

Инструкция по установке
ПО «L-TECH SEND NOTIFICATION»

г. Москва
2025 г.

Содержание

1. ТЕРМИНЫ И СОКРАЩЕНИЯ.....	3
2. СИСТЕМНЫЕ ТРЕБОВАНИЯ.....	3
3. ФОРМАТ РАСПРОСТРАНЕНИЯ ПО	3
4. УСТАНОВКА СЕРВЕРНЫХ КОМПОНЕНТОВ, НЕОБХОДИМЫХ ДЛЯ РАБОТЫ ПО	3
4.1. Состав компонентов, необходимых для работы ПО	3
4.2. Установка и настройка PostgreSQL на Linux.....	4
4.3. Установка Docker и Docker Compose.....	5
4.4. Получение Docker Images для создания Docker-контейнера с необходимыми сервисами.....	6
4.5. Настройка и запуск Docker Compose	6
5. Настройка клиентского приложения на Android.....	8

1. Термины и сокращения

ПО – программное обеспечение.

ПАК – программно-аппаратный комплекс.

2. Системные требования

Для корректной работы ПО «L-TECH SEND NOTIFICATION» (далее также - ПО) необходимо:

- Операционная система: Windows 10 (Home/Pro) или выше, Linux (например, Ubuntu).
- Оперативная память: минимум 4 ГБ.
- Место на жестком диске: минимум 10 ГБ свободного места.
- Подключение к сети Интернет для обновлений и получения уведомлений.

3. Формат распространения ПО

Программное обеспечение «L-TECH SEND NOTIFICATION» поставляется в формате on-premise. Поставка может быть выполнена одним из следующих способов:

- Предоставление файлов для самостоятельной установки.
- Установка программного обеспечения сотрудниками правообладателя ПО.

4. Установка серверных компонентов, необходимых для работы ПО

Для установки компонентов на сервер потребуется соединение с глобальной сетью Интернет.

4.1. Состав компонентов, необходимых для работы ПО

Должны быть установлены следующие дополнительные компоненты, необходимые для работы ПО «L-TECH SEND NOTIFICATION»:

- СУБД PostgreSQL.
- Docker Compose.

4.2. Установка и настройка PostgreSQL на Linux

Необходимо обновить список пакетов в терминале Операционной системы Linux. Для этого используется команда:

sudo apt-get update

Необходимо установить пакет PostgreSQL. Для этого используется команда:
sudo apt-get install postgresql

Необходимо запустить PostgreSQL и настроить автозапуск при старте ПО.
для этого используются команды:

sudo systemctl start postgresql
sudo systemctl enable postgresql

Необходимо создать пароль для пользователя postgres (заменить в командах ниже <new-password> на новый пароль). Для этого используются команды:

sudo -u postgres psql
postgres=# \password postgres
Enter new password: <new-password>
postgres=# \q

Необходимо отредактировать файл конфигурации PostgreSQL, который находится по адресу:

/etc/postgresql/<version>/main/postgresql.conf,

где <version> - это версия PostgreSQL, установленная на вашей системе.

Параметр listen_addresses требуется раскомментировать, добавив адреса, на которых PostgreSQL должен слушать входящие подключения. Например, для всех адресов IPv4, необходимо добавить следующую строку:

listen_addresses = '*'

Необходимо отредактировать файл pg_hba.conf, который находится по адресу:

/etc/postgresql/<version>/main/pg_hba.conf,

где <version> - это версия PostgreSQL. В этом файле настраиваются правила аутентификации и авторизации для подключения к базе данных.

Строку необходимо раскомментировать и добавить подключение с любых сетевых адресов следующим образом:

IPv4 local connections:
host all all 0.0.0.0/0 md5

Для того, чтобы активировать сделанные настройки, необходимо перезапустить PostgreSQL с помощью команды:

systemctl restart postgresql

Необходимые для работы базы данных будут созданы автоматически после запуска docker-образов ПО «L-TECH SEND NOTIFICATION».

4.3. Установка Docker и Docker Compose

Для установки Docker потребуется установка вспомогательных пакетов. Они помогут в безопасности, управлении репозиториями и взаимодействии с интернетом:

curl – это инструмент командной строки для передачи данных и работы с URL-адресами;

software-properties-common – этот пакет предоставляет скрипты для управления программным обеспечением и добавления репозитория;

ca-certificates – необходим для безопасной передачи данных и подтверждения подлинности сертификатов;

apt-transport-https – позволяет работать с репозиториями, которые передают данные по протоколу HTTPS.

Эти пакеты можно установить одной командой:

```
sudo apt install curl software-properties-common ca-certificates apt-transport-https -y
```

Для обеспечения безопасности установки Docker необходимо добавить ключ GPG с официального репозитория. Этот ключ гарантирует подлинность загружаемых пакетов:

```
wget -O- https://download.docker.com/linux/ubuntu/gpg | gpg --dearmor | sudo tee /etc/apt/keyrings/docker.gpg > /dev/null
```

Теперь нужно добавить репозиторий Docker для доступа к его последним версиям:

```
echo "deb [arch=amd64 signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu jammy stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

Повторно обновить индексы пакетов:

```
sudo apt update
```

Для запуска процесса установки Docker используйте команду:

```
sudo apt install docker-ce -y
```

Docker установлен, проверьте его работу командой:

```
sudo systemctl status docker
```

Для установки Docker Compose V2 воспользуйтесь командой:

```
docker compose
```

Или для установки автономной версии:

```
curl -SL https://github.com/docker/compose/releases/download/v2.23.3/docker-compose-linux-x86_64 -o /usr/local/bin/docker-compose
```

4.4. Получение Docker Images для создания Docker-контейнера с необходимыми сервисами

При покупке ПО «L-TECH SEND NOTIFICATION», заказчику предоставляется ссылка для скачивания архива с docker-образами, необходимыми для создания Docker-контейнеров с необходимыми сервисами.

В архиве содержатся:

- docker-compose.yml - файл конфигурации, который задает правила для создания Docker - контейнера с необходимыми сервисами (менять не нужно);
- Конфигурационные файлы с переменными окружения docker.env (требуется скорректировать пароль подключения к PostgreSQL указанный при установке PostgreSQL).
- Дистрибутив клиентского ПО

Необходимо создать папку (например «LSN») на сервере куда будут разархивироваться Docker образы. Для этого можно использовать команды:

```
cd ../  
mkdir lsn
```

Далее в созданную папку необходимо переместить все файлы из скачанного архива.

4.5. Настройка и запуск Docker Compose

После выполнения «Получение Docker Images для создания Docker-контейнера с необходимыми сервисами» будут созданы Docker-контейнеры для следующих сервисов:

- **web** - сервис lsn;
- **sidekiq** - планировщик задач;
- **redis** - база данных для sidekiq;
- **db** - PostgreSQL сервис.

Отредактируйте конфигурационные файлы с переменными окружения docker.env указав url для доступа к PostgreSQL.

Далее необходимо выполнить команду для запуска контейнера с необходимыми сервисами:

```
docker compose run -d
```

1. **chown user/docker ~/lsn/public**
2. **chmod 775 ~/lsn/public**

Возможно настроить автоматический запуск Docker-контейнеров при запуске ПО. Для запуска в автоматическом режиме при старте сервера необходимо использовать менеджеры процессов такие как **upstart**, **systemd** или **supervisor**.

После успешной установки должно отобразиться окно авторизации:

Выход из системы выполнен.

LSN Войти

Эл. почта*

Пароль*

☐ Запомнить меня

Войти

[Забыли пароль?](#)

Далее необходимо ввести логин и пароль, которые пользователь получает после приобретения ПО.

После ввода логина и пароля будет отображен интерфейс ПО:

LSN	Опять неток	Отчеты	API канал	Настройки	Отправка	Администраторы	admin@example.com	Выход
Актив								
Сырой поток								
☐	ID	Номер телефона	Токен	Заголовок	Текст	Статус	Тип сообщения	Премия отправки Image Doc Время создания
☐	49c77b2-205e-4524-b96e-16c0e0da3109		e130dc38ff78cc3528ba27c42a9f537...	Клиент	ваши_ваши_ваши	Успешно доставлено	Пуш	20 марта 2025, 11:56
☐	90c0c3d7-9245-4438-880c-592029a71e4f		e130dc38ff78cc3528ba27c42a9f537...	34523451234	12343241234	Успешно доставлено	Пуш	20 марта 2025, 11:55
☐	18e4c399-8111-860a-8052-901649e84191		e130dc38ff78cc3528ba27c42a9f537...	Поступило	Увед.	Успешно доставлено	Пуш	20 марта 2025, 11:40
☐	77a775da-6092-4426-90c5-c836fa272118		c4e0ba18f99237cc0be1d89dc9635...	Клиент	фильмы_фильмы	Успешно доставлено	Пуш	20 марта 2025, 10:44
☐	a5d303e4-8019-4730-892e-49760513049		1b7d849028348908766138491406...	Смита 22222	ваши_ваши_ваши_фильмы_фильмы	Успешно доставлено	Пуш	20 марта 2025, 10:36
☐	1c01852-1a3c-43aa-b0dc-83021e4909a		c4e0ba18f99237cc0be1d89dc9635...	ваши_ваши_ваши	53452345435	Успешно доставлено	Пуш	20 марта 2025, 10:27
☐	6a6cc9d7-5079-8b0c-819e-2030ee-4888a5		d0cda40167c0be78a15296ba22998...	23412323213	1234512343124	Успешно доставлено	Пуш	20 марта 2025, 10:22
☐	0a3cc63-5874-48ed-98cd-73024095dc84		d0cda40167c0be78a15296ba22998...	Test	Warning!!	Успешно доставлено	Пуш	20 марта 2025, 09:54
☐	a1a007b5-4eb4-472c-855a-811c15598c9c	123	123	123	123	Отправлено	Пуш	19 марта 2025, 11:54
☐	30a0e069-3aaf-49a3-a96c-43c0351c1e49		2a8f1555b0a4eeebc75603894eb0dc...	club	test	Успешно доставлено	Пуш	19 марта 2025, 11:33
☐	9e84ebec-3452-4818-af1c-55e070700a3		1b7d849028348908766138491406...	Тестовый push Сенту	Смита push	Успешно доставлено	Пуш	19 марта 2025, 11:01
☐	5af9a173-0360-4729-899a-b718a8b15981		1b7d849028348908766138491406...	Тестовый push Сенту	Смита push	Успешно доставлено	Пуш	19 марта 2025, 10:58

5. Настройка приложения пользователя на Android

Загрузка и импорт библиотеки в проект

После приобретения ПО «L-TECH SEND NOTIFICATION» заказчику также предоставляется ссылка для скачивания библиотеки, необходимой для дальнейшей интеграции в проект заказчика (приложение пользователя), в формате архива.

Распакуйте архив в папку вашего проекта, например: `./.../MyProject/lib/`. Итоговый путь должен выглядеть следующим образом: `./.../MyProject/lib/ltech-messaging-service`.

Откройте файл **settings.gradle**

Добавьте новый артефакт в **dependencyResolutionManagement**, пример:

```
dependencyResolutionManagement {  
    ...  
    repositories {  
        ...  
        maven { url = uri("${rootProject.projectDir}/lib/ltech-messaging-service") }  
    }  
}
```

Добавьте имплементацию библиотеки в модуль приложения, пример:

```
dependencies {  
    ...  
    implementation("com.ltech:lms:1.0.0")  
    ...  
}
```

Синхронизируйте проект.

5.2. Настройка Push-уведомлений

Фоновые сервисы осуществляют мониторинг наличия непрочитанных уведомлений, и в случае их обнаружения отправляют их в слушатель **LsnMessagingListener**.

LsnMessagingListener реализует единственный метод — **onMessageReceive(message: Message)**, который вызывается при получении нового пуш-уведомления.

Для получения push-уведомлений необходимо создать соответствующий слушатель, например:

```
class MessagingListener(
    private val context: Context
): LsnMessagingListener {

    companion object {
        const val CHANNEL_ID = "02326e10"
        const val CHANNEL_NAME = "Системные"
    }

    private val notificationManager = NotificationManagerCompat.from(context)

    init {
        createNotificationChannel()
    }

    override fun onMessageReceive(message: Message) {
        val notification = NotificationCompat.Builder(context, CHANNEL_ID)
            .setContentTitle(message.title)
            .setContentText(message.message)
            .setAutoCancel(true)
            .setSmallIcon(R.drawable.pic_avatar_1)
            .build()

        notificationManager.notify(-41513, notification)
    }

    private fun createNotificationChannel() {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            notificationManager.createNotificationChannel(
                NotificationChannel(CHANNEL_ID, CHANNEL_NAME,
                    NotificationManager.IMPORTANCE_MIN)
            )
        }
    }
}
```

5.3. Инициализация библиотеки

Создайте Application класс, пример:

```
class App: Application() {  
  
    override fun onCreate() {  
        super.onCreate()  
        ...  
    }  
}
```

Укажите созданный Application класс в манифесте приложения, пример:

```
<application  
    android:name=".App"  
    ....  
>
```

Для корректной инициализации библиотеки требуется предоставить **ApplicationContext** и **LsnAppConfiguration**. В случае если будет использован контекст, отличный от Application, библиотека выдаст исключение, поскольку для её корректной работы требуется именно **ApplicationContext**.

LsnAppConfiguration нужен для настройки библиотеки, создать его можно используя **LsnAppConfiguration.Builder()** передав обязательные параметры:

- host - Хост для доступа к сервису: <https://lsn-dev.l-tech.ru/>.

Важно, что хост должен заканчиваться символом /

- listener - Слушатель для приема новых уведомлений. Будет вызван в случае, если есть новые уведомления.

- notificationConfiguration - Конфигурация уведомления, которое будет показано пользователю в случае, если работа в фоне выполняется слишком долго, например при низкой скорости интернета. Это системное ограничение требующее, чтобы фоновые сервисы показывали Push-Уведомления для продолжения своей работы.

Пример конфигурации:

```
val config = LsnAppConfiguration.Builder(  
    host = " https://lsn-dev.l-tech.ru/",  
    listener = MessagingListener(this),  
    notificationConfiguration = SyncNotificationConfiguration(  
        ...  
    )  
)
```

```

        icon = R.drawable.ic_notification,
        title = context.getString(R.string.sync_notification_title),
        channel = SyncNotificationConfiguration.ChannelConfig(
            id = "02326e10",
            name = context.getString(R.string.sync_notification_channel_name)
        )
    ),
).build()

```

LsnAppConfiguration.Builder так же включает не обязательные настройки:

- `setLogLevel(level: LsnLogLevel)` - Устанавливает уровень логирования в проекте, имеет 3 состояния: `NONE` - Логи не используются, `NOTIFICATION` - Логируются только приходящие уведомления, `ALL` - Логируются все действия.

`setWakeUpPeriod(period: LsnTimeUnit)` - Устанавливает период в котором приложение будет просыпаться в фоне чтобы проверить наличие уведомлений. Стандартный и минимальный тариф - 15 минут.

Пример установки периода:

```

.setWakeUpPeriod(
    period = LsnTimeUnit(15, TimeUnit.MINUTES)
)

```

Полная инициализация библиотеки должна выглядеть так:

```

class App: Application() {

    override fun onCreate() {
        super.onCreate()
        ...
        initializeLsn()
    }

    private fun initializeLsn() {
        val config = LsnAppConfiguration.Builder(
            host = "https://myproject/",
            listener = MessagingListener(this),
            notificationConfiguration = SyncNotificationConfiguration(
                icon = R.drawable.ic_notification,
                title = context.getString(R.string.sync_notification_title),
                channel = SyncNotificationConfiguration.ChannelConfig(

```

```

        id = "02326e10",
        name = context.getString(R.string.sync_notification_channel_name)
    )
),
).build()
Lsn.initialize(
    context = this,
    configuration = config
)
}
}

```

Теперь все готово к получению Push-Уведомлений.

На рисунке ниже показан пример отображения Push-Уведомлений в приложении пользователя:

