UNLIMITED PRODUCTION

eXpress

Система коммуникаций

Outlook-плагин

Руководство по установке

Версия Keycloak 2.1.0.0

Почтовый адрес:	127030, г. Москва, ул. Новослободская, д. 24, стр. 1
Телефон:	+7 (499) 288-01-22
E-mail:	sales@express.ms
Web:	https://express.ms/

ОГЛАВЛЕНИЕ

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

Термин	Определение
AD	Active Directory — служба каталогов корпорации Microsoft для операционных систем семейства Windows Server
API	Application Programming interface — интерфейс для взаимодействия программ и приложений
CTS	Corporate Transport Server — корпоративный сервер
ETS	Enterprise Transport Server — сервер предприятия
JSON	Текстовый формат обмена данными, основанный на JavaScript
Single CTS	Единый корпоративный сервер
БД	База данных
ПК	Персональный компьютер
ПО	Программное обеспечение
Разделенный CTS	Разделенный корпоративный сервер: Front CTS и Back CTS
СК «Express», Express, система	Система коммуникаций «Express»
Чат-бот	Чат-бот Conference Notifier Bot
express-core-service	Серверное приложение для Outlook-плагина, обеспечивающее основную функциональность (создание и изменение конференций)
express-template-service	Серверное приложение для Outlook-плагина, обеспечивающее дополнительную функциональность (создание и изменение шаблонов письма, текст, логотип, подпись и т. п.)
Лог	Запись в журнале событий сервера и/или клиента
KeyCloak	приложение для реализации единой точки идентификации, аутентификации и авторизации. Это система идентификации и управления доступом с открытым исходным кодом (Open Source)
Docker	платформа контейнеризации с открытым исходным кодом, с помощью которой можно автоматизировать создание приложений, их доставку и управление
Контейнер	стандартизированный, изолированный и портативный пакет программного обеспечения, который включает в себя все необходимое для запуска приложения, включая код, среду выполнения, системные инструменты, библиотеки и настройки.
Docker-compose	Надстройка(утилита) над Docker, приложение на Python, которое позволяет запускать множество контейнеров одновременно и маршрутизировать потоки данных между ними.

Глава 1

ОБЩИЕ ТРЕБОВАНИЯ

Внимание! Для выполнения операций из настоящей инструкции необходимо обладать следующими компетенциями:

- администрирование Windows Server;
- администрирование Keycloak;
- администрирование PostgreSQL;
- администрирование Linux;
- администрирование Docker;
- администрирование eXpress;
- понимание JSON.

Перед началом работ убедитесь, что программные и технические средства соответствуют следующим требованиям:

- версия Docker не ниже Docker version 24.0.7;
- версия Docker-compose не ниже version 1.29.2;
- требования к БД:
  - версия PostgreSQL не ниже 13.11 (Debian 13.11-0+deb11u1, Windows);
  - наличие супер-пользователя на сервере PostgreSQL;
  - отдельная БД PostgreSQL, отдельный пользователь БД PostgreSQL: данный пользователь должен быть владельцем это БД и иметь все права на нее;
- наличие установленного и настроенного решения Keycloak не ниже версии 22.0.1;
- версия клиентской части Outlook должна быть не ниже 2013;
- Outlook должен использоваться с почтовым ящиком, который соответствует адресу e-mail, прописанному в учетной записи СК «Express»;
- наличие установленного и настроенного решения СК «Express» не ниже версии 3.8;
- в СК «Express» должен быть настроен чат-бот Conference Notifier Bot. Требования по настройке чат-бота и методы проверки корректности настройки описаны в Главе 3 «Установка и настройка чат-бота Conference Notifier Bot».
- должно быть настроено сетевое взаимодействие между компонентами по соответствующим портам;
- на клиентских ПК должен быть установлен пакет WebView2 Runtime версии не ниже 122.0.2365.66 (https://developer.microsoft.com/ru-ru/microsoft-edge/webview2/?form=MA13LH).

Если на виртуальной машине не будет других сервисов, то сервер Docker должен соответствовать следующим техническим требованиям:

Элемент	Параметры
Процессор	64-bit 2.4 GHz processor 2 core
Оперативная память	4 ГБ
Операционная система	Debian GNU. Не ниже Linux 11
Жесткий диск	Не менее 40 ГБ

Если на виртуальной машине не будет других сервисов, то сервер PostgreSQL должен соответствовать следующим техническим требованиям:

Элемент	Параметры
Процессор	64-bit 1.4 GHz processor 2 core
Оперативная память	2 ГБ
Операционная система	Debian GNU. Не ниже Linux 11
Жесткий диск	Не менее 40 ГБ

Все пользователи, которые хотят использовать Outlook-плагин, должны быть зарегистрированы в СК «Express», и быть активными. Проверить статус пользователя можно в консоли администратора СК «Express» – раздел «Пользователи» (Рисунок 1).

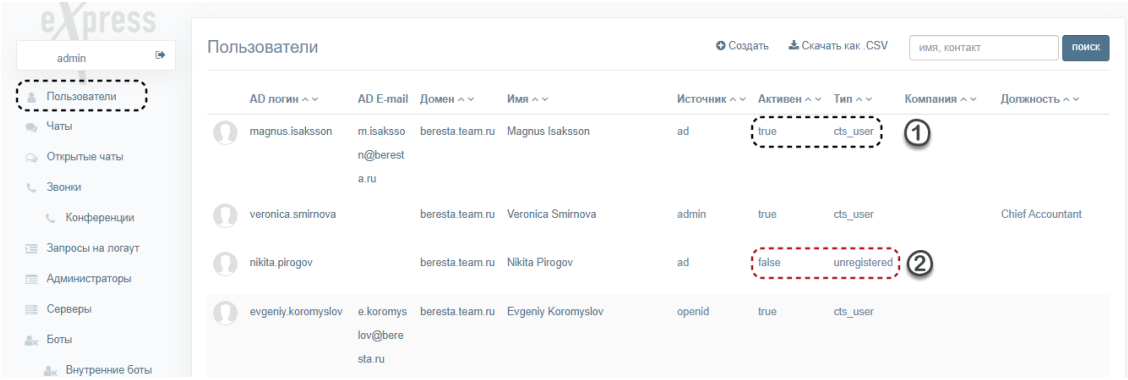

Рисунок 1. Статус пользователя: 1 – активный; 2 – неактивный

Глава 2

АРХИТЕКТУРА

В зависимости от используемого архитектурного решения СК «Express» применяются различные схемы развертывания серверной части плагина Outlook. Ниже описаны наиболее распространенные типовые архитектурные решения.

Конкретные схемы развертывания определяются требованиями заказчика после консультации с компанией-разработчиком.

При использовании единого шаблона письма все express-core-service могут использовать одну БД. При использовании разных шаблонов письма для разных express-core-service разверните для каждого из них свою БД и свой express-template-service.

SINGLE CTS ИЛИ РАЗДЕЛЕННЫЙ CTS

Если СК «Express» содержит один CTS-сервер, то разверните серверную часть плагина в одном экземпляре (Рисунок 2).

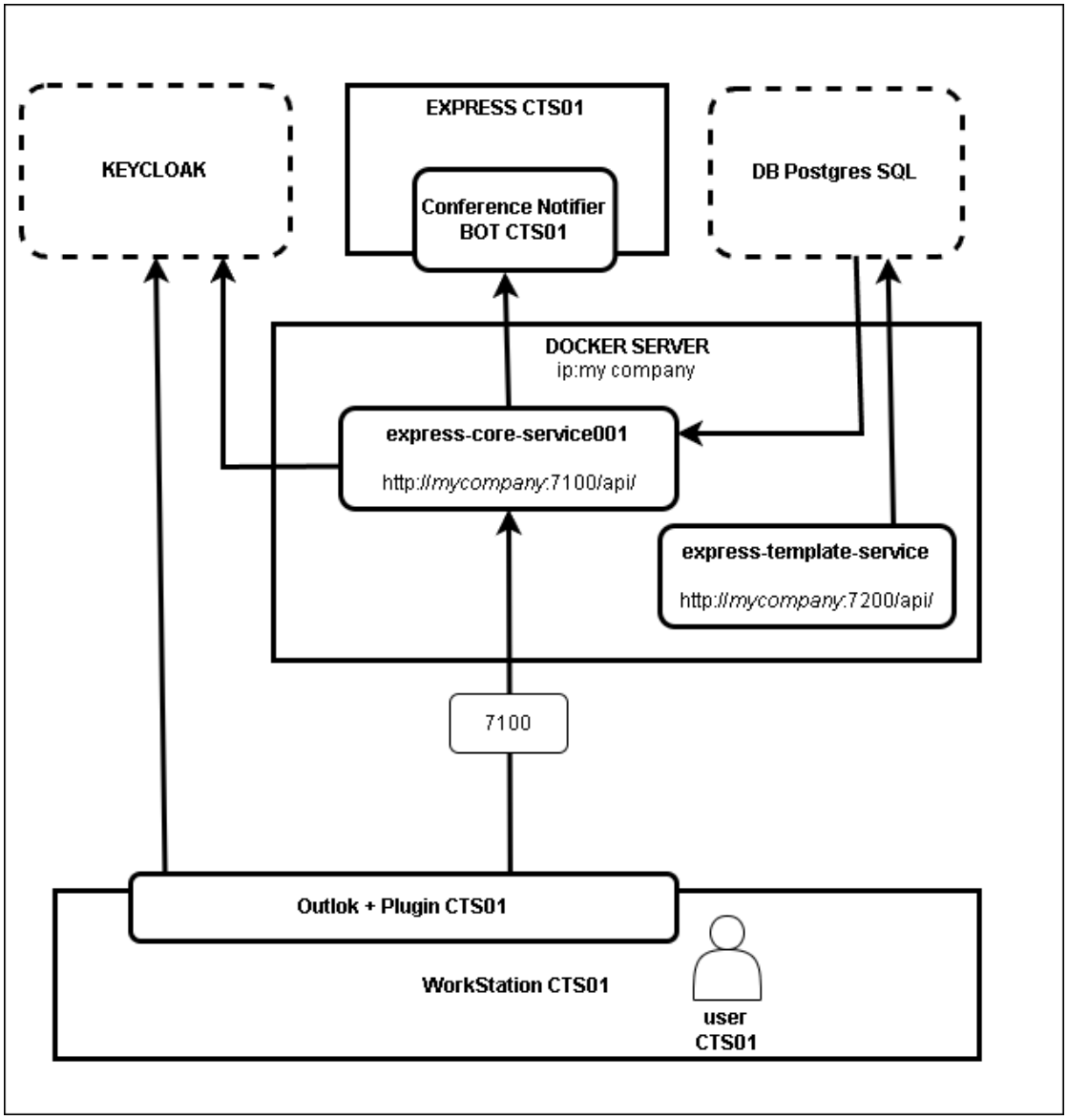

Рисунок 2. Типовая схема: 1 CTS, 1 express-core-service, 1 express-template-service

НЕСКОЛЬКО CTS

Если СК «Express» содержит несколько CTS-серверов, то разверните серверную часть плагина для каждого CTS-сервера отдельно (допускается развертывание на одном сервере Docker, но с разными портами и именами контейнеров, или на нескольких серверах Docker) (Рисунок 3).

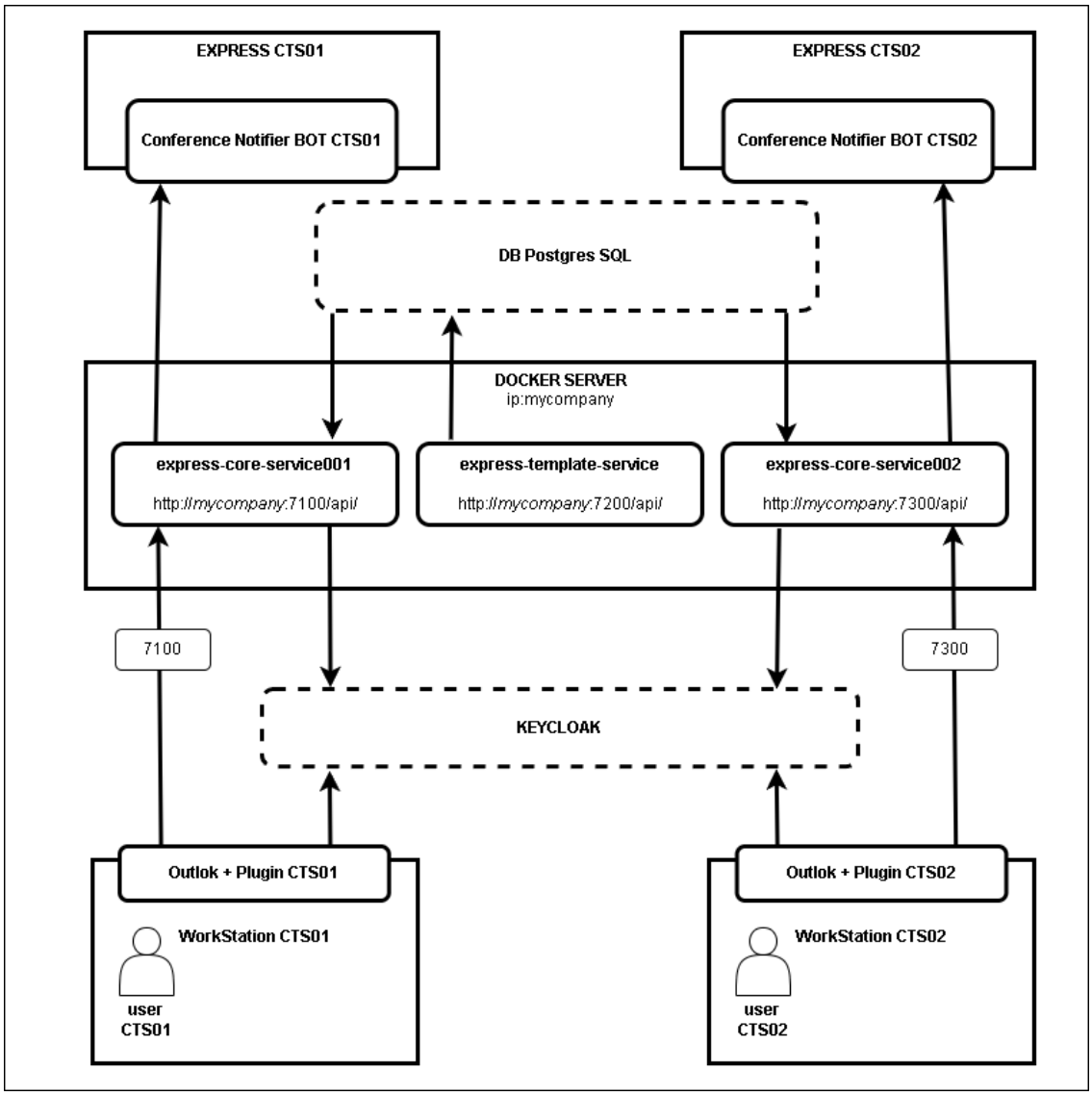

Рисунок 3. Типовая схема: 2 CTS, 2 express-core-service, 1 express-template-service

НЕСКОЛЬКО CTS И ETS

Если СК «Express» содержит несколько CTS-серверов, объединенных ETS-сервером (Рисунок 4), разверните серверную часть плагина только для ETS-сервера.

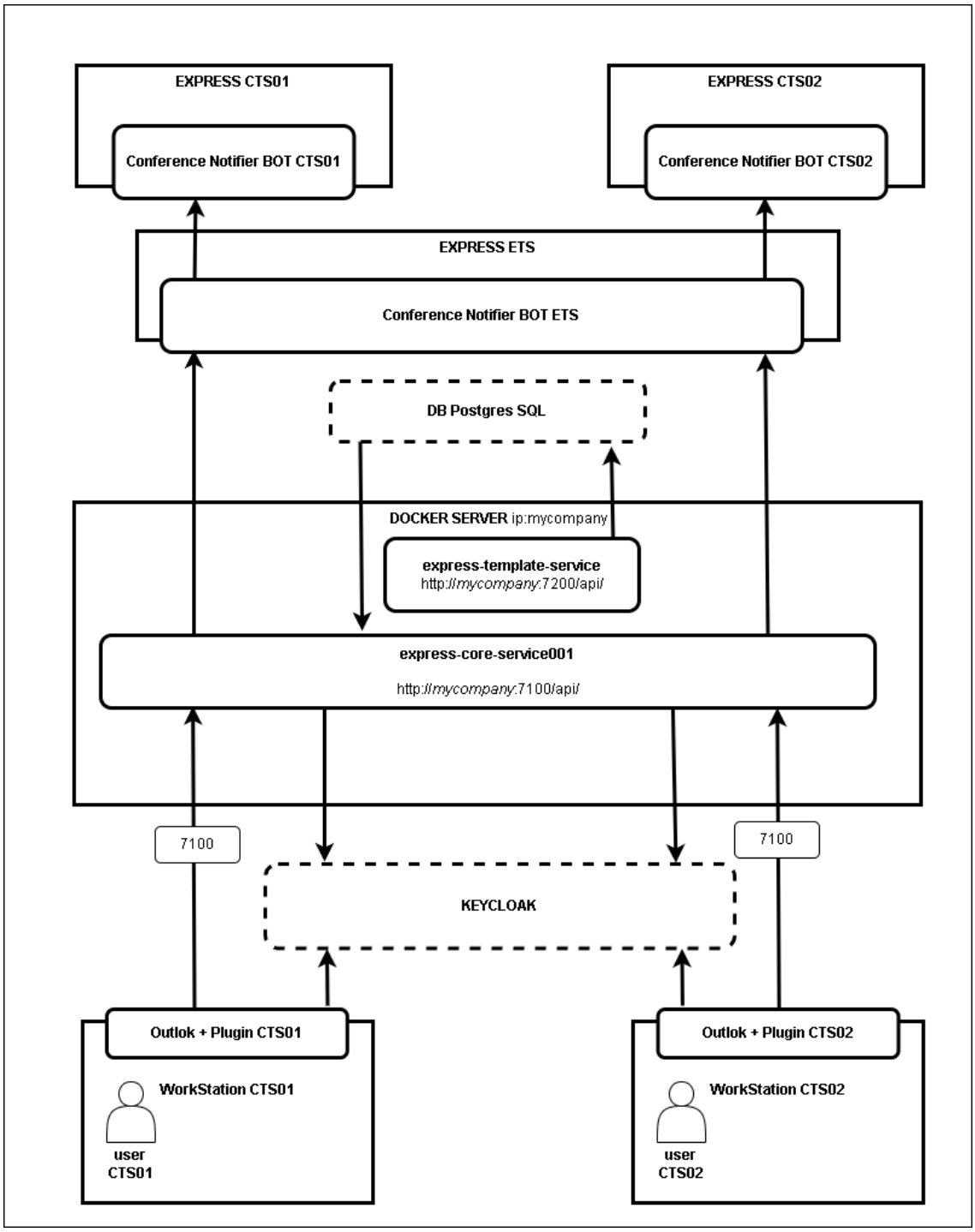

Рисунок 4. Типовая схема: 1 ETC, 2 CTS, 1 express-core-service, 1 express-template-service

Глава 3

УСТАНОВКА И НАСТРОЙКА ЧАТ-БОТА CONFERENCE NOTIFIER BOT

В разделе описывается порядок подключения API чат-бота Conference Notifier Bot на CTS-сервере программного решения СК «Express».

Примечание. Процедура описана для ОС Debian 11.4 и Ubuntu 2X. При возникновении проблем во время настройки чат-бота на других ОС рекомендуем обратиться к компании-разработчику.

ШАГ 1. ВКЛЮЧЕНИЕ API CONFERENCE NOTIFIER BOT БЕЗ ПАРОЛЯ

Внимание! При настройке чат-бота без защиты его может использовать любой пользователь из внешнего контура если он открыт.

Важно:

- если СК «Express» содержит несколько CTS, то данную процедуру следует выполнить на каждом CTS, который будет взаимодействовать с express-core-service;
- если СК «Express» содержит несколько один или несколько ETS, то данную процедуру следует выполнить на каждом ETS и CTS, который будет взаимодействовать с express-core-service.

Для включения API Conference Notifier Bot без пароля:

1. На CTS-сервере добавьте в файл settings.yaml[1] следующий код:

```
conference_bot_env_override:
  API_ENABLED: true
```

   В случае использования разделенной серверной архитектуры изменения вносится только в файл на сервер Back CTS.

2. Сохраните файл и запустите из /OPT/EXPRESS:

```
dpl -d conference_bot
```

3. Откройте веб-консоль администратора и перейдите в раздел «Боты → Внутренние боты → Conference Notifier Bot».

   Откроется окно «Редактирование бота».

4. В открывшемся окне проставьте отметку в поле «Включено» и нажмите кнопку «Сохранить», при необходимости добавьте порт 4000.

5. Проверьте доступность чат-бота с помощью запроса через CURL, например:

```
curl -X POST -H "Content-Type: application/json" -d
'{"name":"test_plugin_001","members":["user001@mydomain.com","
user002@mydomain.com"],"admins":["user001@mydomain.com
"],"creator":"user001@mydomain.com","start_at":"2023-12-
21T10:10:00.0Z","end_at":"2023-12-
21T11:10:00.0Z","link":{"link_type":"public","access_code":null}}'
https://my-CTS-domain.com/api/v1/conference_bot/conferences/
```

   Примечание. Красным цветом отмечены параметры, которые требуется актуализировать под конкретный запрос.

   Если чат-бот доступен, CURL выведет:

```
"status":"ok"
```

6. В веб-интерфейсе администратора CTS-сервера перейдите в раздел «Звонки → Конференции».

[1] Предполагаемый путь /opt/express/settings.yaml.

7. В поле поиска введите test_plugin_001 (был задан в п. 5) и нажмите кнопку «Поиск».

   Если в таблице будет выведена конференция с именем test_plugin_001, то чат-бот настроен правильно.

Для проверки статуса докер-контейнера после установки бота на сервере СК «Express» введите команду:

```
docker ps
```

Контейнер conference_bot должен иметь статусы «Up» и «healthy».

Внимание! Версии контейнеров conference_bot и messaging должны совпадать, если не совпадают, то следует обновить СК «Express» полностью.

ШАГ 2. ЗАЩИТА CONFERENCE NOTIFIER BOT

Важно:

- если СК «Express» содержит несколько CTS, то данную процедуру следует выполнить на каждом CTS, который будет взаимодействовать с express-core-service;
- если СК «Express» содержит один объединенный ETS, то данную процедуру следует выполнить только на нем.

Для защиты Conference Notifier Bot выполните установку bearer token:

1. Убедитесь, что установлена СК «Express» не ниже версии 3.8. Если ниже, то выполните обновление СК «Express» до последней версии.
2. Убедитесь, что версия дистрибутива «Плагин Outlook» (серверная и клиентская) не ниже 1.2.0.0.

   Важно! Введенные учетные данные будут использоваться при настройке Outlook–плагина.

3. На CTS-сервере добавьте в файл settings.yaml[2] в секцию conference_bot_env_override: следующий код:

   ```
   API_AUTH_METHOD: BEARER_TOKEN
   BEARER_TOKEN: 3bDewf52b3268sdg59f1f7fff33w01dd3c0431
   ```

   После чего секция conference_bot_env_override должна принять вид:

   ```
   conference_bot_env_override:
     API_ENABLED: true
     API_AUTH_METHOD: BEARER_TOKEN
     BEARER_TOKEN: 3bDewf52b3268sdg59f1f7fff33w01dd3c0431
   ```

   В случае использования разделенной серверной архитектуры изменения вносятся только в файл на сервер Back CTS.

   Значение токена придумайте самостоятельно или сгенерируйте любым сторонним ПО (он должен быть не короче 40 символов, содержать большие и маленькие латинские буквы и цифры).

4. Сохраните файл и запустите из /OPT/EXPRESS:

   ```
   dpl -d conference_bot
   ```

5. Проверьте недоступность чат-бота с помощью запроса без bearer token через CURL, например:

[2] Предполагаемый путь /opt/express/settings.yaml

```
curl -X POST -H "Content-Type: application/json" -d
'{"name":"test_plugin_001","members":["user001@mydomain.com","
user002@mydomain.com"],"admins":["user001@mydomain.com
"],"creator":"user001@mydomain.com","start_at":"2023-12-
21T10:10:00.0Z","end_at":"2023-12-
21T11:10:00.0Z","link":{"link_type":"public","access_code":null}}'
https://my-CTS-domain.com/api/v1/conference_bot/conferences/
```

Примечание. Красным цветом отмечены параметры, которые требуется актуализировать под конкретный запрос.

Если чат-бот без bearer token недоступен (это так и должно быть), CURL выведет:

```
Пустую строку
```

6. Проверьте доступность чат-бота с помощью запроса с bearer token через CURL, например:

```
curl -X POST -H "Content-Type: application/json" -H "Authorization:
Bearer 3bfef52b32685srdsrhderhFGd1f73301dd3c0431" -d
'{"name":"test_plugin_001","members":["user001@mydomain.com","
user002@mydomain.com"],"admins":["user001@mydomain.com
"],"creator":"user001@mydomain.com","start_at":"2023-12-
21T10:10:00.0Z","end_at":"2023-12-
21T11:10:00.0Z","link":{"link_type":"public","access_code":null}}'
https://my-CTS-domain.com/api/v1/conference_bot/conferences/
```

Примечание. Красным цветом отмечены параметры, которые требуется актуализировать под конкретный запрос.

Если чат-бот доступен, CURL выведет:

```
"status":"ok"
```

7. В веб-интерфейсе администратора CTS-сервера перейдите в раздел «Звонки → Конференции».
8. В поле поиска введите test_plugin_001 (был задан в п. 5) и нажмите кнопку «Поиск».

   Если в таблице будет выведена конференция с именем test_plugin_001, то чат-бот настроен правильно.

Глава 4

УСТАНОВКА OUTLOOK-ПЛАГИНА

Для установки Outlook-плагина выполните шаги, описанные ниже.

ШАГ 1. ОБНОВЛЕНИЕ СЕРВЕРНОЙ ЧАСТИ (ОПЦИОНАЛЬНО)

Шаг выполняется, если сервис уже был развернут в рамках предыдущей версии плагина.

1. Загрузите новые версии image-образов сервисов express-meeting-core-service и express-meeting-template-service в docker.
2. В папках, где лежат файлы форматов .env и .yml, запустите команду:

   ```
   docker-compose --env-file ./.env -f ./docker-compose.yml down
   ```

3. Из новой версии дистрибутива возьмите новые файлы форматов .env и .yml и укажите в них ваши параметры и новые image-образы.
4. Замените старые файлы форматов .env и .yml на новые в соответствующей папке на сервере Docker .
5. Выполните Шаг 2 текущей главы для обновления структуры БД.
6. Запустите из папки docker контейнеры:

   ```
   docker-compose --env-file ./.env -f ./docker-compose.yml up -d
   ```

ШАГ 2. ПОДГОТОВКА БД

Чтобы развернуть БД, в которой хранятся шаблоны блока приглашения в конференцию при создании приглашения в письме Outlook:

1. Перейдите на windows-машину, на которой есть доступ к серверу PostgreSQL. На машине должен быть установлен .NET Desktop Runtime 7.0 (https://dotnet.microsoft.com/en-us/download/dotnet/7.0).
2. Перенесите на машину папку tools из дистрибутива.
3. В файле tools\appsettings.json установите строку подключения к БД PostgreSQL в ключе:

   ```
   "Connection":"Host=127.0.0.1;Port=5432;Database=express_meeting_db;
   Username=db_express;Password=Pass1234;Pooling=true;Minimum Pool
   Size=50;Maximum Pool Size=100;Include Error Detail=True;"
   ```

   Примечание. Параметры «Username=db_express» и «Password=Pass1234» должны соответствовать учетным данным, которые были созданы при настройке PostgreSQL.

   Сохраните изменения.

4. Запустите из папки tools в консоли:

   ```
   ExpressMeeting.Tools.TemplatesDB.exe --ef-migrate
   ```

5. Дождитесь завершения работы утилиты.

   БД готова к использованию.

ШАГ 3. РАЗВОРАЧИВАНИЕ В DOCKER EXPRESS-CORE-SERVICE И TEMPLATE-CORE-SERVICE

Сервис разворачивается в докере.

Для разворачивания:

1. Загрузите из папки images-образов сервисов в docker:

   ```
   docker load --input express-meeting-core-service-main.tar
   docker load --input express-meeting-template-service-main.tar
   ```

2. Подготовьте настройки запуска сервисов в файле docker\.env.

   Файл содержит следующие настройки:

   - наименование образа, из которого стартует контейнер с core-service (не нужно менять):
     ```
     EXPRESS_MEETING_CORE_SERVICE_MAIN_IMAGE_NAME=express-meeting-core-service-main
     ```
   - порт, по которому будет доступен core-service:
     ```
     EXPRESS_MEETING_CORE_SERVICE_PORT=7100
     ```
   - наименование образа, из которого стартует контейнер с template-service (не нужно менять):
     ```
     EXPRESS_MEETING_TEMPLATE_SERVICE_MAIN_IMAGE_NAME=express-meeting-template-service-main
     ```
   - порт, по которому будет доступен template-service:
     ```
     EXPRESS_MEETING_TEMPLATE_SERVICE_PORT=7200
     ```
   - IP-адрес, который соответствует DNS-имени сервера, на котором развернуто API чат-бота Conference Notifier Bot (указать реальные данные):
     ```
     EXPRESS_MEETING_EXTRA_HOST=api.bot.express:127.0.0.1
     ```
   - IP-адрес, который соответствует DNS-имени сервера, на котором развернут Keycloak (указать реальные данные):
     ```
     EXPRESS_MEETING_EXTRA_HOST2=keycloak.mycompany:127.0.0.1
     ```
   - имя файла с настройками core-service (не нужно менять):
     ```
     EXPRESS_MEETING_CORE_ENV_FILE=core.env
     ```
   - имя файла с настройками template-service (не нужно менять):
     ```
     EXPRESS_MEETING_TEMPLATE_ENV_FILE=template.env
     ```
   - папка контейнера с логами сервиса (не нужно менять):
     ```
     EXPRESS_MEETING_DOCKER_LOGS_PATH=/public/express-meeting-logs
     ```
   - папка docker-хоста, в которую будут добавляться логи из контейнеров сервисов (при необходимости изменить на нужную):
     ```
     EXPRESS_MEETING_LOGS_PATH=/public/express-outlook/logs
     ```

3. Подготовьте настройки запуска core-service в файле docker\core.env.

   Файл содержит следующие настройки:

   - адрес API чат-бота Conference Notifier Bot (указать реальный адрес, по которому развернуто API):
     ```
     ExpressOptions__Uri=https://api.bot.express/api/v1/conference_bot/conferences/
     ```
   - схема авторизации для API чат-бота Conference Notifier Bot (Принимает значения: Basic – авторизация по логину/паролю; StaticToken – авторизация по статическому токену):
     ```
     ExpressOptions__AuthenticationScheme=StaticToken
     ```
   - токен, который будет использован при обращении к API чат-бота Conference Notifier Bot (используется если в параметре ExpressOptions__AuthenticationScheme стоит значение StaticToken):
     ```
     ExpressOptions__Token=Bearer 3bb326abc5609fed301dd3c0431
     ```
   - логин учетной записи, от имени которой происходит обращение к API чат-бота Conference Notifier Bot (используется если в параметре ExpressOptions__AuthenticationScheme стоит значение Basic):
     ```
     ExpressOptions__UserLogin=admin
     ```
   - пароль учетной записи, от имени которой происходит обращение к API чат-бота Conference Notifier Bot (используется если в параметре ExpressOptions__AuthenticationScheme стоит значение Basic):
     ```
     ExpressOptions__UserPassword=admin
     ```

- тип авторизации для core-service (не нужно менять):
  ```
  AuthorizationOptions__AuthType=keycloak
  ```
- адрес Keycloak (указать реальный):
  ```
  AuthorizationOptions__Configuration__Url=https://keycloak.mycompany
  ```
- realm, который используется для авторизации в Keycloak (указать реальный):
  ```
  AuthorizationOptions__Configuration__Realm=realm
  ```
- значение Client ID, который используется для авторизации в Keycloak (указать реальный):
  ```
  AuthorizationOptions__Configuration__ServiceClientId=client
  ```
- значение Client Secret, который используется для авторизации в Keycloak (указать реальный):
  ```
  AuthorizationOptions__Configuration__ServiceClientSecret=secret
  ```
- строка подключения к БД PostgreSQL (это БД, в которой хранятся шаблоны блока приглашения в конференцию при создании приглашения в письме Outlook):
  ```
  DatabaseOptions__Connection=Host=127.0.0.1;Port=5432;Database=express_meeting_db;Username=db_express;Password=Pass1234;Pooling=true;Minimum Pool Size=50;Maximum Pool Size=100;Include Error Detail=True;
  ```

  Примечание. Параметры «Username=db_express» и «Password=Pass1234» должны соответствовать учетным данным, которые были созданы при настройке PostgreSQL.
- папка в контейнере core-service, в которую будут сохраняться логи (не нужно менять):
  ```
  Serilog__WriteTo__1__Args__configureLogger__WriteTo__0__Args__path=/public/express-meeting-logs/coreservise.log
  ```
- папка в контейнере core-service, в которую будут сохраняться логи, полученные с клиентов (не нужно менять):
  ```
  Serilog__WriteTo__2__Args__configureLogger__WriteTo__0__Args__path=/public/express-meeting-logs/client/plugin.log
  ```
- среда выполнения сервиса (не нужно менять):
  ```
  ASPNETCORE_ENVIRONMENT=Production
  ```

4. Подготовка настроек запуска template-service в файле docker\template.env.

   Файл содержит следующие настройки:

   - строку подключения к БД PostgreSQL (это БД, в которой хранятся шаблоны блока приглашения в конференцию при создании приглашения в письме Outlook):
     ```
     DatabaseOptions__Connection=Host=127.0.0.1;Port=5432;Database=express_meeting_db;Username=db_express;Password=Pass1234;Pooling=true;Minimum Pool Size=50;Maximum Pool Size=100;Include Error Detail=True;
     ```

     Примечание. Параметры «Username=db_express» и «Password=Pass1234» должны соответствовать учетным данным, которые были созданы при настройке PostgreSQL.
   - произвольный набор символов, который используется для генерации ключа, которым будет подписан JWT-токен доступа к сервису:
     ```
     JWTOptions__Key=JFRINXV0LGIhLXlEVzlYdXlae316UU0zJzR9WVUtIWQ
     ```
   - значение поля «Issuer» при генерации и проверке JWT-токена:
     ```
     JWTOptions__Issuer=http://127.0.0.1
     ```
   - значение поля «Audience» при генерации и проверке JWT-токена:

```
JWTOptions__Audience=http://127.0.0.1
```

- папку в контейнере template-service, в которую будут сохраняться логи (не нужно менять):

```
Serilog__WriteTo__1__Args__path=/public/express-meeting-logs/templateservise.log
```

- среда выполнения сервиса (не нужно менять):

```
ASPNETCORE_ENVIRONMENT=Production
```

5. Запустите из папки docker-контейнеры:

```
docker-compose --env-file ./.env -f ./docker-compose.yml up -d
```

После запуска контейнеров, если в файле docker\core.env в ключах ExpressOptions__Uri, AuthorizationOptions__Configuration__Url были указаны https-адреса, добавьте в контейнер express-meeting-core-service-main root-сертификаты для соответствующих https-соединений.

После запуска template-service контейнера при переходе на обслуживаемый этим контейнером адрес по пути /front (пример: http://localhost:7200/front) в браузере должна открыться страница логина в систему администрирования шаблонов плагина.

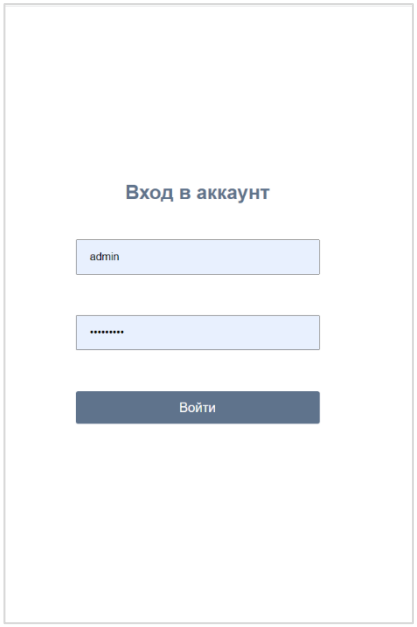

Рисунок 5

Примечание. Первый вход осуществляется под учетной записью admin с пустым паролем

ШАГ 4. РАЗВОРАЧИВАНИЕ OUTLOOK-ПЛАГИНА НА КЛИЕНТСКИХ ПК

1. Запустите инсталлятор ExpressMeeting_v_2.1.0.0.msi на машине, на которой установлен Outlook.
2. После инсталляции в файле ExpressMeeting.dll.config[3] установите необходимые настройки для работы плагина:
   - адрес core-service (серверной части плагина) в ключе (замените mycompany на ip/dns-имя машины docker-хоста, где развернут core-service):

     ```
     <add key="ExpressMeetingUrl" value= "http://mycompany:
     7100/api/" />
     ```

     Примечание. Параметр можно автоматически проинициализировать при установке msi-пакета. Для этого в командной строке необходимо передать параметр ExpressMeetingUrl.

     Пример: ExpressMeeting_v_2.1.0.0.msi ExpressMeetingUrl="http://mycompany:7100/api/"

   - адрес Keycloak точки входа для методов oauth2-авторизации в ключе:

     ```
     <add key="KeycloakUrl"
     value="https://keycloak.mycompany/realms/realm-
     name/protocol/openid-connect/" />
     ```

     Примечание. Параметр можно автоматически проинициализировать при установке msi-пакета. Для этого в командной строке необходимо передать параметр KeycloakUrl.

     Пример: ExpressMeeting_v_2.1.0.0.msi KeycloakUrl="https://keycloak.mycompany/realms/realm-name/protocol/openid-connect/"

   - Значение Client ID клиента, который используется для авторизации в Keycloak:

     ```
     <add key="ClientId" value="client" />
     ```

     Примечание. Параметр можно автоматически проинициализировать при установке msi-пакета. Для этого в командной строке необходимо передать параметр ClientId.

     Пример: ExpressMeeting_v_2.1.0.0.msi ClientId="client"

   - Таймаут в секундах на установление соединения при обращении к Keycloak:

     ```
     <add key="Timeout" value="120"/>
     ```

     Примечание. Параметр можно автоматически проинициализировать при установке msi-пакета. Для этого в командной строке необходимо передать параметр Timeout.

     Пример: ExpressMeeting_v_2.1.0.0.msi Timeout="120"

   - При необходимости установите режим запрета выбора типа конференции. Выпадающий список с выбором типа (Общее/Корпоративное/Доверенное) будет скрыт в настройках встречи. Всегда будут создаваться общие встречи. Для этого добавьте в секцию configuration/appSettings параметр:

     ```
     <add key="DenyLinkType" value="true"/>
     ```

[3] Предполагаемый путь c:\Program Files\Express\ExpressMeeting\ExpressMeeting.dll.config

Примечание. Параметр можно автоматически проинициализировать при установке msi-пакета. Для этого в командной строке необходимо передать параметр DenyLinkType.

Пример: ExpressMeeting_v_2.1.0.0.msi DenyLinkType=true

- При необходимости установите тип создаваемой по умолчанию конференции (возможные значения: Public, Trusts, Corporate). По кнопке создать конференцию будут создаваться встречи указанного типа. Для этого добавьте в секцию configuration/appSettings параметр:

```
<add key="DefaultLinkType" value="Corporate"/>
```

Примечание. Параметр можно автоматически проинициализировать при установке msi-пакета. Для этого в командной строке необходимо передать параметр DefaultLinkType.

Пример: ExpressMeeting_v_2.1.0.0.msi DefaultLinkType=Corporate

3. В файле serilogSettings.json[4] установите адрес core-service (серверной части плагина) в ключе (замените mycompany на IP/DNS-имя машины docker-хоста, где развернут core-service):

```
Serilog.WriteTo[Name=Telemetry].Args.telemetryUrl="http://mycompany
:7100/api/log/json"
```

Примечание. Если при установке msi-пакета в командной строке был передан параметр ExpressMeetingUrl, то этот параметр проинициализируется автоматически.

4. Плагин поддерживает разные уровни логирования. Для изменения уровня логирования необходимо в файле serilogSettings.json[5] установить значение ключа Serilog.MinimumLevel.Default на требуемый уровень. Возможные значения уровня логирования: Verbose, Debug, Information, Warning, Error, Fatal. Где Verbose – логировать все, Debug – логировать Debug и выше, Information – логировать Information и выше и т.п.

Примечание. Параметр можно автоматически проинициализировать при установке msi-пакета. Для этого в командной строке необходимо передать параметр LogLevel.

Пример: ExpressMeeting_v_2.1.0.0.msi LogLevel=Information

[4] Предполагаемый путь c:\Program Files\Express\ExpressMeeting\serilogSettings.json

[5] Предполагаемый путь c:\Program Files\Express\ExpressMeeting\serilogSettings.json

Глава 5

ДИАГНОСТИКА НЕИСПРАВНОСТЕЙ ПЛАГИНА OUTLOOK

Данный раздел содержит рекомендации для проведения корректной диагностики.

Внимание! Раздел не содержит указаний по исправлению выявленных неисправностей.

Принцип работы плагина Outlook:

1. При нажатии кнопки  в Outlook формируется и отправляется запрос на сервер Keycloak.
2. На Keycloak-сервере производится аутентификация и авторизация.
3. В случае успешного прохождения аутентификации и авторизации, Keycloak передает Token в express-meeting-core-service.
4. Сервис express-meeting-core-service передает в плагин Outlook шаблон письма из БД и отравляет запрос на создание шаблона будущей конференции в чат-бот.
5. Conference Notifier Bot создает шаблон будущей конференции в СК «Express».
6. Клиентская часть плагина Outlook создает письмо в Outlook из шаблона, вставив в него ссылку на будущую конференцию.
7. Пользователь указывает участников, дату, время и название будущей конференции, заполняя соответствующие поля в письме Outlook.
8. Когда пользователь нажимает кнопку «Отправить», Outlook-плагин формирует и отправляет запрос на изменение созданной раннее конференции в express-meeting-core-service с учетом параметров, заполненных пользователем.
9. В случае успешного прохождения аутентификации и авторизации express-meeting-core-service отравляет запрос на изменение шаблона будущей конференции в чат-бот с учетом параметров, заполненных пользователем.
10. Чат-бот меняет параметры конференции в СК «Express» на актуальные.

Если описанный выше принцип работы плагина не выполняется, то используйте следующие шаги для диагностики проблемы:

ШАГ 1. ПРОВЕРКА РАБОТОСПОСОБНОСТИ CONFERENCE NOTIFIER BOT

В первую очередь проверьте работоспособность Conference Notifier Bot в серверной части СК «Express», так как если чат-бот не принимает или не создает конференции в СК «Express», то дальнейшие действия по настройке и диагностике других компонентов системы будут бесполезны.

Для проверки работоспособности Conference Notifier Bot:

1. Зайдите по SSH на CTS-сервер СК «Express».
2. Выполните запрос к Conference Notifier Bot через программу «Curl».
3. Выполните запрос к Conference Notifier Bot через чат мобильного приложения или веб/десктоп-версию приложения СК «Express»:
   - если запрос к Conference Notifier Bot через программу «Curl» и чат в eXpress выполнен успешно, переходите к следующему шагу;
   - если запрос к Conference Notifier Bot через программу «Curl» или чат в eXpress не был выполнен, тогда нужно восстановить работоспособность (проверить настройки) Conference Notifier Bot и перейти к следующему шагу.

   Действия по проверке работоспособности Conference Notifier Bot через программу Curl описаны в Главе 4 «Установка Outlook-плагина» настоящей инструкции:

- если безопасность чат-бота не настраивалась, см. пп. 5—7 раздела «Шаг 1. Включение API Conference Notifier Bot без пароля»;
- если безопасность чат-бота настраивалась, см. пп. 6—8 раздела «Шаг 2. Защита Conference Notifier Bot».

ШАГ 2. ПРОВЕРКА ДОСТУПНОСТИ CONFERENCE NOTIFIER BOT С СЕРВЕРА DOCKER

Далее рекомендуется проверить доступность Conference Notifier Bot с сервера Docker.

Для проверки доступности Conference Notifier Bot с сервера Docker:

1. Зайдите по SSH или другим способом на сервер Docker, где установлено приложение express-core-service.
2. Проведите диагностику аналогично Шагу 1 Для проверки работоспособности Conference Notifier Bot – выше.

ШАГ 3. ПРОВЕРКА ДОСТУПНОСТИ ПРИЛОЖЕНИЯ EXPRESS-MEETING-CORE-SERVICE С КЛИЕНТСКОЙ РАБОЧЕЙ СТАНЦИИ

Далее рекомендуется проверить доступность приложения express-core-service на сервере Docker с клиентской рабочей станции.

Для проверки доступности сервера Docker с клиентской рабочей станции:

1. Подключитесь через RDP (или другим способом) к клиентской рабочей станции, на которой установлены приложение Microsoft Outlook и плагин для Outlook.

   В PowerShell ISE запустите диагностический скрипт из Диагностический скрипт № 1 и отредактируйте значения переменных: учетную запись пользователя Outlook, ссылку на Docker и токен, полученный из Keycloak. Токен Keycloak можно получить, запросив его у Администратора KeyCloak Заказчика.
2. Если скрипт отработал с ошибками, проанализируйте и устраните их причины. Вероятные причины проблемы в настройках Docker и приложения express-core-service: отсутствует сетевой доступ, проблемы аутентификации и авторизации. Перезапускайте скрипт, пока он не отработает без ошибок.
3. Если скрипт отработал без ошибок и выдал ID конференции, которую можно найти в консоли администрирования CTS-сервера СК «Express», тогда считается, что Conference Notifier Bot доступен с сервера Docker.

   Переходите к следующему шагу.

ШАГ 4. ОБЩАЯ ДИАГНОСТИКА ПЛАГИНА OUTLOOK

Если предыдущие 3 шага пройдены успешно, но плагин Outlook все равно не работает, проверьте корректность настроек в конфигурационном файле клиентского ПО (см. раздел «Шаг 3. Разворачивание в Docker express-core-service и template-core-» настоящей инструкции»).

Если конфигурационный файл клиентского ПО корректен, изучите файлы логов из клиентских приложений, описанные в разделе «Шаг 3. Разворачивание в Docker express-core-service и template-core-» настоящей инструкции, и серверные файлы логов, описанные в разделе «Шаг 2. ». Устраните причины ошибок.

Если все перечисленные выше шаги не помогли, обратитесь в техническую поддержку производителя за помощью.

Приложение 1

ДИАГНОСТИЧЕСКИЙ СКРИПТ № 1

Приложение не содержит указаний по исправлению выявленных неисправностей, а разъясняет принцип работы скрипта диагностики.

```
$headers = New-Object "System.Collections.Generic.Dictionary[[String],[String]]"
$headers.Add("Content-Type", "application/json")
$headers.Add("Authorization", токен)

# ниже заполнить создателя
$body = @"
{`"Creator`":`"Пользователь_Экспесс@Ваш_домен.ru`",`"LanguageId`":1049,`"isRecurrent`"
:false,`"`$type`":`"MeetingCreateRequest`"}
"@


#  ниже ввести адрес Docker express-core-service  такаяже ссылка должна быть в
настройках клиентской части плагина.

$response = Invoke-RestMethod 'http://Ваш_Docker:Ваш_port/api/meetings' -Method 'POST'
-Headers $headers -Body $body
$response | ConvertTo-Json
```