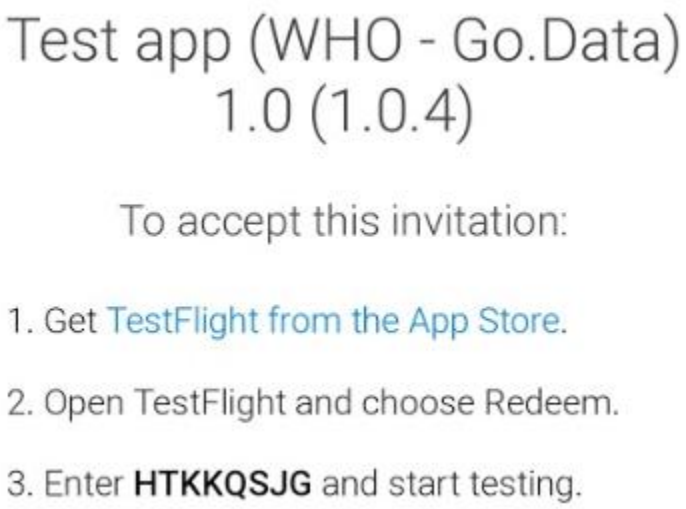
Clarisoft Technologies, LLC invited you to join their team in App Store Connect. To get started, activate your account and sign in with your Apple ID. If you don't have an Apple ID, follow the activation link to create one.

If you have any questions, contact us.

Best regards,
The App Store Team

После создания учетной записи/принятия приглашения разработчик должен будет пригласить пользователя протестировать приложение. После этого пользователь получит электронное письмо от TestFlight с инструкциями по установке приложения. Вам

необходимо нажать на кнопку *View in TestFlight* в письме. После этого вы получите пошаговые указания по установке TestFlight и самого приложения.



Необходимая платформа:

Тестирование может осуществляться на устройствах iPhone, iPad или iPod touch под управлением iOS 8 и более поздних версий.

Приложение TestFlight:

Когда разработчик приглашает тестировщиков, они получают электронное письмо с приглашением и предложением установить бесплатное приложение TestFlight из App Store на свой iPhone, iPad, iPod touch или Apple TV, если оно еще не установлено.

Отказ от тестирования:

Если вы не принимаете приглашение, бета-версия приложения не будет установлена и вас не внесут в список тестировщиков. Кроме того, вы можете отписаться от приглашений, используя ссылку в нижней части письма с приглашением, и разработчик удалит вас из списка. Если вы приняли приглашение, но не хотите больше тестировать приложение, вы можете удалить свои данные на странице данных о приложении в TestFlight.

Установка:

Для установки бета-версии приложения из электронного письма на ваше устройство под управлением iOS:

1. Откройте письмо с приглашением.
2. Нажмите Start Testing (Начать тестирование) в TestFlight.
3. Нажмите Accept (Принять), Install (Установить) или Update (Обновить).

Если на вашем устройстве уже установлена рабочая версия приложения, она будет заменена на бета-версию. Загруженная бета-версия имеет оранжевую точку рядом с наименованием, позволяющую отличить ее от других версий.

Тестирование нескольких сборок:

Хотя при просмотре приложений в TestFlight по умолчанию показывается последняя доступная сборка, у вас есть возможность протестировать другие доступные сборки.

Не рекомендуется устанавливать более ранние версии Go.Data, поскольку более новые версии могут содержать несовместимые изменения.

1. В приложении TestFlight перейдите на страницу с данными о приложении.
2. Нажмите на Previous Builds (Предыдущие сборки).
3. Выберите и установите сборку, которую хотите протестировать. Выбранная вами сборка заменит установленную.

Android

Для тестирования версии приложения для Android вы получите файл .apk. Android Package Kit (краткое наименование - apk) представляет собой формат установочного файла, используемый ОС Android для распространения и установки мобильных приложений.

Разрешение для установки файлов APK:

1. Откройте Settings (Настройки) системы Android
2. Откройте вкладку *Apps and notifications (Приложения и уведомления).* На Samsung Galaxy вместо этого нужно открыть *Biometrics and security (Биометрия и безопасность).*
3. Выберите *Install unknown apps (Устанавливать неизвестные приложения)*. Эта опция может также называться *Install other apps (Устанавливать приложения из других источников)*. На некоторых устройствах может сначала понадобиться выбрать *Special access (Специальный доступ)*.
    a. В системе Android 8.0 Oreo процедура несколько отличается. Вместо выбора установки из неизвестных источников в общих настройках вы получите подсказку, советующую разрешить браузеру или файловому менеджеру устанавливать файлы APK по первому требованию.

Установка:

1. Вы можете либо открыть файл на телефоне из электронного письма, полученного от разработчика, либо загрузить файл на компьютер и переместить его на телефон, используя кабель USB.
2. После загрузки файла на устройство используйте файловый менеджер, чтобы перейти в папку, где был сохранен файл.
3. Запустите файл и установите приложение.

Резервное копирование и восстановление

Параметры

Go.Data обладает собственным функционалом для резервного копирования и восстановления, который подробно описан в руководстве пользователя. При резервном копировании Go.Data создает ZIP-файл, содержащий описания своих данных в формате json, которые могут быть использованы для аварийного восстановления/сброса параметров системы используя определенную точку восстановления.

При резервном копировании создается полная резервная копия базы данных, а если вам требуется только подмножество данных, хранящихся в системе, следует использовать пакеты синхронизации, которые также описаны в руководстве пользователя.

Расположение файлов резервного копирования не настраивается. Они имеют определенный путь в каталоге установки Go.Data. Поэтому для полного восстановления после сбоев необходимо настроить запланированное копирование файлов резервной копии, создаваемых Go.Data на другом компьютере или настроить каталог для участия в облачном резервном копировании при помощи соответствующего программного средства, такого как OneDrive.

Файлы резервной копии защищены паролем, который хранится в файле config.json, как показано ниже в данном документе: -

```
"backUp": {
  "password": "x$^56fwm137s1a#2@oalxbnk0z541lsd"
},
```

Администраторам также может понадобиться создать копии файлов config.json и datasources.json при резервном копировании.

Резервное копирование для Windows

Особые требования к резервному копированию в Windows отсутствуют, вы можете использовать встроенные функции Go.Data, описанные в предыдущем разделе.

Методы из Go.Data API также могут быть использованы для сохранения данных. Для их вызова используйте стороннит утилиты или тригеры Go.Data.

Резервное копирование для Linux

Для ручного резервного копирования данных Go.Data на устройстве под Linux: -

1. Войдите как пользователь на веб-портал Go.Data
2. Нажмите Menu (Меню)->System Configuration (Системная конфигурация)->Backups (Резервное копирование)
3. Нажмите Quick actions (Быстрые действия)->Create Backup (Создать резервную копию)

4. Примите параметры по умолчанию или измените, при необходимости
5. Нажмите Create Backup (Создать резервную копию)

Для восстановления данных Go.Data на устройстве под Linux: -

1. Войдите в Linux
2. Запустите терминал
3. Перейдите в каталог с установленной Go.Data
4. Выполните команду <installdir>\go-data-restore-backup-x64.sh --file <backupfile>

Конфигурирование Go.Data

При установке Go.Data создается ряд конфигурационных файлов, которые находятся в каталоге установки и могут использоваться для изменения различных системных настроек.

Для вступления в силу изменений в конфигурационном файле необходимо перезапустить Go.Data.

Расположение

Основные конфигурационные файлы, используемые Go.Data, включают: -

{your\installation\directory}\bin\resources\go-data\build\server\config.json
{your\installation\directory}\bin\resources\go-data\build\server\datasources.json.

Поэтому при установке по умолчанию в Windows, например, расположение конфигурационных файлов будет следующим: -

C:\Go.Data\bin\resources\go-data\build\server

В этих двух файлах содержится ряд свойств, используемых компонентом Loopback, для предоставления API и частных свойств, относящихся к самой программе Go.Data.

В системе Linux те же файлы имеют расположение {your\installation\directory}\go-data\build\server\

Файл config.json

Файл config.json содержит свойства, необходимые для Loopback и ряд пользовательских свойств, относящихся к самой программе Go.Data.

В данном документе рассматриваются только частные свойства, поскольку базовая конфигурация описана в документации по Loopback:
https://loopback.io/doc/en/lb3/config.json.html

JSON Fragment	Пояснение
"restApiRoot": "/api",	Хотя restApiRoot, host и port являются свойствами, определяемыми Loopback, они не должны изменяться, поскольку они также требуются для работы других служб (например, restApiRoot, являющийся алгоритмом Loopback API, требуется также для работы веб-клиента).
"public": { "host": "localhost", "protocol": "http", "port": "8000" },	Указывает конечную точку, в которой, как ожидается, должно работать клиентское (веб) приложение Go.Data. Данная информация используется для проверки при запуске того, что Go.Data присутствует по этому URL, а также для создания ссылки в электронном письме для восстановления пароля. • public.host - веб-домен / IP • public.protocol - веб-протокол • public.port - веб-порт

"passwordReset": { "ttl": 900, "path": "/auth/reset-password", "from": "no-reply@who.int" },	Свойства passwordReset используются при конфигурировании электронного письма для восстановления пароля. Для отправки этого письма Go.Data требуется доступ к серверу SMTP (программа не имеет собственных функциональных возможностей SMTP) и данные этого сервера должны быть указаны в файле datasources.json. • passwordReset.ttl - время жизни маркера восстановления пароля (через какое время восстановленный пароль, ссылка на который дана в электронном письме, становится недействительным) - по умолчанию это время составляет 15 минут • passwordReset.path - путь к конечной точке API для восстановления пароля. Для восстановления пароля используется электронное письмо с соответствующей ссылкой. Данная ссылка имеет следующий вид: `${config.public.protocol}://${config.public.host}:${config.public.port}${config.passwordReset.path}` (поэтому это поле должно оставаться без изменений) • passwordReset.subject - тема письма для восстановления пароля • passwordReset.from - адрес 'От' в письме для восстановления пароля

"logging": {
"level": "info",
"maxSize": 10000000,
"maxFiles": 10,
"requestResponse": {
"trim": true,
"maxLength": 1024
},
"trim": true,
"maxLength": 1024
},
	Данный фрагмент используется для конфигурирования логгера API.
	
	• logging.level - определяет тип ошибок, которые будут регистрироваться (допустимые типы - error / info / debug)
	• logging.maxSize - максимальный размер файла журнала в байтах
	• logging.maxFiles - максимальное количество файлов журнала до запуска кругового алгоритма
	• logging.requestResponse.trim - определяет, следует ли обрезать ответ API, который будет включен в сведения об ошибке
	• logging.requestResponse.maxLength - максимальная длина ответа перед обрезкой
	• logging.trim - определяет, следует ли обрезать сообщение об ошибке
	• logging.maxLength - максимальная длина сообщения об ошибке перед обрезкой (например, исключения могут содержать информацию трассировки стека, которая может быть очень длинной)

"mapsApi": { "enabled": true, "clientId": "xyphXQJjQwLVB8rY", "clientSecret": "8dea73811df04e4e9709f15b2177b1f9", "tokenExpirationInMinutes": "20160", "tokenUrl": "https://www.arcgis.com/sharing/oauth2/token", "geocodeServerUrl": "http://geocode.arcgis.com/arcgis/rest/ services/World/GeocodeServer" },	Свойства, используемые для определения широты и долготы местоположения адреса и отображения картографических данных.

"sync": {
"asyncActionsSettings": {
"intervalTimeout": 10000,
"actionTimeout": 3600000
},
"actionCleanupInterval": 24,
"encrypt": false,
"debug": false
},
	Настройки, используемые для синхронизации данных между серверами Go.Data.

"pushNotifications": { "serverURL": "http://whocd.clarisoft.com:1337/api", "appId": "b61f5946-1af3-4e07-9986-9ffd1e36ae93", "masterKey": "KlYddh2OdVycHuVBhXv2" },	Конфигурация сервера синтаксического анализа, используемого для удаленной очистки данных с мобильных устройств (в случае кражи устройства и т. д.)

"backUp": { "password": "x$^56fwm173s1a#2@oaxlnbk0z541lsd" },	Используется для шифрования файлов синхронизации и резервного копирования (может быть удален, если шифрование файлов не требуется).

"defaultArcGisServers": [
{
"name": "WHO Background",
"url": "MAP1"
},
{
"name": "WHO Reference",
"url": "MAP2"
}
],
	Массив слоев карты, используемых по умолчанию для каждой вспышки заболевания, например, если вы оставляете вторую вкладку экрана «Outbreak» (Вспышка) пустой при создании новой вспышки, то используются эти картографические серверы.
	
	Эта информация используется только по умолчанию и всегда может быть перезаписана при создании / обновлении вспышки в программном средстве.
	
	Эти серверы используются в качестве фоновых слоев для отображения карт (например, страница случая заболевания / перемещения контакта, страница подсчета случаев и т. д.)

"cors": { "enabled": false, "whitelist": [] },	Go.Data содержит функции блокировки совместного использования ресурсов между разными источниками, чтобы защитить программу от потенциального вредоносного использования. Если CORS включен, то в заголовке «Origin» (Источник) будут приниматься только домены из белого списка и запросы без заголовка Origin (между серверами, с мобильных устройств). Это свойство отличается от свойства cors remoting (межпроцессное взаимодействие) (которое всегда должно быть false). • cors.enabled: включить/отключить CORS • cors.whitelist: «белый» список доменов
"signoutUsersOnRestart": false,	Это свойство определяет поведение в отношении маркеров аутентификации и то, будут ли они признаны недействительными при перезапуске службы Go.Data. Если установлено значение false, перезапуск сервера Go.Data не выводит из системы ни одного активного сеанса, и их маркеры остаются действительными. Если установлено значение true, при перезапуске сервера Go.Data маркеры уничтожаются и пользователям нужно будет снова войти в систему. По умолчанию после обновления приложения этот флаг сбрасывается до значения по умолчанию, поэтому, если он был установлен, потребуется снова установить его после завершения процесса обновления.
"authToken": { "ttl": 600 },	Маркер аутентификации «Time To Live» (Время жизни) в секундах. Указывает срок действия маркера, если не были выданы другие запросы. Рекомендуется значение не менее 300 секунд.
"session": { "appSId": "GoData", "secret": "5d854dbc-5d5b-11ea-bc55-0242ac130003" },	Свойство, используемое для настройки переменных сеанса для API> / • session.appSId : "GoData" - идентификатор cookie, используемый переменными сеанса, • session.secret : "5asdasdbc-5ddf-11ea-bc55-0242ac130003" => секретный ключ, используемый для генерирования переменных сеанса
"login": { "maxRetries": 10, "resetTime": 30, "resetTimeUnit": "minutes" },	Ограничения для конечных точек маркеров входа в систему и авторизации, разрешающие только настроенное количество повторных попыток входа в систему в течение заданного периода времени. (Примечание: если пользователь успешно восстанавливает свой пароль, запрет снимается.) • login.maxRetries: 10, (количество попыток хода в систему после которого пользователь получает запрет на определенное время) • resetTime: 1, (количество часов/минут/секунд действия запрета) • resetTimeUnit: «minutes» (минуты) (единица времени для свойства «resetTime») (допустимые значения: год, месяц, неделя, день, час, минута, секунда, миллисекунда)
"removeSyncSnapshotsAfter": 24, "removeTmpUploadedFilesAfter": 24, "removeTmpUploadedImportFilesAfter": 24,	removeSyncSnapshotsAfter: количество (по умолчанию: 24, количество часов до удаления файла моментального снимка - этот zip-файл используется при синхронизации между мобильным телефоном и API) removeTmpUploadedFilesAfter: количество (по умолчанию: 24, количество часов до удаления временного файла или каталога - эти временные файлы создаются API для выполнения различных действий)

	emoveTmpUploadedImportFilesAfter: количество (по умолчанию: 24, количество часов до удаления импортированного файла - эти временные файлы создаются при загрузке файла для импорта его в систему)
"captcha": { "login": false, "forgotPassword": false, "resetPasswordQuestions": false },	Настройки поведения CAPTCHA (CAPTCHA доступен в Go.Data начиная с версии 34.5): • captcha.login => boolean (по умолчанию - false) ○ true - включить captcha для экрана входа в систему ○ false - отключить captcha для экрана входа в систему • captcha.forgotPassword => boolean (по умолчанию - false) ○ true - включить captcha для экрана забытого пароля ○ false - отключить captcha для экрана забытого пароля • captcha.resetPasswordQuestions => boolean (по умолчанию - false) ○ true - включить captcha для восстановления пароля с использованием вопросов на экране ○ false - отключить captcha для восстановления пароля с использованием вопросов на экране

Файл datasources.json

Как следует из названия, файл используется для определения различных конфигураций доступа к источникам данных (например, mysql connector, mongo connector и т. д.) Основную информацию об этом файле можно найти по ссылке: https://loopback.io/doc/en/lb3/datasources.json.html

JSON Fragment	Пояснение

"email": {
"name": "email",
"connector": "mail",
"transports": [
{
"type": "SMTP",
"host": "smtp.example.com",
"secure": true,
"port": 465,
"auth": {
"user": "user",
"pass": "password"
}
}
]
},
	Настройте SMTP-сервер, используемый для отправки электронного письма при забытом пароле Go.Data – это единственное электронное письмо, генерируемое системой.
	
	В разделе средства передачи данных указывается адрес сервера и порт для связи с SMTP, а также то, является ли соединение безопасным (secure) или нет.
	
	Если secure имеет значение true, то раздел auth (аутентификация) должен быть включен, в противном случае он должен быть удален.
	
	После обновления Go.Data следует повторно проверить этот раздел, так как он может быть восстановлен при обновлении и ненужные фрагменты JSON могут быть повторно вставлены.

Для получения более подробной информации о конфигурации SMTP см. соответствующий раздел далее в данном документе

Файл component-config.json

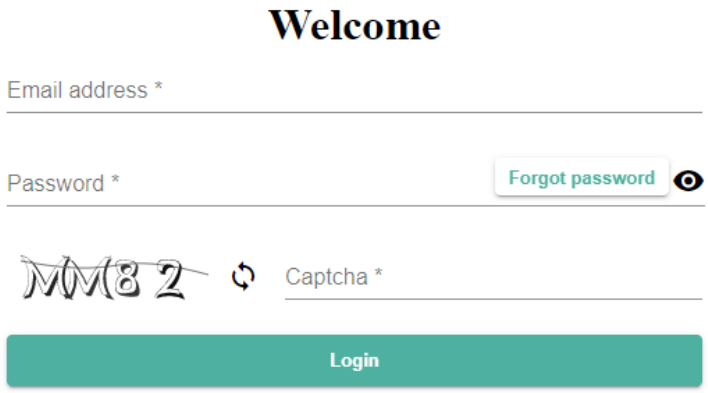
Этот файл предназначен для использования loopback, входящий с состав Go.Data, и не должен подвергаться изменению по какой-либо причине.

Файл middleware.json

Этот файл предназначен для внутренней настройки Go.Data и не должен подвергаться изменению.

Конфигурирование captcha

Тест captcha включен в Go.Data, начиная с версии V34.5, и может быть дополнительно активирован для проверки пользователя при входе в систему, запросе электронного письма для восстановления пароля и сбросе пароля.



Captcha настраивается с помощью следующего фрагмента в файле config.json, как описано в предыдущем разделе, посвященном этому файлу.

```
"captcha": {
  "login": false,
  "forgotPassword": false,
  "resetPasswordQuestions": false

},
```

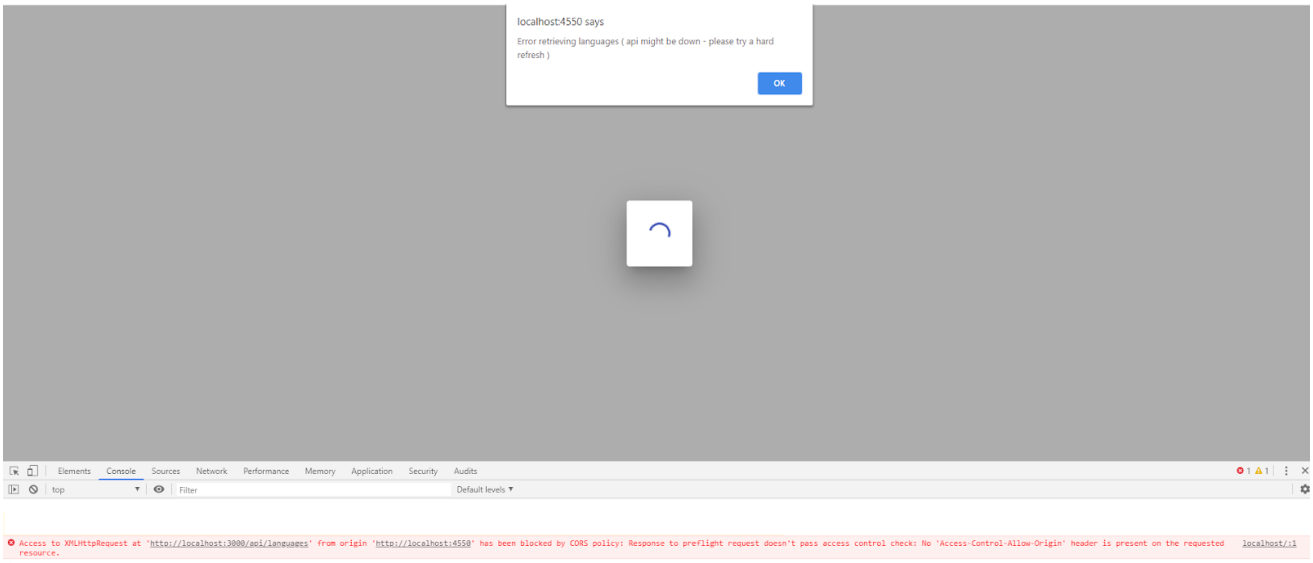
Конфигурирование CORS

Совместное использование ресурсов настраивается в файле config.json с использованием следующего фрагмента json: -

```
"cors": {
  "enabled": false,
  "whitelist": []
},
```

Это может вызвать проблемы, особенно если ваш сервер Go.Data находится за обратным прокси-сервером, с которым Go.Data может воспринимать входящие запросч как кросс-доменные. В этом случае, при попытке просмотра Go.Data, скорее всего, появится ошибка при первой попытке связи с API и получения списка языка интерфейсов для страницы входа.

Если вы видите ошибку, подобную приведенной ниже, и страница входа в систему отказывается загружаться, нужно проверить файл config.json и установить для CORS значение false или внести в «белый» список прокси-сервер и перезапустить Go.Data.



Если при открытии страницы входа Go.Data вы видите красную консоль с ошибкой: «Access to … from origin has been blocked by CORS policy...» (Доступ к... с данного домена был блокирован политикой CORS), это означает, что CORS включен, но не был правильно настроен, или веб-сайт, с которого осуществляется попытка доступа отсутствует в «белом» списке.

Начиная с версии 2.34.4 CORS отключен по умолчанию во избежание несоответствий. В более ранних версиях политика CORS была включена по умолчанию.

Для правильного конфигурирования службы необходимо изменить дополнительные настройки в файле **config.json**, расположенном в {your\installation\directory}\bin\resources\go-data\build\server

Вот список возможных параметров конфигурации:

1.1. Отключение CORS в файле **config.json**, после чего следует перезапуск API. **Важно**: необходимо обновить переключаемое свойство **в объекте «cors»**, а **не** свойство cors в remoting.

```
"cors": {
  "enabled": false,
  "whitelist": []
}
```

1,2. При активном CORS отключите перезапись конфигурации и настройте общедоступные свойства, чтобы они соответствовали способу доступа к веб-сайту (как и выше, требуется перезапуск API после изменения конфигурации)

```
"public": {
  "host": "test.host.com",
  "protocol": "http",
  "port": 80
},
"enableConfigRewrite": false,
```

1,3. При активном CORS настройте «белый» список CORS (как и выше, требуется перезапуск API)

```
"cors": {
  "enabled": true,
  "whitelist": [
        "http://www.test.com"
  ]}
```

Как общее правило, необходимо убедится, что CORS не включен или что домены, из которых поступают запросы, правильно внесены в «белый» список (если запросы делаются из браузера, который отправляет заголовки, используемые CORS).

Конфигурирование SMTP

Определите расположение файла *datasources.json*. В ОС Windows данный файл по умолчанию устанавливается в C:\Go.Data\bin\resources\go-data\build\server

Обновите данный фрагмент json, включив данные используемого SMTP сервера: -

```
"type": "SMTP",
"host": "smtp.example.com",
"secure": true,
"port": 465,
"auth": {
  "user": "user",
  "pass": "password"
}
```

Если сервер SMTP не требует аутентификации, то весь блок «auth» может быть удален.

Измените файл datasource.json в исходных файлах API (каталог сервера).

Библиотека, используемая для отправки электронных писем (nodemailer), позволяет выполнять SMTP-запросы без аутентификации путем полного удаления раздела «auth» из конфигурации «email».

См.: https://nodemailer.com/smtp/#authentication

Общие параметры

- port – порт для подключения (по умолчанию выставляется 587 если *secure* имеет значение *false* или 465, если *true*)
- host – имя хоста или IP-адрес для подключения (по умолчанию используется *'localhost'*)
- auth – определяет данные аутентификации (см. раздел authentication ниже)
- authMethod – определяет метод аутентификации, по умолчанию установлено значение 'PLAIN'

Имена хостов для поля **host** определяются с помощью `dns.resolve()`. Если вы используете неразрешимое имя хоста (например, что-то перечисленное в */etc/hosts* или вы используете другой определитель для ваших приложений Node), то укажите IP-адрес сервера SMTP как **host** и параметр **tls.servername** для фактического пользователя имени хоста. Таким образом, не предпринимается попытка определения имени хоста, но проверка TLS при этом работает.

Опции TLS

- secure – при значении *true* подключение будет использовать TLS для связи с сервером. При значении *false* (по умолчанию) TLS используется если сервер поддерживает расширение STARTTLS. В большинстве случаев необходимо установить значение *true* для соединения с портом 465. Для порта 587 или 25 значение должно быть *false*
- tls – определяет дополнительные опции node.js TLSSocket, отправляемые конструктору сокета, например, *{rejectUnauthorized: true}*.
- tls.servername - опциональное имя хоста для TLS проверки, если для host был назначен IP адрес
- ignoreTLS – если установлено значение *true*, а для параметра *secure* - false, то TLS не используется даже если сервер поддерживает расширение STARTTLS
- requireTLS – если установлено значение *true*, а для параметра *secure* - false, то Nodemailer будет пытаться использовать STARTTLS даже если сервер не заявляет о его поддержке. Если шифрованное соединение не может быть установлено, сообщение не будет отправлено

Если для параметра **secure** выбрано значение **false**, это не означает, что будет использоваться не шифрованное соединение. Большинство серверов SMTP допускают изменение подключения посредством команды [STARTTLS](https://tools.ietf.org/html/rfc3207#section-2), но для этого сперва требуется установить подключение, используя plaintext

Опции подключения

- name – опциональное имя хоста клиента, используемое для идентификации на сервере, по умолчанию используется хост-имя компьютера
- localAddress – локальный интерфейс для связи в случае сетевых подключений
- connectionTimeout – время в миллисекундах для установления соединения (по умолчанию составляет 2 минуты)
- greetingTimeout – время в миллисекундах для формирования приветствия после установления соединения (по умолчанию составляет 30 секунд)
- socketTimeout – разрешенное время отсутствия активности в миллисекундах (по умолчанию составляет 10 минут)

Опции отладки

- logger – опциональный, совместимый с bunyan. При значении true осуществляется вывод данных в консоль. Если значение не установлено, или установлено значение false, то вывод не осуществляется
- debug – при значении true регистрируется трафик SMTP, в противном случае регистрируются только события транзакций

Параметры безопасности

- disableFileAccess – при значении true файлы не могут использоваться в качестве содержимого. Эта опция используется если требуется использовать данные JSON из недоверенного источника как email. Если узел прикрепления или сообщения пытается получить что-то из файла, в ответ будет отправлено сообщение об ошибке
- disableUrlAccess – если установлено значение true, использование URL в качестве содержимого не будет разрешено

Если данные аутентификации отсутствуют, подключение будет считаться изначально прошедшим аутентификацию. В противном случае потребуется предоставить объект аутентификации.

auth - объект аутентификации:

- type указывает на тип аутентификации, по умолчанию назначается ‘login’, но может быть изменен на ‘oauth2’
- user - имя пользователя
- pass - пароль пользователя при использовании обычного имени пользователя

Конфигурирование основных карт

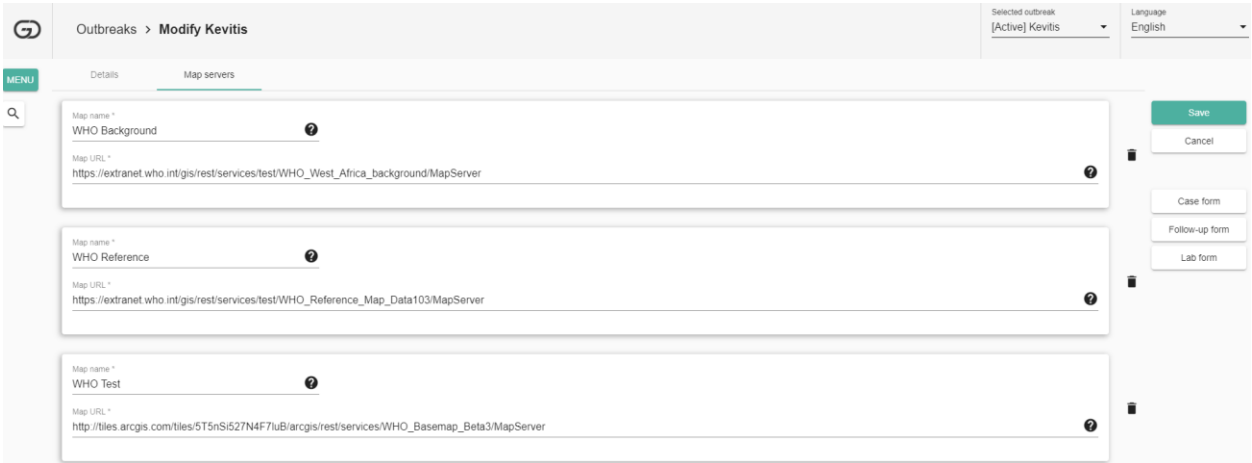
«Веб» сервер Go.Data предоставляет несколько отчетов, обеспечивающих пространственную визуализацию данных – карту подсчета случаев и карту перемещений случая/контакта. Go.Data не является полноценным программным продуктом геоинформационной службы (GIS) и предоставляет лишь основные инструменты для ключевых операций прослеживания контакта.

В приложении Go.Data для мобильных телефонов для пространственной визуализации используется собственное картографическое программное обеспечение телефона.

Карты на сервере Go.Data создаются с использованием двух элементов: -

1) Все карты являются точечными (хороплет-карты отсутствуют) и используют широту и долготу, которые выводятся либо непосредственно как часть адреса случая/контакта или извлекаются из справочных данных Go.Data о местоположении.
2) Эти точки накладываются на одну или несколько основных карт «base maps» на фоне которых отображаются данные.

Для «основных карт» в качестве этих данных может быть установлена вспышка заболевания путем передачи Go.Data адресов URL картографических серверов во второй вкладке при создании/редактировании вспышки (Outbreak).



Данные URL должны обеспечивать переход на картографический сервер, чтобы Go.Data могла считать необходимую информацию и создать фоновую карту. По этой причине

отображение карты доступно для Go.Data только в режиме онлайн. Кэширование данной информации для использования оффлайн не предусмотрено.

Если даны несколько картографических серверов, как на скриншоте выше, Go.Data будет использовать их в том порядке, в котором они предоставлены, то есть, первый на странице будет использован первым, второй будет накладываться на первый, и т.д.

При первом создании вспышки заболевания вкладка «Map servers» (Картографические серверы) (показана на скриншоте выше) будет пустой. Если ее оставить пустой, Go.Data назначит картографические серверы по умолчанию, указанные в файле *config.json* в фрагменте json: -

```
  "defaultArcGisServers": [
    {
      "name": "WHO Background",
      "url":
"http://maps.who.int/arcgis/rest/services/Basemap/WHO_West_Africa_background_7/MapServe
r"
    },
    {
      "name": "WHO Reference",
      "url": "http://maps.who.int/arcgis/rest/services/Basemap/WHO_Reference_layer/MapServer"
    }
```

После установки Go.Data предоставляет два картографических сервера ВОЗ в качестве настроек по умолчанию - основную карту ВОЗ и слой справочных данных с географическими названиями.

Список картографических серверов, которые могут использоваться Go.Data, находится по ссылке: -

https://community.microstrategy.com/s/article/KB47086-List-of-some-available-basemap-examples-to-be-used-with?language=en_US

Разделение веб-элементов и элементов базы данных

По умолчанию Go.Data устанавливает исходный код своей рабочей среды и хранилище данных mongoDB на одном компьютере. Это подходит для большинства случаев при установке на персональных компьютерах или серверах.

Однако для некоторых сценариев работы может оказаться необходимым разделить их, назначив определенные компьютеры для каждого компонента программного продукта Go.Data.

Такое разделение является предпочтительным для:

1) Работы с Go.Data при загрузке больших объемов данных из соображений быстродействия. Например, если в Go.Data загружается более 10000 случаев на сервер с минимальными характеристиками, пользователи могут заметить некоторое падение быстродействия. При таком сценарии выходом может стать раздельное размещение веб элементов и базы данных для разделения нагрузки между двумя серверами.
2) С точки зрения защиты и восстановления после сбоев, некоторые сетевые администраторы могут воспринимать размещение веб элементов и базы данных на одном компьютере как повышенный риск.

Процесс развертывания Go.Data на двух отдельных компьютерах заключается в следующем:

A. Установите Go.Data на Сервер A.
B. Установите Go.Data (ту же версию, что и на Сервере A) на Сервер B.
C. Определите порт, на котором mongoDB работает на Сервере B и IP-адрес или URL компьютера для Сервера B. Он обязательно должен быть виден с Сервера A.
D. Если используется Go.Data для Windows, перейдите в следующий каталог на Сервере A C:\Go.Data\bin\resources\go-data\build\server и найдите файл datasources.json.
E. Откройте файл datasources.json для редактирования в WordPad или другом текстовом редакторе и найдите следующий фрагмент, в котором можно указать хост, порт и url (при необходимости) для того, чтобы программа Go.Data на Сервере A могла найти свою базу данных на Сервере B.
F. Чтобы изменения вступили в силу, потребуется остановить и перезапустить службу Go.Data на Сервере A.

```
"mongoDb": {
  "host": "127.0.0.1",
  "port": 27017,
  "url": "",
  "database": "go-data",
  "password": "",
  "name": "mongoDb",
  "user": "root",
  "connector": "mongodb",
  "allowExtendedOperators": true,
  "enableGeoIndexing": true,
  "ignoreUndefined": true,
  "maxDepthOfQuery": 24,
  "lazyConnect": true,
  "authSource": "admin"
```

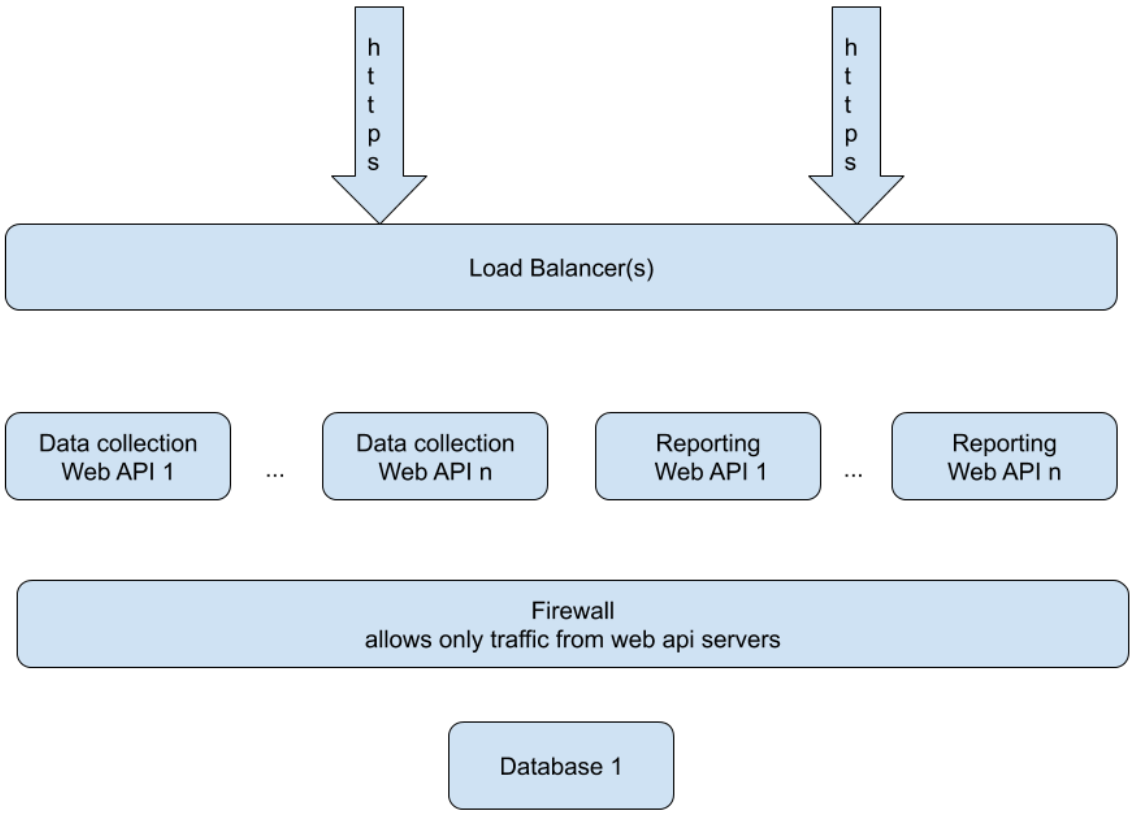
},

G. Сервер B следует настроить только на прием входящего трафика через порт базы данных с адреса Сервера A/IP-адреса, указав веб API и порт, чтобы предотвратить вероятность того, что пользователь случайно зайдет на Сервер B и попытается использовать на нем веб-приложение Go.Data. С точки зрения безопасности это наилучший подход для ограничения доступа к базе данных.

Указания выше относятся к установке новой копии Go.Data на два компьютера. Если требуется разделить базу данных для существующего развертывания Go.Data тогда процесс будет похожим, но с одним дополнительным шагом. После установки новой, пустой копии Go.Data на шаге B необходимо выполнить резервное копирование Go.Data с Сервера A и восстановить ее на Сервере B, чтобы убедиться, что база данных скопирована для совместного использования.

Конфигурирование балансировки нагрузки

Go.Data может быть развернута с балансировкой нагрузки для создания компьютерной сети, обеспечивающей лучшее быстродействие и автоматическое переключение в случае отказа. Стандартное сетевое развертывание показано на схеме на следующей странице.



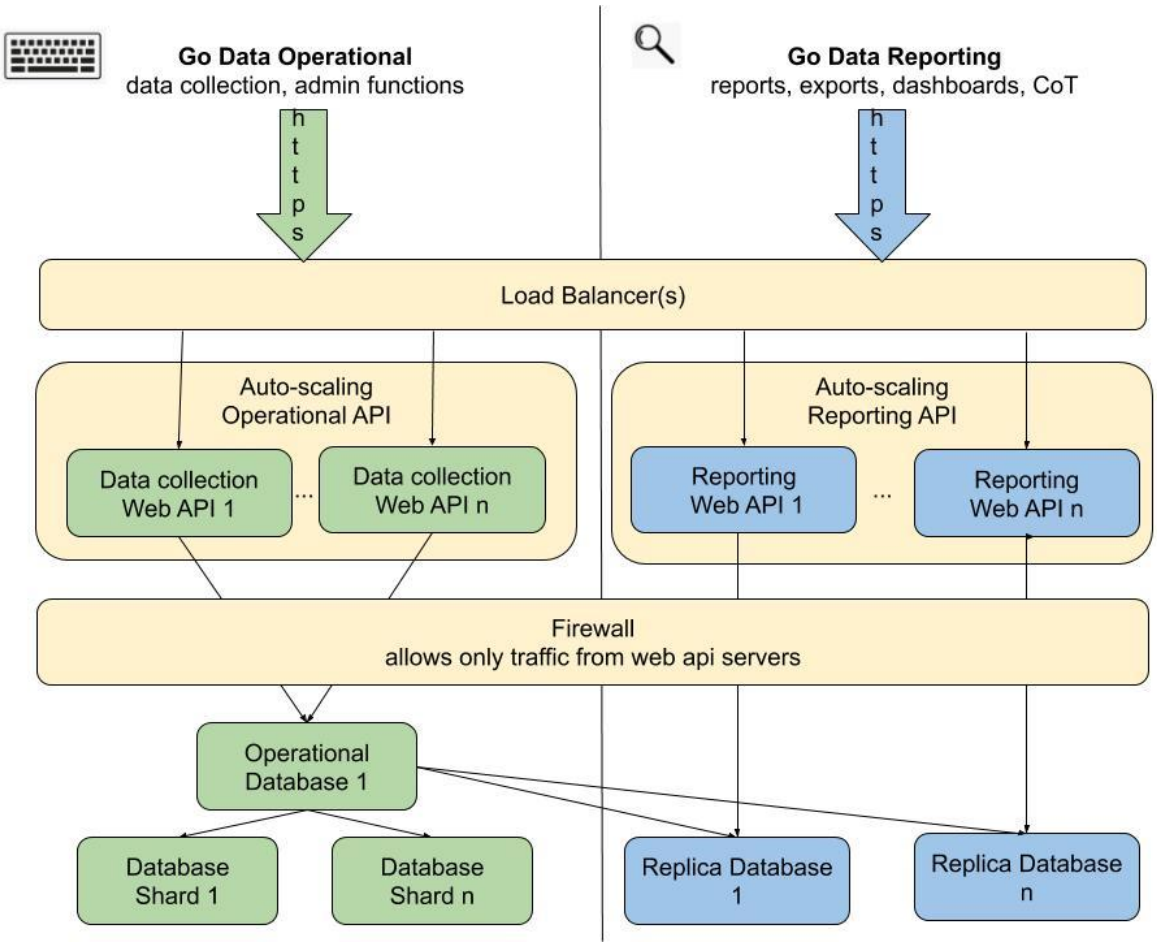
Подробности настройки:

A) Трафик принимается подсистемой балансировки нагрузки.
B) HTTPS может быть разрешен на уровне подсистемы балансировки нагрузки.
C) Подсистема балансировки нагрузки отвечает за маршрутизацию трафика на 2 сервера или более, на которых установлена Go.Data. При таком подходе не требуется привязка сессий.
D) База данных Go.Data должна быть отделена от веб-серверов, как описано в предыдущем разделе, для того, чтобы все копии использовали одну централизованную систему mongoDB.
E) Организация множества веб-серверов Go.Data может быть любой, устраивающей пользователей. Обычное рекомендуемое развертывание: -
    a. Несколько компьютеров, используемых для сбора данных (чтения и записи данных).
    b. Компьютеры, которые могут быть назначены для создания отчетов (чтения данных), таких как использование информационных панелей или экспорт данных.

F) Следует использовать межсетевой экран или другой механизм, как описано в предыдущем разделе, гарантирующий, что только авторизованные компьютеры могут взаимодействовать с базой данных Go.Data.

Если ваш хостинг поддерживает автоматическое масштабирование ресурсов, то серверы API можно настроить как часть одной и той же группы масштабирования, поэтому при повышенном использовании ЦП или памяти новые копии создаются автоматически, а при низком использовании ЦП или памяти вновь созданные копии будут уничтожены. Это обеспечит более высокую доступность, лучшее время реакции системы и эффективность затрат. Все основные поставщики облачных решений (Azure, Google, и т.д.) предлагают тот или иной вид автоматического масштабирования ресурсов.

Конфигурация для рабочей среды с отдельной средой создания отчетностей



Поскольку отчетность требует значительного использования API и базы данных, рекомендуется отделить среду создания отчетностей от рабочей среды, как показано на рисунке выше. Данная

настройка похожа на настройку с балансировкой нагрузки, описанной в главе Конфигурирование балансировки нагрузки, но имеет следующие отличия:

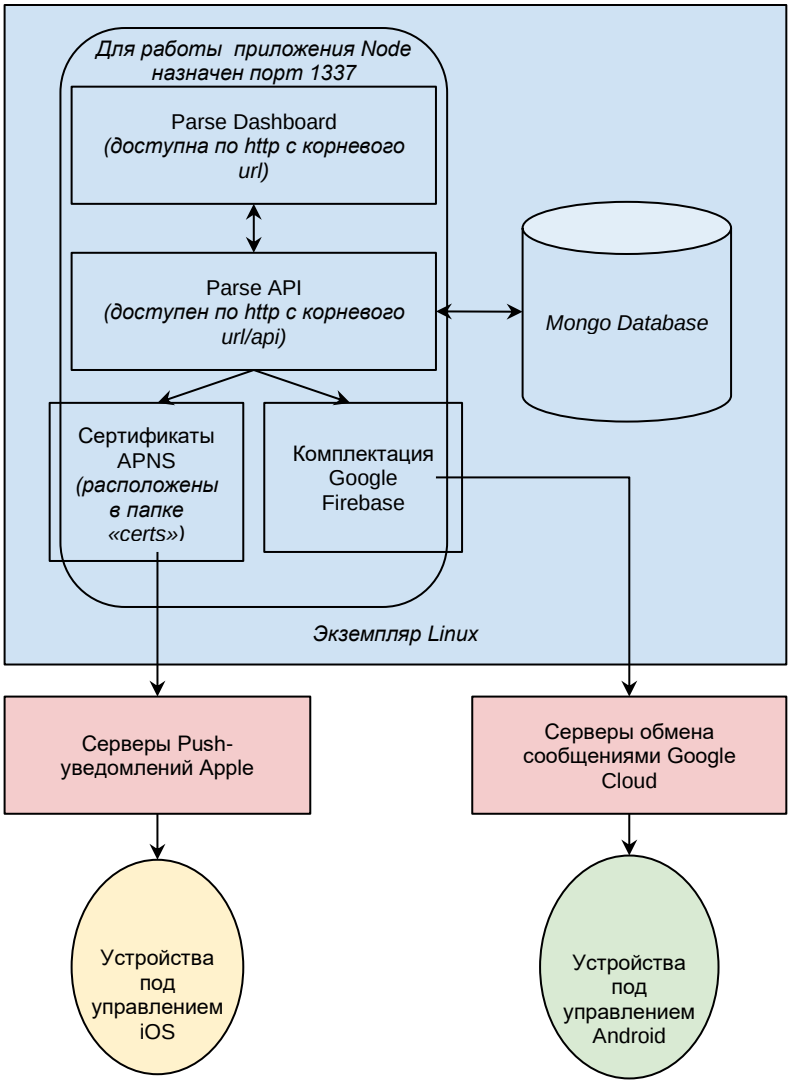
1) Среда создания отчетностей должна иметь свой собственный URL и Серверы. (Например: godata.country.gov для рабочей среды и godatareporting.country.gov для отчетности)
2) Системный администратор должен настроить одну или несколько реплик рабочей базы данных Mongo на отдельных серверах баз данных, как описано в документации Mongo.
3) Затем системный администратор должен подключить веб сервер отчетности API для указания на базы данных реплик.
4) Как рабочие сервера API, так и сервера для отчетности могут автоматически масштабироваться в зависимости от нагрузки, если это разрешено вашим сетевым или облачным провайдером.

Конфигурирование сервера parse

Сервер Parse - это выделенный сервер, используемый для отправки push-уведомлений на устройства с установленным мобильным приложением Go.Data. Push-уведомления используются для удаленной очистки устройств.

Сервер push-уведомлений

Приложение имеет следующее устройство:



Предположим, что IP данной копии Linux связан с поддоменом http://parse.who.int. Это означает, что информационная панель будет доступна по адресу http://parse.who.int:1337, а Parse API будет доступен по адресу http://parse.who.int:1337/api. Аутентификация информационной панели выполняется через пользователей из конфигурационного файла, а аутентификация API - через значения «appId» и «masterKey».

Для разрешения доступа к приложению из сети Интернет может потребоваться установить правило для межсетевого экрана на порту приложения.

Файл конфигурации

Файл конфигурации для сервера parse имеет следующий вид:

```
{
  "port": 1337,
  "database": {
    "uri": "mongodb://172.31.47.246:27017/go-data-push-notifications"
  },
  "app": {
    "serverURL": "http://whocd.clarisoft.com:1337/api",
    "appId": "b61f5946-1af3-4e07-9986-9ffd1e36ae93",
    "masterKey": "KlYddh2OdVycHuVBhXv2",
    "appName": "Go.Data"
  },
  "users": [
    {
      "user": "admin@who.int",
      "pass": "admin"
    }
  ],
  "allowHttpAccess": true,
  "push": {
    "android": {
      "senderId": "674826924884",
      "apiKey": "AIzaSyBk3ARREFLke2Srem4OGT6lGtUuZm5xSQ4"
    },
    "ios": {
      "pfx": "certs/WHO-APNS.p12",
      "passphrase": "a",
      "topic": "com.clarisoft.who",
      "bundleId": "",
      "production": false
    }
  }
}
```

- port - порт, на котором работает приложение. По умолчанию назначен порт 1337, который может быть изменен на любой порт от 1025 до 65535.
- database.url - URL базы данных. Поскольку база данных mongo работает с той же копией, что и приложение, может быть выбран URL «mongodb://127.0.0.1:port/databaseName». Порт Mongo может быть сконфигурирован при запуске службы Mongo (см. Конфигурирование Mongo), при этом может быть выбрано любое имя базы данных, приложение создаст его автоматически.
- app.serverURL. Это API приложения, с которым информационная панель приложения будет соединяться для отображения данных на портале Parse.
- app.appId - Это регулярная строка, используемая со значением «app.masterKey» для аутентификации запросов к API. Идентификатор приложения не должен быть в совместном использовании.
- app.masterKey - Это регулярная строка, используемая со значением «app.appId» для авторизации запросов к API. Главный ключ не должен быть в совместном использовании.
- app.appName - Данное значение используется только для отображения имени приложения в информационной панели Parse Dashboard.
- users - список имен и паролей пользователей, имеющих доступ к Parse Dashboard. Пожалуйста, измените учетные данные на более защищенные, чем установленные по умолчанию.
- allowHttpAccess - при задании значения «false» будет разрешен только доступ по https.
- push.android.senderId - данное значение можно найти в настройках Firebase Console (см. Настройка обмена сообщениями в Firebase Cloud).
- push.android.apiKey - данное значение можно найти в настройках Firebase Console (см. Настройка обмена сообщениями в Firebase Cloud).

- push.iOS.pfx - путь к сертификатам APNS. Срок действия данных сертификатов составляет один год, поэтому они должны обновляться ежегодно.
- push.iOS.passphrase - после создания сертификаты APNS могут быть экспортированы в формат p12 и, по выбору, для них может быть установлен пароль. Если пароль установлен, данное значение должно содержать его.
- push.iOS.topic - идентификатор пакета приложений iOS.
- push.iOS.bundleId - оставить пустым.
- push.iOS.production - установить значение «true».

MongoDB

Если система Mongo не установлена на вашем пользовательской копии или вы хотите переместить сервер Parse на новую копию, установите Mongo 3.6, используя официальную документацию. По умолчанию для работы Mongo назначен порт 27017. Вместо него может быть назначен любой свободный порт от 1025 до 65535.

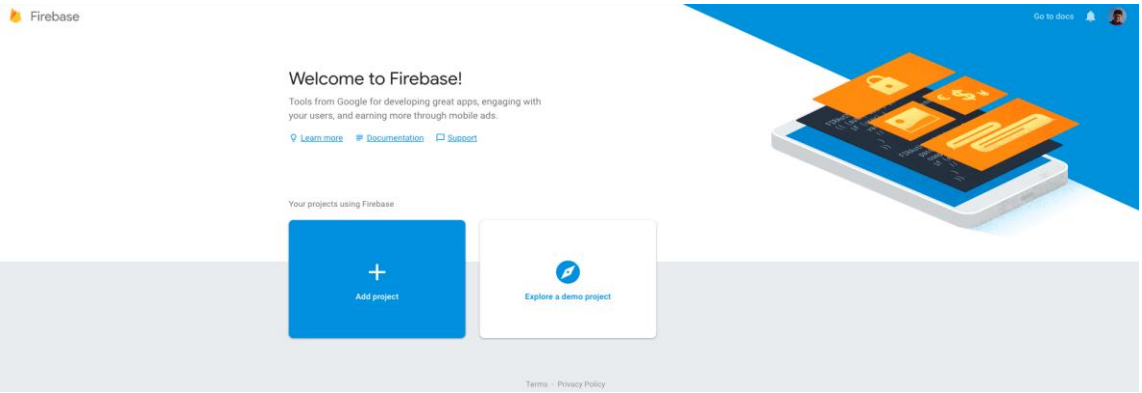
По умолчанию технология доступа к базам данных не имеет аутентификации и сервер Parse будет подключаться к базе данных без аутентификации.

В связи с этим, крайне важно предотвратить любой внешний доступ к порту базы данных с использованием правил межсетевого экрана (блокировать внешний доступ к порту 27017 или другому настроенному порту) и настройки параметра «bindIp» в Mongo.

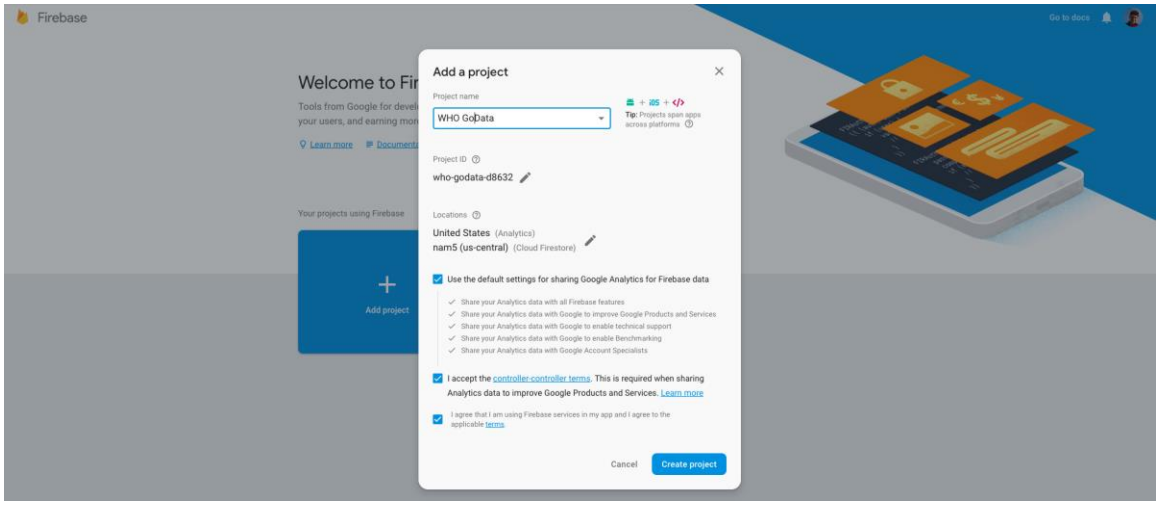
По умолчанию, начиная с Mongo 3.6, технология доступа к базам данных позволяет осуществлять соединения только с локального хоста, что необходимо для предотвращения несанкционированного доступа. Тем не менее, данную настройку необходимо проверить.

Настройка службы firebase cloud

Google Firebase является одним из компонентов настройки сервера parse, настроить который можно в Google Firebase Console. Если проект Go.Data не был сконфигурирован или при необходимости использовать другой аккаунт Google, вам потребуется создать новый проект.

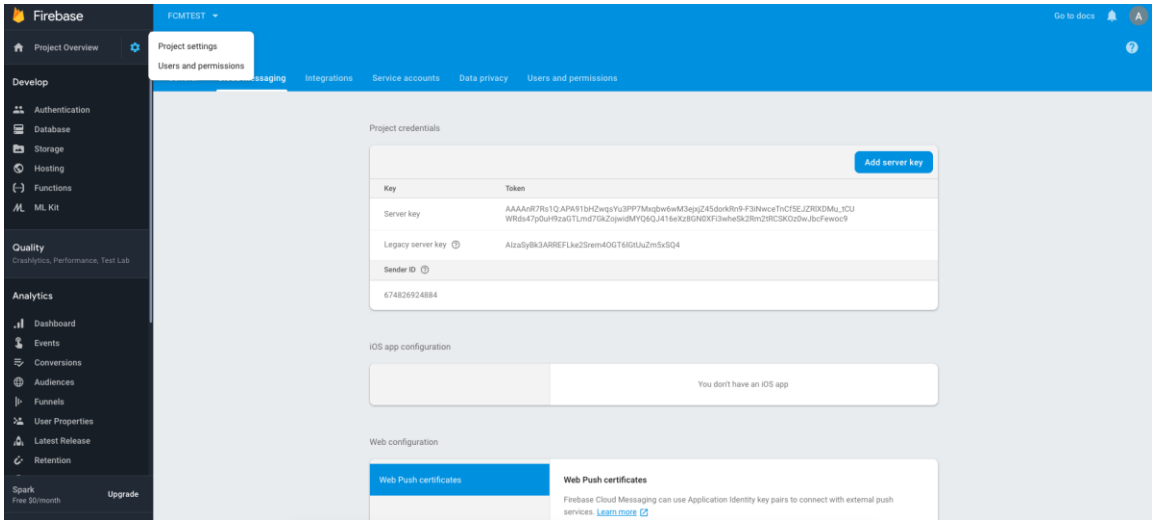


Создайте новый проект с настройками по умолчанию и примите предлагаемые условия.



Ключ API

Перейдите в Project Overview (Обзор проекта) - Project Settings (Настройки проекта) и выберите вкладку «Cloud Messaging».



Там можно найти ключ API («Legacy server key») и Sender ID, которые нужно указать в файле конфигурации Parse.

Журналы регистрации

В ОС Windows журналы регистрации Go.Data могут иметь два различных расположения, в зависимости от типа установки:

1. Если выбирается установка Go.Data только для своего пользователя:
   1.1. Журналы приложения: {your\installation\folder}\Go.Data\logs\app

Например: C:\Users\YourUser\AppData\Roaming\Go.Data\logs\app

1.2. Журналы API/Service: {your\installation\folder}\Go.Data\resources\go-data\build\logs

Например: C:\Users\YourUser\AppData\Local\Programs\Go.Data\resources\go-data\build\logs

1.3. Журналы базы данных: {your\installation\folder}\Go.Data\db

Например: C:\Users\YourUser\AppData\Roaming\Go.Data\db

2. Если выбирается установка Go.Data для всех:

2.1. Журналы приложения: {your\installation\folder}\Go.Data\data\logs\app

Например: C:\Go.Data\data\logs\app

2.2. Журналы API/Service: {your\installation\folder}\Go.Data\bin\resources\go-data\build\logs

Например: C:\Go.Data\bin\resources\go-data\build\logs

2.3. Журналы базы данных: {your\installation\folder}\Go.Data\data\logs\db

Например: C:\Go.Data\data\logs\db

Уровень протоколирования может быть настроен так же, как SMTP путем изменения файла конфигурации (config.json в каталоге сервера).

Go.Data может регистрировать в файла журнала любое действие, имеющее равный или более высокий уровень, чем тот, который установлен в файле config.json.

Может быть установлено одно из следующих значений:

- debug => log everything
- info
- error

На Linux

1. Журналы приложения (только для Windows):
{your\installation\folder}\Go.Data\data\logs\app
Например: C:\Go.Data\data\logs\app
2. Журналы API: {your\installation\folder}\Go.Data\bin\resources\go-data\build\logs
Например: C:\Go.Data\bin\resources\go-data\build\logs

3. Журналы базы данных: {your\installation\folder}\Go.Data\data\logs\db
Например: C:\Go.Data\data\logs\db

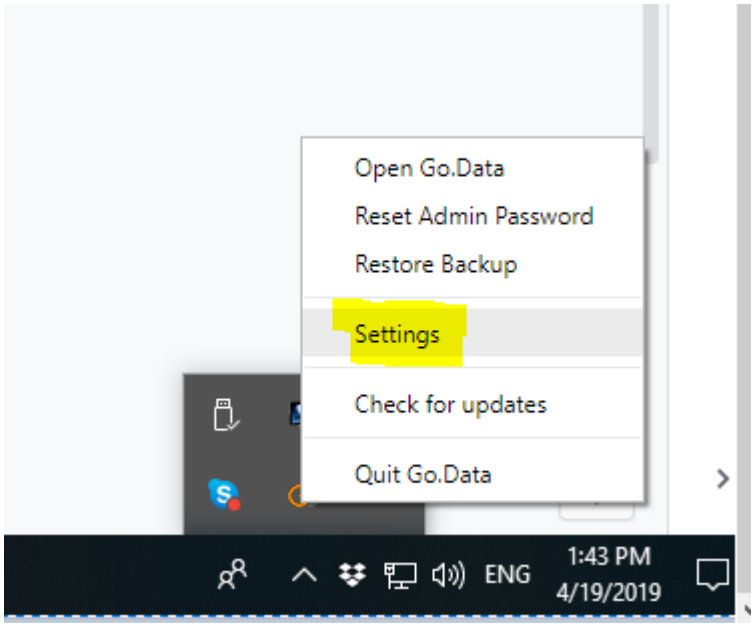
Работа с MongoDB

При нормальной работе пользователям не требуется доступ к MongoDB, и непосредственное взаимодействие с исходной базой данных не рекомендуется во избежание повреждения данных.

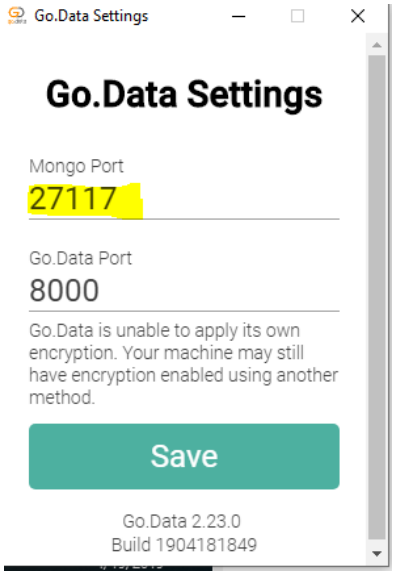
Однако для целей отладки или вмешательства на низком уровне пользователю необходим доступ к MongoDB, для потребуется следующие действия.

Для приложений под Windows:

1. Установите Robo 3T (Robo 3T).
2. Запустите приложение Go.Data
3. Определите порт, на котором работает Mongo
    a. Откройте всплывающее окно настройки



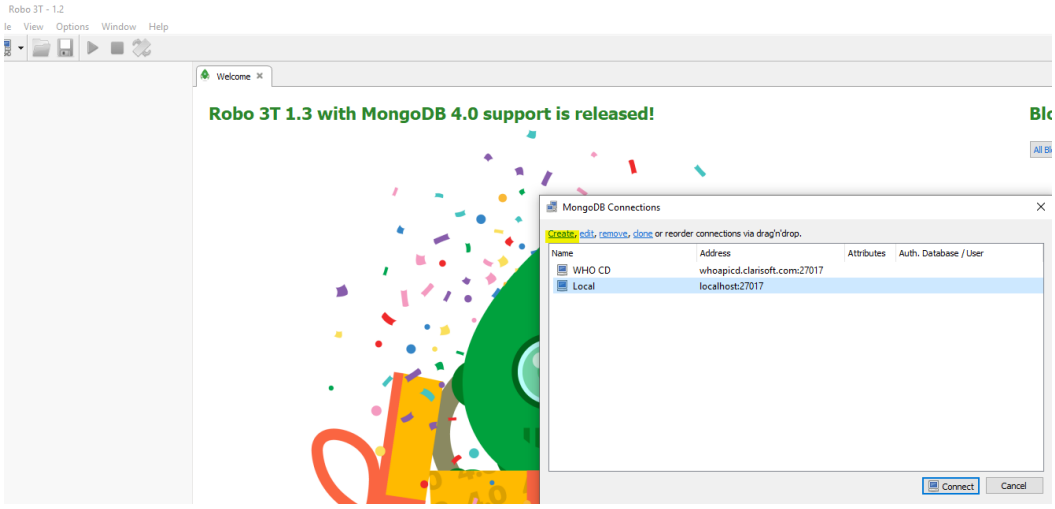
    b. Определите порт mongo (если значение не было установлено самостоятельно, в большинстве случаев используется порт 27017)



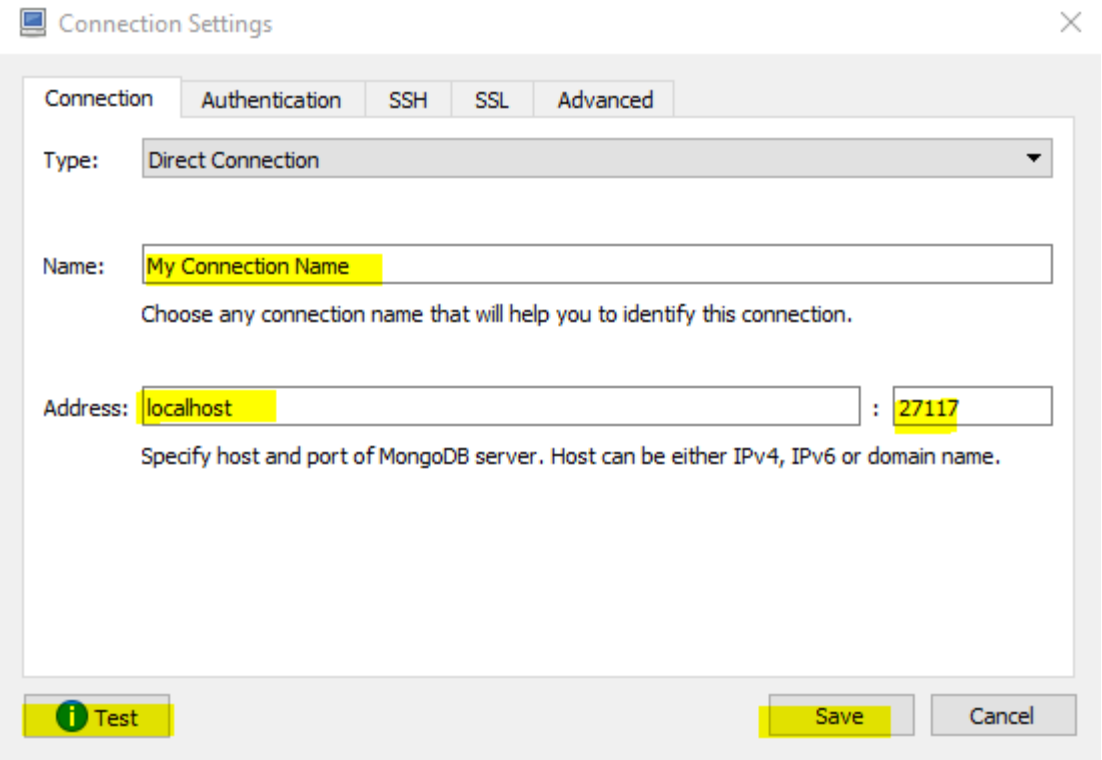
4. Подключитесь к базе данных:
   ВНИМАНИЕ: Любые изменения, внесенные в структуру базы данных/записей, могут нарушить работу приложения, если они нарушают ожидаемое поведение.

   Рекомендуется произвести резервное копирование перед началом работы с базой данных.

   a. Откройте приложение Robo 3T

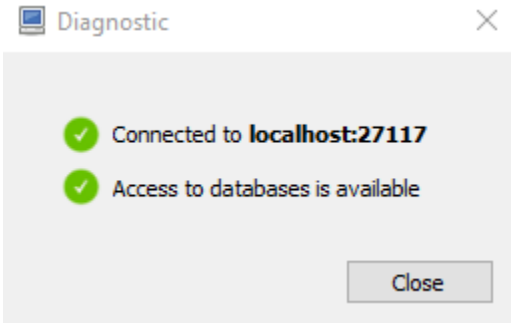


   b. Нажмите кнопку «Create» (Создать) подключение, если вы этого еще не сделали (этот шаг выполняется только один раз, поскольку далее будет использоваться ранее созданное и сохраненное подключение)



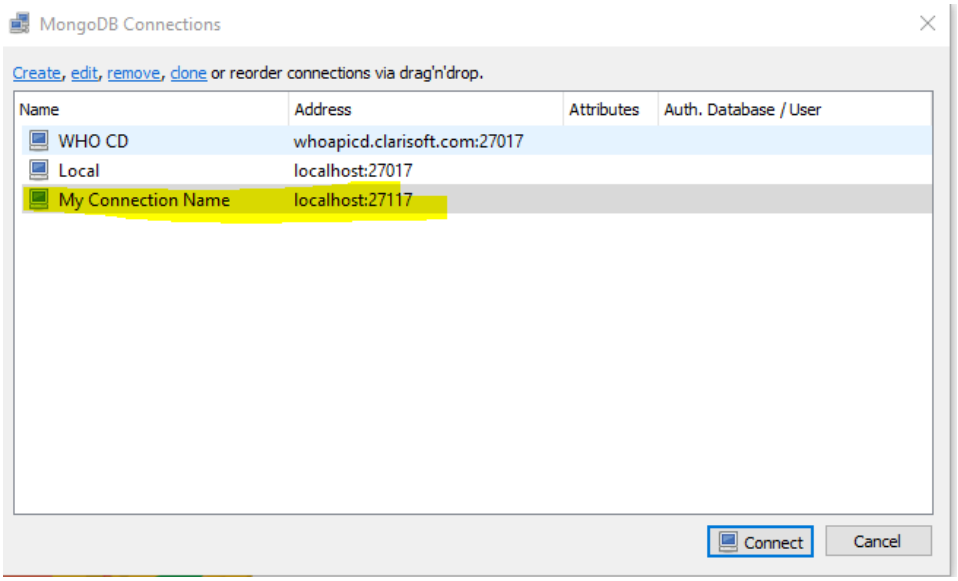
i. Должны быть заполнены следующие поля
   - Name: Имя подключения - может быть любым
   - Address
     - если копия Go.Data была установлена на данном компьютере, в этом поле необходимо оставить значение **localhost**
     - Port: сюда подставляется порт, определенный на шаге 3b (в большинстве случаев используется порт 27017, если он не был изменен)

ii. Нажмите «Test» (Тест)
   - Должно появиться сообщение, что подключение установлено



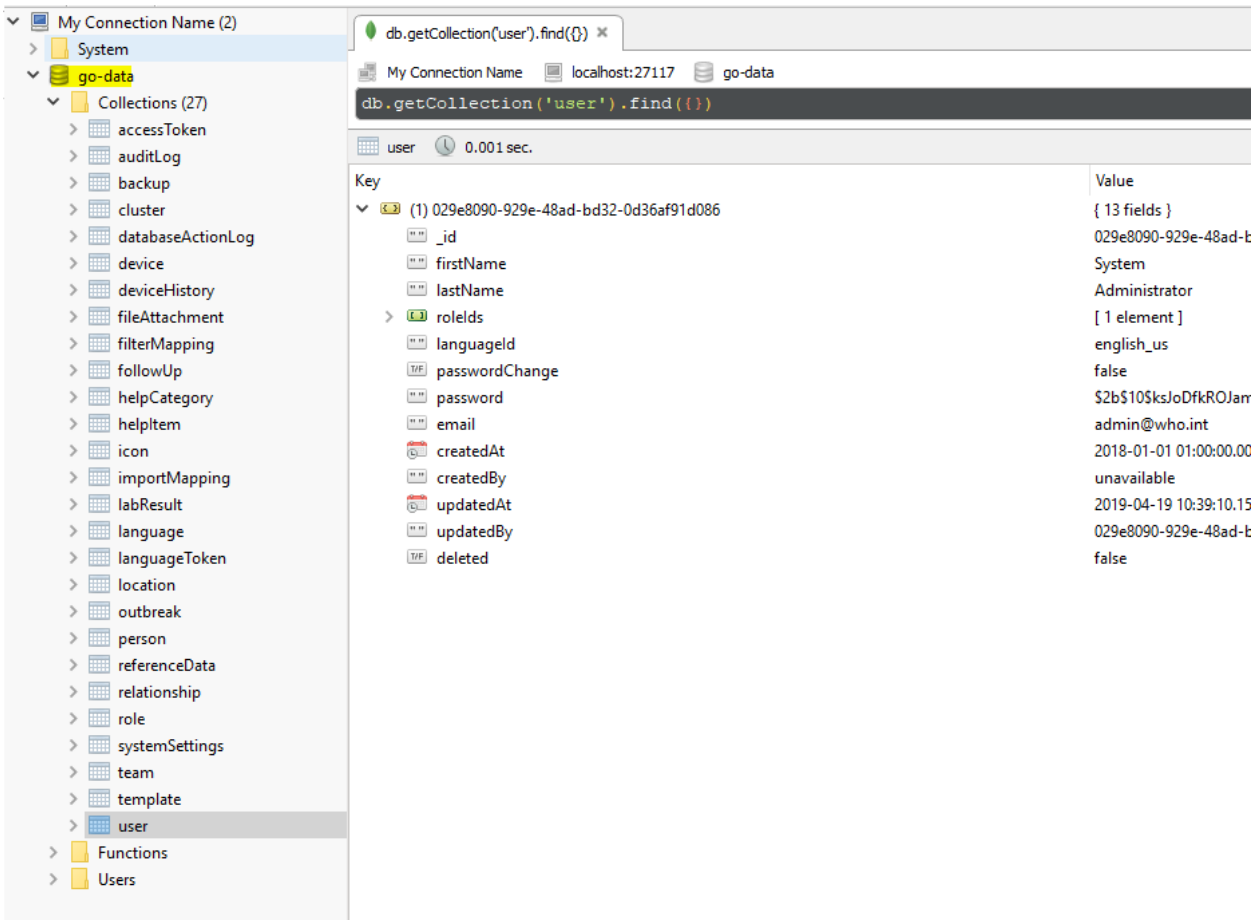
   - В противном случае убедитесь, что:
     - Приложение Go. Data не закрыто
     - Выбран верный порт подключения (см. шаги 3a и 3b)

iii. После установления подключения можете нажать «Save» (Сохранить). Подключение появится в диалоговом окне подключений, которое открывается каждый раз, когда запускается приложение Robo 3T.



c. Выберите свое подключение и нажмите кнопку «Connect» (Подключить)
   i. Если подключение было создано ранее, но вы его не видите, убедитесь, что:
      - Приложение Go. Data не закрыто
      - Выбран правильный порт для подключения (см. шаги 3a и 3b)

d. Если все правильно, вы должны видеть базу данных Go.Data. Здесь можно просматривать или изменять записи (**ВНИМАНИЕ:** Любые изменения, внесенные в структуру базы данных/записей, могут нарушить работу приложения, если они нарушают ожидаемое поведение.)



Настройка производительности

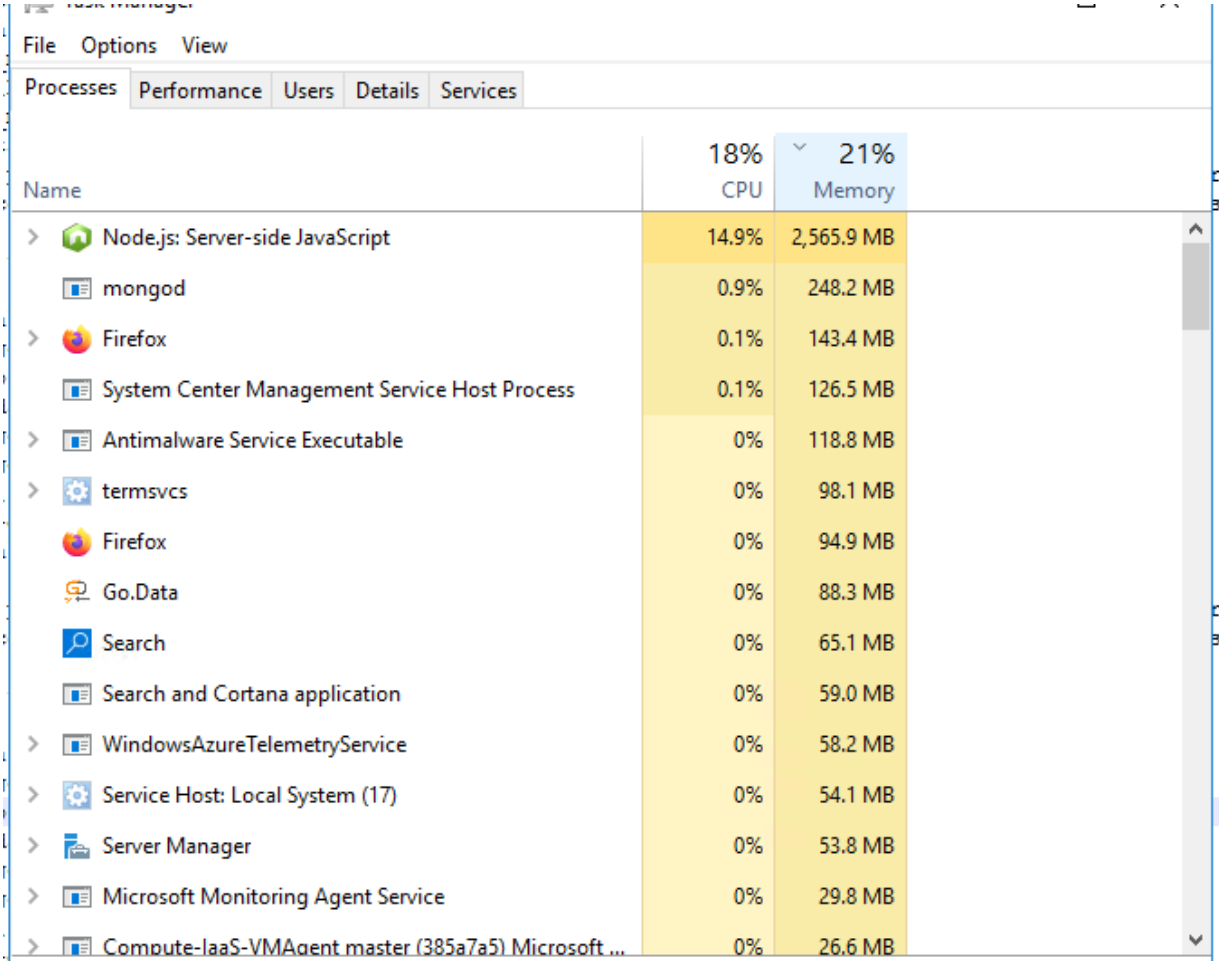
Программа Go.Data была подвергнута нагрузочному тестированию с использованием примерно до 100 000 случаев и контактов (всего) на компьютере со средними характеристиками для проверки стабильной производительности серверного компонента.

Загрузка данных в мобильное приложение всегда должна быть ниже, так как целью является отслеживание контактов, и количество контактов, которые человек может увидеть за один день, должно быть ограниченно, т.е. то, что Go.Data отправляет только подмножество данных на мобильный телефон, обосновано деловыми соображениями.

В случае снижения производительности при более высоких объемах загрузки данных в Go.Data доступны следующие параметры для настройки данного программного продукта:
-

1) Основной рабочий процесс Go.Data - это процесс Node.js, как показано ниже. Необходимо проверить, нет ли ограничения на использование объем оперативной памяти, который может быть предоставлен этому процессу, и увеличить его, если такое ограничение установлено. На сервере под Windows 10 данный процесс

иногда прерывался при достижении 1,4 Гб памяти, поэтому увеличение ее объема от 3 Гб до 8 Гб увеличит производительность при необходимости.



2) Go.Data использует одно процессорное ядро и не может воспользоваться несколькими физическими или виртуальными ядрами. При установке Go.Data на виртуальные машины лучше иметь одно ядро с высокой производительностью, чем несколько ядер.
3) Go.Data работает с набором данных для одной вспышки заболевания за раз, и поэтому, при вспышке, которая может привести к большому объему данных, можно рассмотреть возможность настройки этого набора данных как нескольких вспышек. Типичным сценарием является создание отдельных вспышек для регионов страны, а не одной национальной вспышки. Такая настройка действительно затрудняет объединение данных для глобальной отчетности, но будет способствовать повышению производительности.
4) Go.Data можно разделить между несколькими компьютерами для распределения нагрузки на обслуживание запросов – см. предыдущий раздел о настройке балансировщика нагрузки и разделении веб-компонентов и компонентов базы данных.

Для пункта 1, чтобы настроить выделение памяти для всех процессов nodejs, запущенных на компьютере, нужно установить переменную окружения (как в Windows, так и в Linux) и

настроить ее так, чтобы она сохранялась после перезагрузки компьютера и не теряла своего значения.

Для этого требуется переменная среды **NODE_OPTIONS**. Более подробную информацию о ней можно найти здесь:

https://nodejs.org/docs/latest-v8.x/api/cli.html#cli_node_options_options

Свойство **max-old-space-size** представляет максимальную выделяемую память. Значение переменной среды NODE_OPTIONS должно иметь примерно такой вид: -

'--max-old-space-size=4096'

Эта запись расшифровывается как «максимальная выделенная память для каждого процесса nodejs составляет 4096 Мб».

Максимальный объем памяти выражается в мегабайтах и может быть любым значением, если хост-компьютер обладает достаточной памятью для выделения этого объема.

Поскольку несколько процессов nodejs могут выполняться одновременно, это означает, что хост-компьютер должен обеспечить максимальное выделение памяти для всех активных процессов, то есть количество процессов * 4096 в приведенном выше примере.

Память будет использоваться по необходимости, поэтому большинство процессов nodejs не достигнут максимального разрешенного им объема памяти. Трудно предсказать, сколько памяти должен иметь хост, так как невозможно предвидеть, сколько процессов nodejs будет выполняться одновременно и сколько памяти потребуется каждому из них (это зависит от таких моментов, как количество одновременных пользователей и так далее).

Безопасность в программе Go.Data

Безопасность

Go.Data участвовала в трех проверках безопасности перед выпуском программы, две из которых проводились самой ВОЗ и одна - государством-членом ВОЗ. Были проведены как автоматизированные, так и ручные (тестирование на проникновение) тесты.

Ниже приведен список особенностей, связанных с безопасностью Go.Data: -

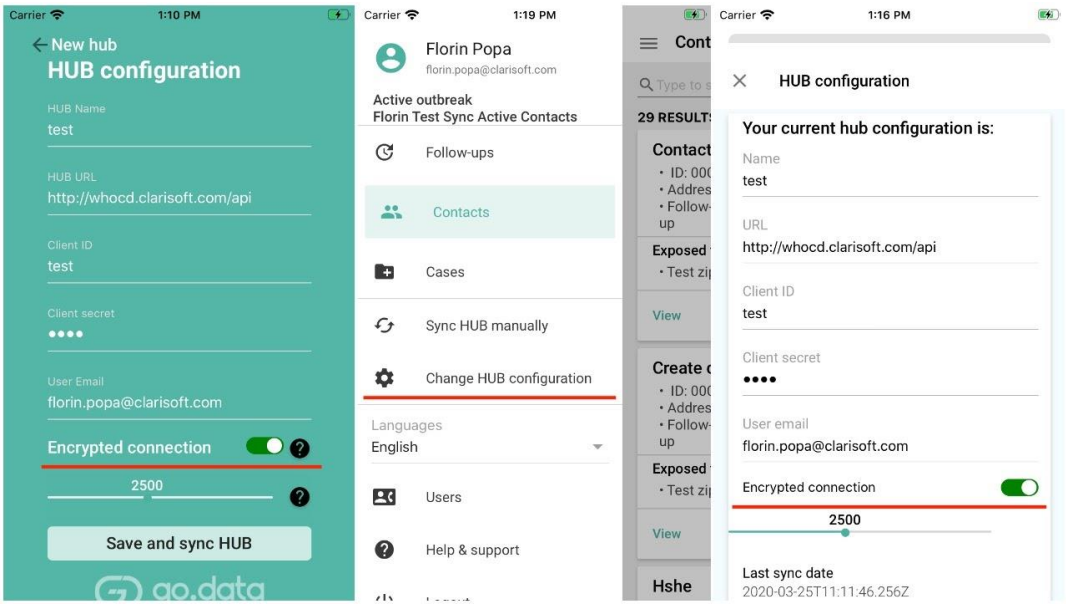
- Конфиденциальные данные, такие как пароли, шифруются.
- Captcha может быть дополнительно включен на экранах, связанных с паролями.
- По умолчанию все маркеры аутентификации имеют время жизни (TTL) 10 минут. Данное значение можно уменьшить или увеличить, внеся изменения в файл config.json. TTL маркера учитывается начиная с последнего запроса, сделанного с использованием данного маркера. Поэтому при активном использовании веб-сайта время жизни маркера не должно заканчиваться за это время.

```
"authToken": {
  "ttl": 600 // seconds
}
```

- Одновременно может быть активен только один маркер аутентификации для каждой учетной записи пользователя (при входе в систему предыдущие маркеры, связанные с этой учетной записью, отключаются). Поэтому в случае, если используется стороннее приложение для подключения к Go.Data API, рекомендуется периодически входить в систему, чтобы аннулировать предыдущий маркер. Таким образом уменьшается вероятность кражи действующего маркера.
- Все резервные копии можно зашифровать, указав ключ шифрования в файле config.json:

```
"backUp": {
  "password": "key"
}
```

- Связь между мобильными приложениями и API также может быть зашифрована: пользователи имеют возможность обмениваться зашифрованной информацией между приложением и API. Эта опция присутствует на экране конфигурации концентратора как переключатель «Encrypted connection» (Шифрованное подключение). По умолчанию шифрованное подключение включено. Позже эту опцию можно отключить на экране конфигурации концентратора.



- Для системы MongoDB шифрование не предусмотрено, поэтому она всегда должна иметь безопасное размещение.
- Go.Data не включает в себя собственный веб-сервер. Если требуется установить уровень безопасности HTTPS, программное средство должно быть размещено за соответствующим обратным прокси-сервером (или балансировщиком нагрузки). См. далее в данном документе.

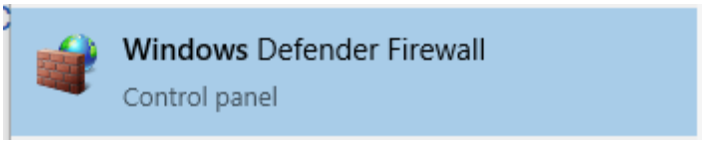
Порты

Go.Data использует различные порты для своего веб-интерфейса/интерфейса API и для связи с MongoDB. Если у вас возникли проблемы с подключением, особенно в случае мобильных телефонов, подключающихся к Go.Data, вы можете проверить правила межсетевого экрана для веб-порта, которые могут ограничивать трафик.

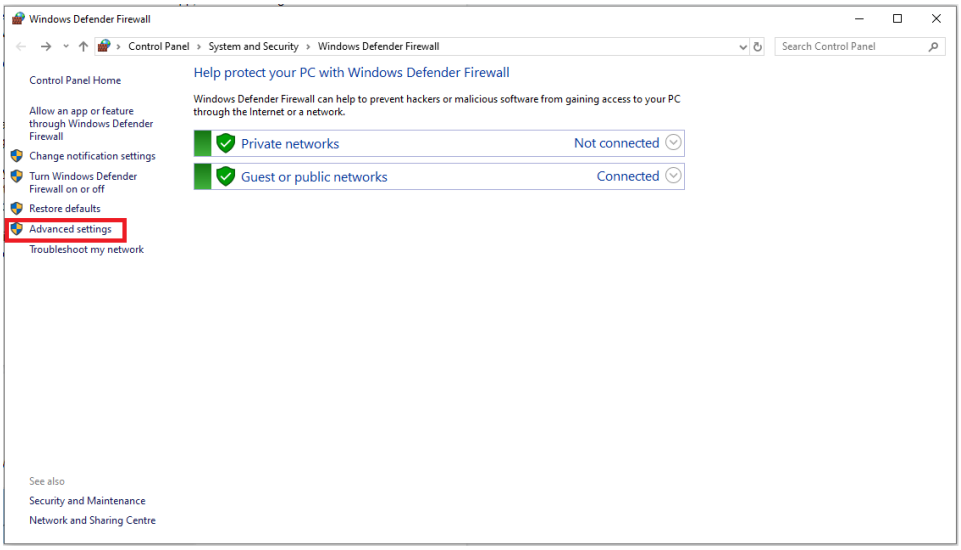
1. Чтобы иметь возможность синхронизировать смартфон с установленной Go.Data с копией на сервере, смартфон должен иметь возможность доступа к конечной точке Go.Data API на настроенном порту. Поэтому и телефон, и компьютер, на котором установлен сервер Go.Data, должны быть в одной сети Wi-Fi или Go.Data должна быть открыта на общедоступном URL-адресе через обратный прокси-сервер. Первый тест всегда заключается в том, чтобы протестировать IP-адрес Go.Data со смартфона и посмотреть, виден ли компьютер телефону.

2. Если проблема не устранена, а Go.Data установлена в Windows, может потребоваться добавить исключение в межсетевой экран Windows. Для этого выполните следующие шаги:

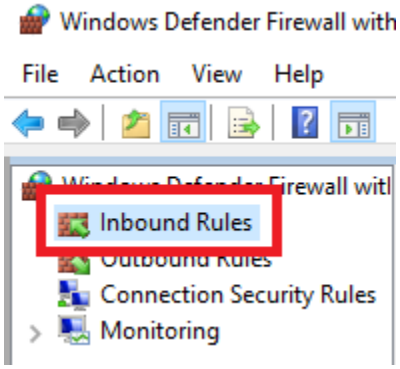
2.1 Найдите межсетевой экран Защитника Windows в Панели Управления



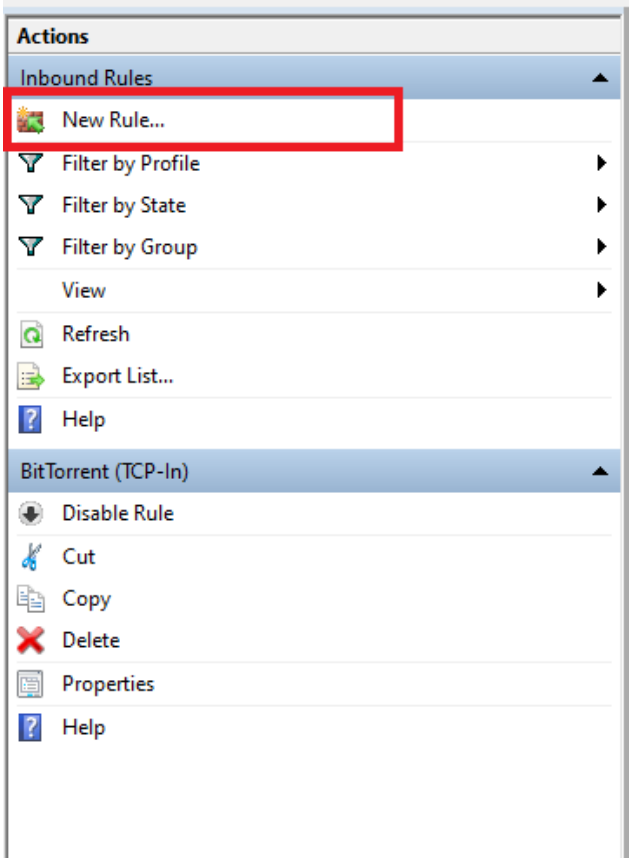
2.2 Перейдите к Расширенным настройкам



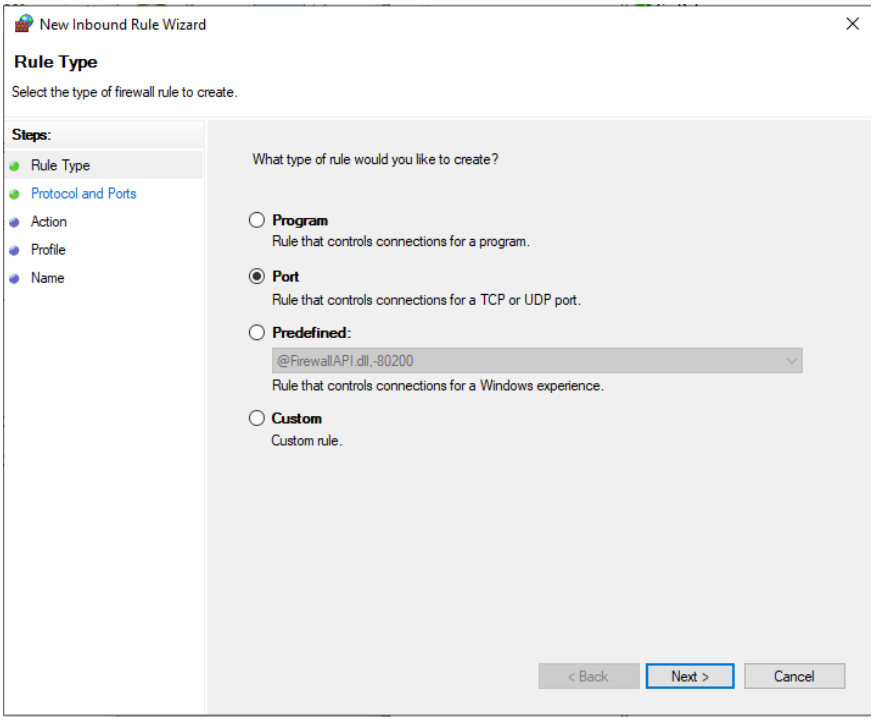
2.3 Выберите «Правила для входящих подключений»



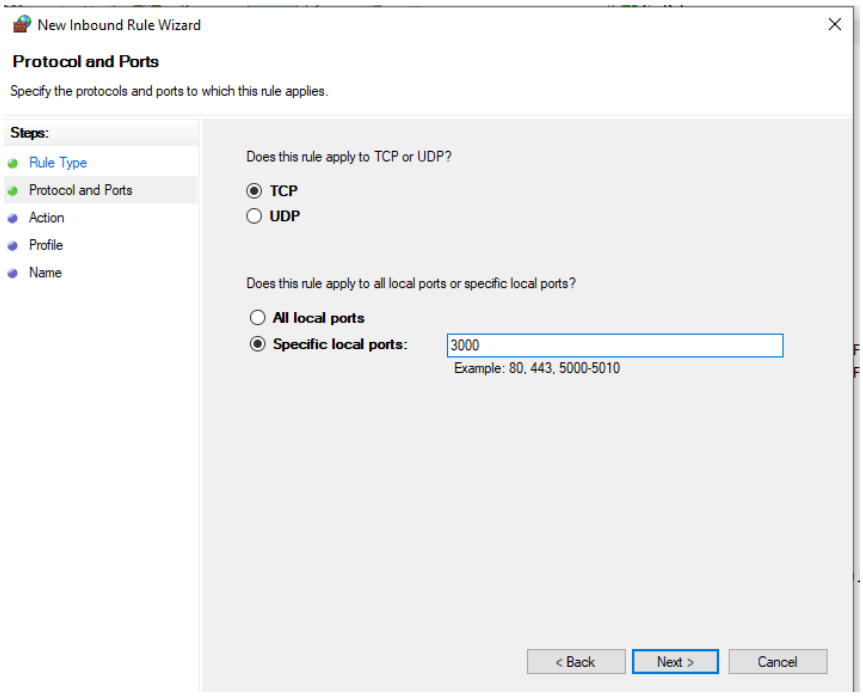
2.4 В панели справа нажмите на «Новое правило»



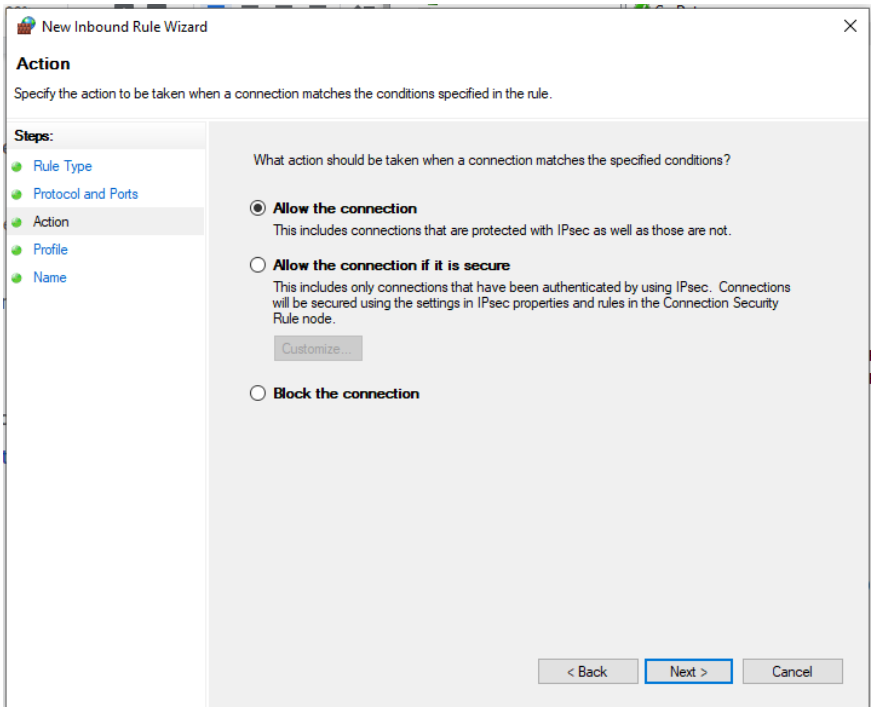
2.5 Выберите «Порт»



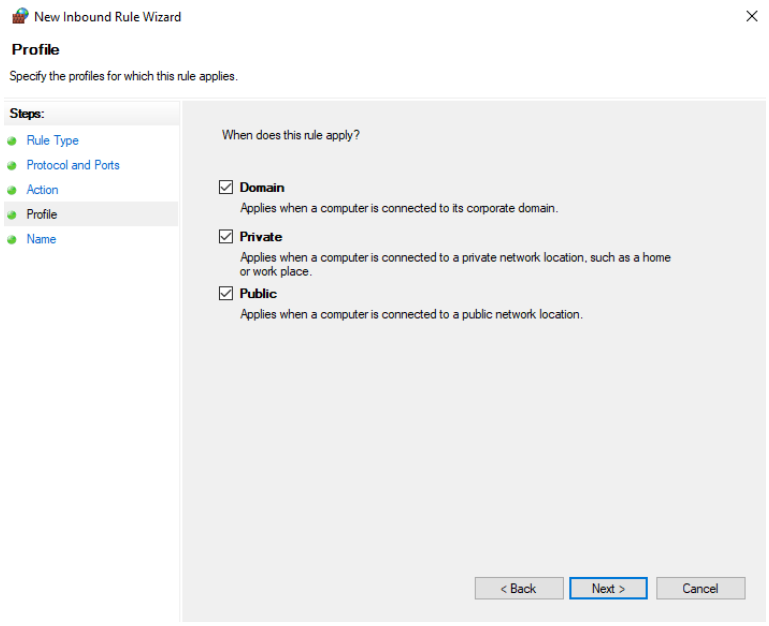
2.6 Выберите «TCP» и добавьте номер порта (по умолчанию 3000)



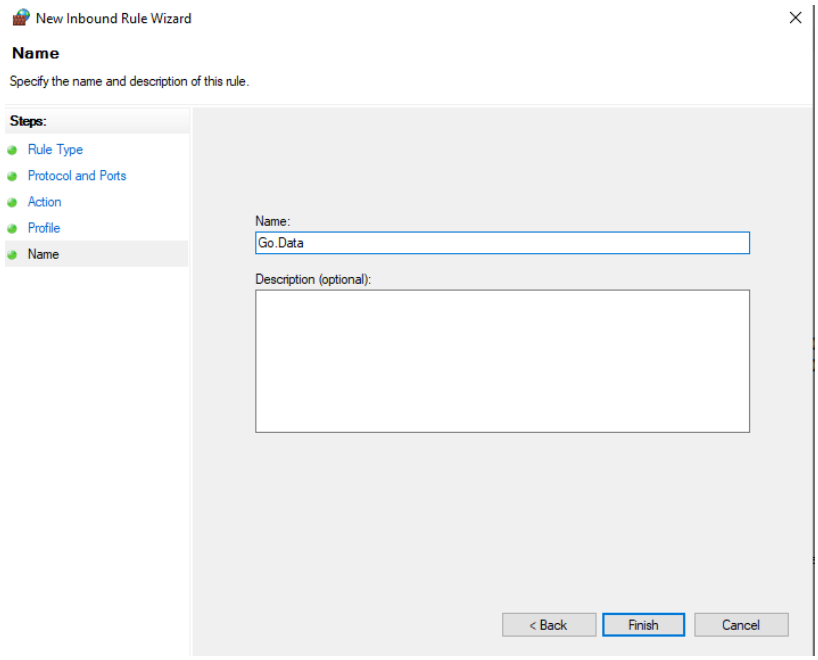
2.7 Выберите «Разрешить подключение»



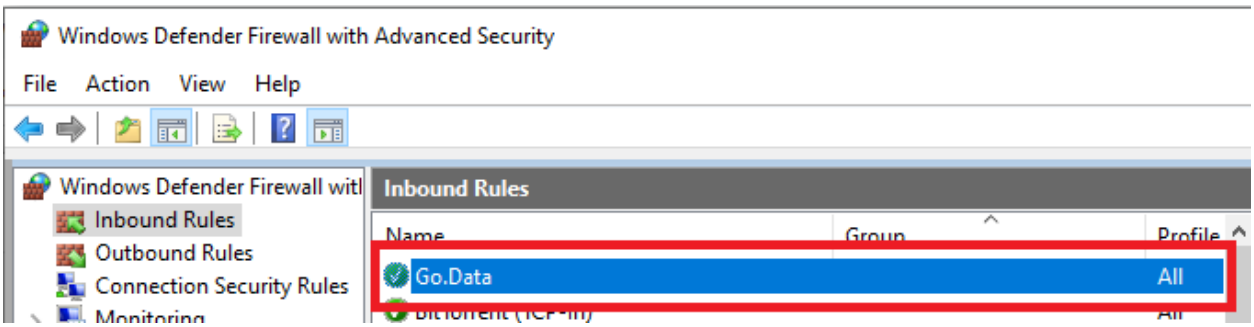
2.8 Выберите, в каком случае правило должно применяться



2.9 Задайте имя нового правила



2.10 Нажмите «Завершить»
2.11 Проверьте работу нового правила для входящих подключений



HTTPS

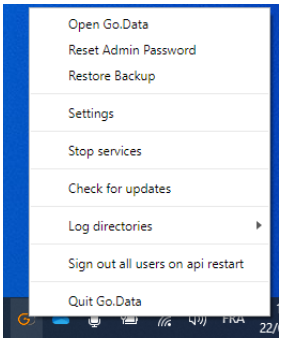
Для общедоступных систем в настоящее время нет возможности включить https из самого приложения Go.Data. Эта возможность может появиться в будущих версиях программного средства.

Свойства «public» в файле **config.json** используются только для сброса пароля и приложений electron. Поэтому недостаточно просто изменить свойство протокола. Пользователи должны будут иметь прокси-сервер перед службой Go.Data.

Этого можно легко достичь, настроив сервер Apache, Nginx, IIS, балансировщик нагрузки и т.д. с помощью SSL-сертификата, который виден вне частной сети (прокси-сервера). Этот сервер будет осуществлять маршрутизацию данных на частный сервер Go.Data (который доступен только в частной сети).
Эту настройку можно выполнить на том же самом компьютере при условии правильной настройки.

До тех пор, пока Go.Data не будет поддерживать прямое применение SSL-сертификатов, это единственный способ обеспечения безопасного доступа.

Сброс пароля администратора (Windows)



Для сброса пароля администратора в ОС Windows администратор должен получить доступ к самому серверу Go.Data и щелкнуть правой кнопкой на иконку Go.Data в панели задач.

Будет показан список опций, который включает в себя, помимо других важных операций, «Reset Admin Password» (Сбросить пароль администратора).

Эта опция является надежным способом получить контроль над пользовательской копией приложения Go.Data если все другие параметры входа в систему были утеряны, а запрос электронной почты для сброса пароля не может быть выполнен.

Сброс пароля администратора (Linux)

Для сброса пароля администратора в Linux администратор должен выполнить следующие действия:

1. Войдите в Linux
2. Запустите оболочку консоли терминала
3. Измените каталог на установочный каталог Go.Data
4. Выполните команду <installdir>\go-data-reset-password-x64.sh --file <backupfile>
5. После завершения пароль будет сброшен на пароль по умолчанию. При повторном входе в систему пользователю будет предложено сменить пароль.

Go.Data API

Go.Data предоставляет прикладной программный интерфейс (API), который используется для всех взаимодействий между веб-интерфейсом, приложениями для смартфонов и даже между копиями Go.Data, если настроено несколько копий программного продукта для обмена данными в модели «сервер, предоставляющий службу/клиентское приложение».

Это означает высокую гибкость API, и любая операция, которая может быть выполнена из веб-интерфейса/со смартфона, также может быть выполнена путем вызова соответствующего метода direct.

Осуществление доступа

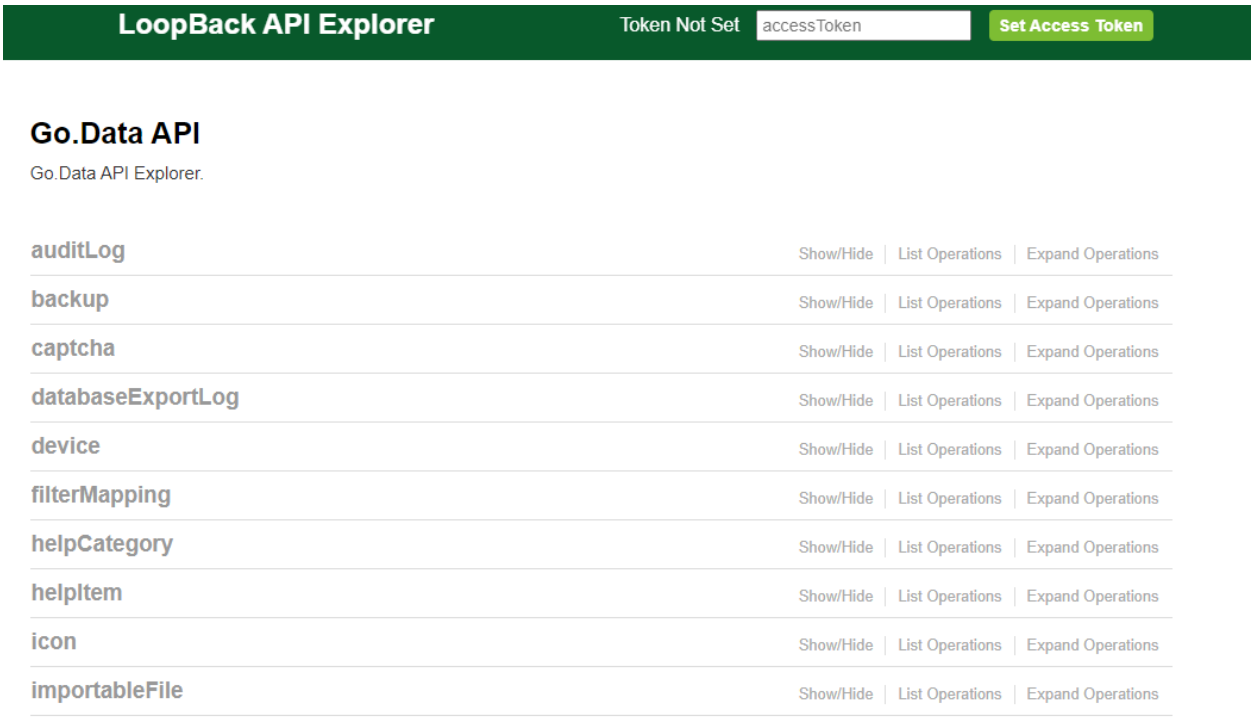
Как указывалось ранее в данном документе, самодокументированное описание методов API можно просмотреть с помощью Loopback Explorer, добавив /explorer на конце любого URL-адреса Go.Data. Например, при установке Go.Data на локальном компьютере с настройками по умолчанию:

http://localhost:8000/explorer

Loopback explorer предоставляет некоторые примеры возможных операций, перечисляет параметры интерфейса и позволяет вручную вводить входные данные json и тестировать их с помощью API.

Существуют методы GET, POST, PASTE и т.д. для двустороннего обмена данными с Go.Data.

Знакомство с типами объектов данных, присутствующих в Go.Data, является необходимым для понимания API.



Аутентификация

Почти все методы, предоставляемые Go.Data, требуют, чтобы запрос направлялся аутентифицированным пользователем. Аутентификация осуществляется следующим образом: -

1) Необходимо осуществить запрос методом POST для передачи /users/login в электронном письме и пароля для действительного пользователя в Go.Data. Ниже приведен пример:

```
{
    "email": "email@test.com",
    "password": "your_password_here"
}
```

2) Если переданные данные пользователя приняты, то этот метод вернет следующий ответ json, содержащий свойство «id», которое является маркером аутентификации, который этот пользователь будет использовать для последующих вызовов.

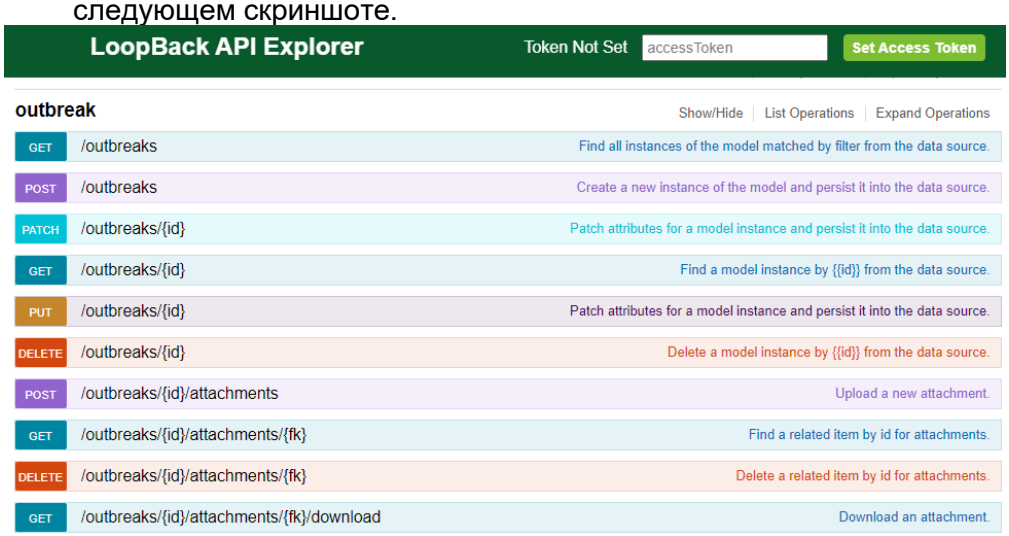
```
{
  "id": "Iv0S7Tj1F8RJaQSpBPHUlQ518KOUmK3SC7FlDvqLgbfbtjBt6ZFB77ViEQnKtqRq",
  "ttl": 600,
  "created": "2020-06-22T13:20:49.549Z",
  "userId": "18bf64ac-a36e-4890-a074-64aec702e21b",
  "createdAt": "2020-06-22T13:20:49.550Z",
  "createdBy": "system",
  "updatedAt": "2020-06-22T13:20:49.550Z",
  "updatedBy": "system",
```

```
  "deleted": false
}
```

3) Полученный маркер передается в заголовке последующих вызовов. В интерфейсе loopback explorer есть возможность ввести этот маркер в правом верхнем углу и запросить его сохранение для тестирования дальнейших вызовов.

4) Обратите внимание, что для всех пользователей Go.Data существует одна «активная» вспышка заболевания, и именно ее данные возвращаются в последующих вызовах с использованием маркера аутентификации, полученного для пользователя. Если вам нужно работать с несколькими вспышками в вашем коде, то вам необходимо будет или сменить пользователей, ИЛИ переключить активную вспышку текущего пользователя с помощью вызова API.

5) По этой же причине следующий вызов для работы с данными по вспышке обычно включает методы по алгоритму API «вспышка», как показано на следующем скриншоте.



6) Возможны и другие инструменты, такие как SoapUI или даже создание запроса и его отправка непосредственно в виде URL-адреса.
7) Например, чтобы получить список всех вспышек в Go.Data используя маркер аутентификации, уже полученный при вызове API моей локальной копии программы, вызов будет построен следующим образом: -

- `http://localhost:8000/api/outbreaks?access_token=VUCA57YkMIcF5R2M8vl6PuaNHoMU0Q3Mr4cmGEOH06CPblx6xflnAw0AfQaMYdZP`

Обратите внимание, что если ответ возвращает 422 «Invalid captcha» (Неверный ввод captcha) при попытке вызова метода входа из API, то это известная проблема, связанная с тем, что Go.Data использует файлы cookie, которую можно обойти следующим образом:

1. Captcha сохраняется в переменной сеанса, связанной с cookie, предоставляемой браузером только для конкретной веб-страницы. Важно, чтобы в вызове '**GET** /captcha/generate-svg' не был установлен forComponent=login. Если вы хотите войти в систему непосредственно из API, не используйте вызов этого метода, так как именно этот метод заставляет метод входа ограничивать доступ без ввода captcha. В этом случае перезапустите API, после чего вызовите вход в систему без предварительного вызова этого метода.

2. Если вы используете запросы API Explorer, при которых браузер прикрепляет файл cookie, используемый переменной сеанса, один из примеров использования которого: откройте страницу входа в систему, затем откройте explorer и попробуйте войти в систему, то вы получите ошибку 422 «Invalid captcha» (Неверный ввод captcha), после чего выполните следующие действия:

- закройте все страницы веб-сайта, закройте браузер
- откройте браузер
- откройте explorer непосредственно без захода на страницу входа в систему (в этом случае браузер сбрасывает переменную cookie, и работа может быть продолжена без необходимости перезапуска API).

Другие возможные решения:

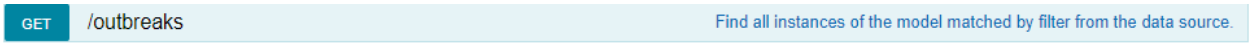
- используйте браузер в режиме инкогнито
- используйте другой браузер вместо того, который используется для веб-интерфейса пользователя
- закройте вкладку, после чего очистите кэш cookie, связанных с данным веб-сайтом

Работа с данными

После аутентификации появляется возможность двустороннего обмена данных с API Go.Data, а время жизни маркера аутентификации продлевается при каждом успешном сеансе связи.

Пример вспышки заболевания

Например, для получения подробной информации об одной вспышке заболевания (наименование вспышки, местоположение и т.д.), можно вызвать метод GET outbreaks, который будет построен следующим образом, путем передачи как маркера доступа (полученного при аутентификации), так и идентификатор интересующей вспышки.

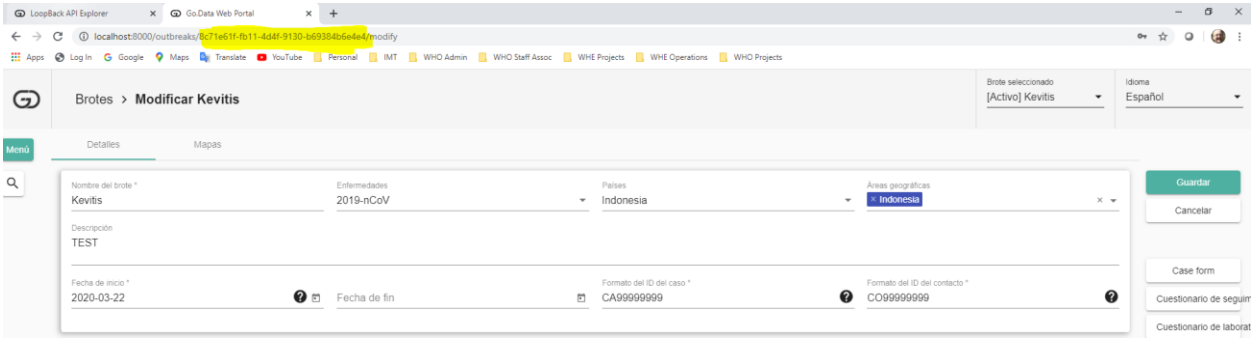



```
http://localhost:8000/api/outbreaks/8c71e61f-fb11-4d4f-9130-
b69384b6e4e4?access_token=VUCA57YkMIcF5R2M8vl6PuaNHoMU0Q3Mr4cmGEOH06CPblx6xfl
nAw0AfQaMYdZP
```

В этих вызовах используются глобальные уникальные идентификаторы (GUID), применяемые как основные ключи для записей в MongoDB.

В приведенном выше примере есть два способа, с помощью которых пользователь может найти идентификатор вспышки для использования: -

1) Если пользователь входит в Go.Data и переходит к записи, после чего идентификатор обычно появляется в URL-адресе в верхней части страницы. Ниже приведен пример для просмотра примера вспышки «Кевитис» и появившегося идентификатора.



2) Если пользователю нужно найти идентификатор, то это также всегда можно сделать с помощью метода. В этом примере метод GET outbreaks можно использовать для поиска всех вспышек, а затем найти идентификатор той, которая вас интересует.

Пример случая заболевания

Во втором примере вызов для получения всех случаев, относящихся к конкретной вспышке заболевания, будет иметь вид: -


```
http://localhost:8000/api/outbreaks/8c71e61f-fb11-4d4f-9130-
b69384b6e4e4/cases?access_token=VUCA57YkMIcF5R2M8vl6PuaNHoMU0Q3Mr4cmGEOH06CPb
lx6xflnAw0AfQaMYdZP
```

Обратите внимание, что любые ограничения, обусловленные требованиями безопасности, ограничивающие данные, которые пользователь может видеть, также будут применяться к вызовам через API. Пользователи должны проходить аутентификацию с помощью учетной записи пользователя, которая имеет достаточные привилегии и доступ к данным, к которым они хотят получить доступ или которые требуется изменить.

Вызов для доступа к записи по конкретному случаю заболевания будет включать в себя маркер доступа, идентификатор вспышки и идентификатор случая, который программа должна вернуть:


```
http://localhost:8000/api/outbreaks/8c71e61f-fb11-4d4f-9130-
b69384b6e4e4/cases/3cd71bf6-afac-40d3-a32d-
a1793cfe7638?access_token=HNm29JYiCIa0sNk5kjyTl8FeGKJmhFMiWAhGL6FOBVcBSCc2s2J
DQ3EnLH4dFt4l
```

Для отправки данных обратно в Go.Data следует использовать те же структуры json, но если есть какие-либо поля, которые не должны быть изменены, то эти свойства JSON следует опустить, а не оставлять пустыми.

Например, если мы хотим обновить первое имя для Case (Случая), который был получен в предыдущем вызове, то нужно сделать вызов следующему методу PUT:

Передаваемые данные будут представлять собой только поле, которое должно быть изменено: -

```
{
  "firstName": "TestDemo"
}
```

При этом осуществляется следующий вызов: -

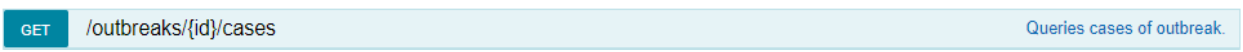
```
http://localhost:8000/api/outbreaks/8c71e61f-fb11-4d4f-9130-
b69384b6e4e4/cases/3cd71bf6-afac-40d3-a32d-
a1793cfe7638?access_token=HNm29JYiCIa0sNk5kjyTl8FeGKJmhFMiWAhGL6FOBVcBSCc2s2J
DQ3EnLH4dFt4l
```

Пример использования фильтра

Для использования фильтров, доступных для вызовов метода, в синтаксисе должно использоваться ключевое слово «where» и последовательность элементов для применения фильтра:

{"where":{"***fieldname***": "***filtervalue***"}}

Для примера, если осуществляется фильтрование по методу GET /outbreak/{id}/cases, основанному на Case ID,



то фильтр будет иметь вид: -

{"where":{"visualId": "CA00000001"}}

При передаче в качестве части URL для этой строки потребуется шифрование URL. Ниже приводится полный список примеров.

ЗАПРОС JSON *{"where":{"createdAt":{"$gt":"2020-04-14T00:00:00Z"}}}*

ЗАШИФРОВАННЫЙ URL *%7B%22where%22%3A%7B%22createdAt%22%3A%7B%22%24gt%22%3A%222020-04-14T00%3A00%3A00Z%22%7D%7D%7D*

КОМАНДА НА ЗАПРОС */outbreaks/* ***[[МАРКЕР ВСПЫШКИ]]*** */cases?filter=%7B%22where%22%3A%7B%22createdAt%22%3A%7B%22%24gt%22%3A%222020-04-14T00%3A00%3A00Z%22%7D%7D%7D&access_token=****[[МАРКЕР ДОСТУПА]]***

ОКОНЧАТЕЛЬНЫЙ ЗАПРОС GET *https://godata.gov.mt/api//outbreaks/* ***[[МАРКЕР ВСПЫШКИ]]*** */cases?filter=%7B%22where%22%3A%7B%22createdAt%22%3A%7B%22%24gt%22%3A%222020-04-14T00%3A00%3A00Z%22%7D%7D%7D&access_token=****[[МАРКЕР ДОСТУПА]]***

Пример программы

На сегодняшний день партнерами Go.Data создан код интеграции с использованием C#, Python и R. Вы можете присоединиться к сообществу Go.Data по ссылке (https://community-godata.who.int).

На момент составления данного руководства ВОЗ проверяет пример программы на вызов API и вскоре опубликует его в своем репозитории Git.