Приложение помогающее планировать обслуживание кондиционеров и вести учет доходов/расходов.

# Пользователи

Всего существует 3 группы пользователей:

* Группа 1 -> доступ к заработной плате сотрудников, финансовым операциям и всем данным из базы (наличие кондиционеров, время ремонта, план ремонта и т/д) -> Administrator
* Группа 2 -> доступ к финансовым операциям и всем данным из базы (наличие кондиционеров, время ремонта, план ремонта и т/д) -> Manager
* Группа 3 -> доступ ко всем данным из базы (наличие кондиционеров, время ремонта, план ремонта и т/д) ->Еmployee

Проверка доступа осуществляется при помощи JWT-Token (Spring Boot)

Пользователи имеют уникальный e-mail (или телефон), при помощи которого происходит подтверждение регистрации, и на который будут приходить планы и задачи (например – какой сотрудник в каком цехе какой кондиционер ремонтирует)

Пользователей может добавлять Administrator или Manager

Менеджера может добавлять Administrator

Administrator регистрируется самостоятельно, используя для этого «секретный код» который ему выдаст SuperAdministrator (он же программист)

## Регистрация пользователей:

### SuperAdministrator

1. SuperAdministrator получает от программиста секретный код.
2. SuperAdministrator входит на страницу регистрации и вводит свои данные ( имя, фамилия, логин, пароль, e-mail)
3. Данные SuperAdministrator сохраняются в базу данных (в колонке «подтвержденный e-mail) по умолчанию стоит false а в колонке e-mail-token стоит токен для подтверждения.
4. На введенный SuperAdministrator электронный адрес отправляется письмо с ссылкой подтверждения почты и токеном
5. SuperAdministrator нажимает на ссылку и таким образом подтверждает свою почту.
6. SuperAdministrator может заходить в приложение и работать

### Manager

1. SuperAdministrator добавляет данные менеджера в базу данных

(имя, фамилия, логин, e-mail)

1. Данные Manager сохраняются в базу данных (в колонке «подтвержденный e-mail) по умолчанию стоит false а в колонке e-mail-token стоит токен для подтверждения.
2. На введенный Manager электронный адрес отправляется письмо с ссылкой подтверждения почты и токеном
3. Manager нажимает на ссылку и таким образом подтверждает свою почту.
4. Manager заходит в приложение и нажимает кнопку «забыл пароль???»
5. Manager вводит свой пароль
6. На электронный адрес менеджера отправляется письмо, нажав на которое менеджер опадает на страницу смены пароля.
7. Менеджер вводит свой пароль (который сохраняется в базу данных)
8. Manager может заходить в приложение и работать

### Еmployee

1. SuperAdministrator добавляет данные менеджера в базу данных

( имя, фамилия, логин, e-mail)

1. Данные Еmployee сохраняются в базу данных (в колонке «подтвержденный e-mail) по умолчанию стоит false а в колонке e-mail-token стоит токен для подтверждения.
2. На введенный Еmployee электронный адрес отправляется письмо с ссылкой подтверждения почты и токеном
3. м нажимает на ссылку и таким образом подтверждает свою почту.
4. Еmployee заходит в приложение и нажимает кнопку «забыл пароль???»
5. Еmployee вводит свой пароль
6. На электронный адрес менеджера отправляется письмо, нажав на которое менеджер опадает на страницу смены пароля.
7. Менеджер вводит свой пароль (который сохраняется в базу данных)
8. Еmployee может заходить в приложение и работать

# Тип обслуживания

## Создание типа обслуживания

Пользователь (Администратор или менеджер) создает новый кондиционер создает тип обслуживания.

### API: create\_type\_maintenance:

### Link: …/maintenance

### Response: POST

#### DTO:

* String uuidTypeMaintenance // уникальное id ТО;
* String nameMaintenance//название ТО// Required;
* String peopleHours// сколько человеко/часов необходимо на проведение ТО// Required ;
* Boolean deleted = false// удаление ТО (по умолчанию - false)

#### JSON:

{

"nameMaintenance":"TO1",

"peopleHours":"10"

}

### Request:

{

"uuidTypeMaintenance": "f3410e61-6060-48d2-8ecd-2b7236e9a348",

"nameMaintenance": "TO2",

"peopleHours": "10",

"deleted": false

}

### Примечание:

## Получение типа обслуживания по его id

Пользователь (Администратор или менеджер) может получить ТО по его уникальному Uuid

### API: get\_type\_maintenance\_by\_id:

### Link: …/maintenance/ {typeMaintenanceUuid}

### Response: GET

### Request:

{

"uuidTypeMaintenance": "f3410e61-6060-48d2-8ecd-2b7236e9a348",

"nameMaintenance": "TO2",

"peopleHours": "10",

"deleted": false

}

### Примечание:

Если кондиционер удален (то есть deleted= true), вернется сообщение, что такого кондиционера нет в БД

## Получение всех типов обслуживания

Пользователь (Администратор или менеджер) может получить все ТО из базы данных

### API: get\_all\_type\_maintenance:

### Link: …/maintenance/

### Response: GET

### Request:

[

{

"uuidTypeMaintenance": "53df145b-d281-4b59-b4a6-2e1083f12e22",

"nameMaintenance": "TO1",

"peopleHours": "10",

"deleted": true

},

{

"uuidTypeMaintenance": "f3410e61-6060-48d2-8ecd-2b7236e9a348",

"nameMaintenance": "TO2",

"peopleHours": "10",

"deleted": false

}

]

### Примечание:

## Удаление типа обслуживания по его id

Пользователь (Администратор или менеджер) может удалить ТО по его уникальному Uuid

### API: delete\_type\_maintenance\_by\_id:

### Link: …/maintenance/ {typeMaintenanceUuid}

### Response: DELETE

### Request:

{

"uuidTypeMaintenance": "53df145b-d281-4b59-b4a6-2e1083f12e22",

"nameMaintenance": "TO1",

"peopleHours": "10",

"deleted": true

}

### Примечание:

# Создание кондиционера

Пользователь (Администратор или менеджер) создает новый кондиционер.

Когда пользователь выбирает API create\_conditioner, автоматически вызывается API [get\_all\_type\_maintenance](#_Получение_всех_типов), который предоставляет пользователю возможность выбрать один из существующих типов обслуживания.

Если пользователь не хочет на данном этапе присваивать тип обслуживания конкретному кондиционеру – он может оставить это поле пустым ([JSON\_create\_conditioner\_without\_type\_maintenance](#_Создание_кондиционера_без))

## Создание кондиционера

### API: create\_conditioner: DEPRECATED([link](#_Примечание:_Не_актуально.))

### Link: …/conditioner

### Response: POST

#### DTO:

* String uuidConditioner //// уникальное id кондиционера
* String nameConditioner;// название кондиционера// Required
* String inventoryNumber;//уникальный инвентарный номер кондиционера// Required
* String place;//место, где установлен кондиционер// Required
* Date startDate; //дата, когда кондиционер начал работать
* List<TypeMaintenanceForDto> maintenance = new ArrayList<>();//список ТО кондиционера
* Integer workedHours;// количество часов, которое кондиционер должен проработать до ТО // Required
* Boolean deleted;// // удаление ТО (по умолчанию - false)

#### JSON:

{

"nameConditioner":"conditioner",

"inventoryNumber":"CON120",

"place":"KMK",

"workedHours":120,

"deleted":false,

"maintenance": [

{

"uuidTypeMaintenance": "f3410e61-6060-48d2-8ecd-2b7236e9a348",

"nameMaintenance": "ТО2",

"peopleHours": "14",

"deleted": false

}

]

}

### Request:

{

"uuidConditioner": "6dda7926-00bf-4c55-9029-e59ac9af05af",

"nameConditioner": "conditioner",

"inventoryNumber": "CON120",

"place": "KMK",

"startDate": null,

"maintenance": [

{

"uuidTypeMaintenance": "f3410e61-6060-48d2-8ecd-2b7236e9a348",

"nameMaintenance": "ТО2",

"peopleHours": "14",

"deleted": false

}

],

"workedHours": 120,

"deleted": false

}

### Примечание: Не актуально. ТО будет назначаться уже готовым кондиционерам!!!

## Создание кондиционера без ТО

### API: create\_conditioner:

### Link: …/conditioner

### Response: POST

#### DTO:

* String uuidConditioner //// уникальное id кондиционера
* String nameConditioner;// название кондиционера// Required
* String inventoryNumber;//уникальный инвентарный номер кондиционера// Required
* String place;//место, где установлен кондиционер// Required
* Date startDate; //дата, когда кондиционер начал работать
* Integer workedHours;// количество часов, которое кондиционер должен проработать до ТО // Required
* Boolean deleted;// // удаление ТО (по умолчанию - false)

#### JSON:

{

"nameConditioner":"conditioner",

"inventoryNumber":"CON120",

"place":"KMK",

"workedHours":120,

"deleted":false,

"maintenance": [ ]

}

### Request:

{

"uuidConditioner": "6dda7926-00bf-4c55-9029-e59ac9af05af",

"nameConditioner": "conditioner",

"inventoryNumber": "CON120",

"place": "KMK",

"startDate": null,

"maintenance": [ ],

"workedHours": 120,

"deleted": false

}

### Примечание:

## Получение кондиционера по его id

Пользователь (Администратор или менеджер) может получить кондиционер по его уникальному Uuid

API: get\_conditioner\_by\_id:

Link: …/conditioner/{conditionerUuid}

### Response: GET

### Request:

{

"uuidConditioner": "203119ea-3aa9-46b5-a4fd-3d5e4cd3f37a",

"nameConditioner": "cond4",

"inventoryNumber": "number5",

"place": "KMK",

"startDate": "2020-07-26T10:07:37.490+00:00",

"maintenance": [

{

"uuidTypeMaintenance": "a19ad67c-8408-4be1-acfa-e2f11c3a70f5",

"nameMaintenance": "main2",

"peopleHours": "10",

"deleted": false

}

],

"workedHours": 120,

"deleted": false

}

### Примечание:

Если кондиционер удален (то есть deleted= true), вернется сообщение, что такого кондиционера нет в БД

## Получение всех кондиционеров

Пользователь (Администратор или менеджер) может получить все кондиционеры из базы данных

### API: get\_all\_conditioners:

### Link: …/ conditioner /

### Response: GET

### Request:

[

{

"uuidConditioner": "f59d8ecd-b5f0-4efd-b849-27988ceecb5d",

"nameConditioner": "cond1",

"inventoryNumber": "number1",

"place": "KMK",

"startDate": null,

"maintenance": [],

"workedHours": 120,

"deleted": true

},

{

"uuidConditioner": "1a1c7a1d-c368-4402-b2c2-b7ee20827e2f",

"nameConditioner": "cond7",

"inventoryNumber": "number57",

"place": "KMK",

"startDate": null,

"maintenance": [

{

"uuidTypeMaintenance": "a19ad67c-8408-4be1-acfa-e2f11c3a70f5",

"nameMaintenance": "main1",

"peopleHours": "10",

"deleted": false

}

],

"workedHours": 120,

"deleted": true

},

]

### Примечание:

## Удаление кондиционера по его id

Пользователь (Администратор или менеджер) может удалить уондиционер по его уникальному Uuid

### API: delete\_conditioner\_by\_id:

### Link: …/ conditioner/{conditionerUuid}

### Response: DELETE

### Request:

{

"uuidConditioner": "1a1c7a1d-c368-4402-b2c2-b7ee20827e2f",

"nameConditioner": "cond7",

"inventoryNumber": "number57",

"place": "KMK",

"startDate": null,

"maintenance": [

{

"uuidTypeMaintenance": "a19ad67c-8408-4be1-acfa-e2f11c3a70f5",

"nameMaintenance": "main1",

"peopleHours": "10",

"deleted": false

}

],

"workedHours": 120,

"deleted": true

}

Примечание: /кондиционер не удаляется физически из БД  
ставится отметка, что он удален и все.

## Запуск кондиционера в работу

Любой пользователь может отметить кондиционер, как тот, который начал работать. Как только кондиционер запущен в работу, его поле «start» становится true, а в поле start\_date появляется время, когда кондиционер был запущен в работу

!!!При старте кондиционера запись о нем добавляется в таблицу – «для планирования»

!!!Только тот кондиционер, у которого уже назначены ТО может быть запущен в работу

### API: start\_conditioner\_by\_id:

Link: …/conditioner/start/{conditionerUuid}

### Response: PUT

### Request:

" Conditioner with inventoryNumber CON20r started to work"

### Примечание:

Если кондиционер уже запущен в работу, и пользователь пытается запустить его второй раз – он получит предупреждение:

" Conditioner with inventoryNumber CONdfd1s20r is working now "

## Остановка кондиционера

Любой пользователь может отметить кондиционер, как тот, который остановлен. Как только кондиционер остановлен в работу, его поле «start» становится false, а в поле start\_date появляется null, stop = true, stop\_date время, когда кондиционер был остановлен в работу, workHours появляется количество часов, которые кондиционер отработал с момента последнего ТО

### API: start\_conditioner\_by\_id:

Link: …/conditioner/start/{conditionerUuid}

### Response: PUT

### Request:

" Conditioner with inventoryNumber CON20r started to work"

### Примечание:

Если кондиционер уже запущен в работу, и пользователь пытается запустить его второй раз – он получит предупреждение:

" Conditioner with inventoryNumber CONdfd1s20r is working now "

## Получение всех не запущенных кондиционеров

Пользователь (Администратор или менеджер) может получить все кондиционеры, которые не запущены в работу из базы данных

### API: get\_all\_not-started-conditioners:

### Link: …/conditioner/not-start/

### Response: GET

### Request:

[

{

"uuidConditioner": "9772de0c-3789-4965-9df6-d125f2161006",

"nameConditioner": "conditioner5",

"inventoryNumber": "CON6r",

"place": "KMK",

"startDate": null,

"maintenance": [

{

"uuidTypeMaintenance": "524d1cda-2a82-4978-84c9-0a683691b454",

"nameMaintenance": "TO2",

"peopleHours": 10,

"deleted": false,

"hoursBeforeTypeMaintenance": 10

}

],

"workedHours": 120,

"deleted": false

}

]

### Примечание:

возвращает все кондиционеры, которые еще не стартовали, и у которых уже назначены ТО этот API нуже для того, что бы после по его данным запускать кондиционеры в работу.

## Получение всех без ТО кондиционеров

Пользователь (Администратор или менеджер) может получить все кондиционеры, которым не назначено ТО

### API: get\_all\_not-type-maintenance:

### Link: …/conditioner/not-type-maintenance/

### Response: GET

### Request:

[

{

"uuidConditioner": "9772de0c-3789-4965-9df6-d125f2161006",

"nameConditioner": "conditioner5",

"inventoryNumber": "CON6r",

"place": "KMK",

"startDate": null,

"maintenance": [],

"workedHours": 120,

"deleted": false

}

]

### Примечание:

Возвращает список кондиционеров для того, что бы назначить им потом ТО

## Добавление вида ТО конкретному кондиционеру

Пользователь (Администратор или менеджер) может добавить кондиционеру вид его технического обслуживания

### Пользователь вызывает API: [get\_all\_type\_maintenance](#_Получение_всех_типов): и получает список всех типов обслуживания, которые существуют. Пользователь выбирает из выпадающего списка тип обслуживания.

1. Пользователь вызывает API: [get\_conditioner\_by\_id:](#_Получение_кондиционера_по) и получает тот кондиционер, которому он хочет присвоить тип технического обслуживания.
2. Пользователь вызывает

### API : add\_type\_maintenance \_to\_conditioner

### LINK

Link: …/conditioner/{conditionerUuid}/{typeMaintenanceUuid}

### Response: PUT

### Request:

{

"uuidConditioner": "203119ea-3aa9-46b5-a4fd-3d5e4cd3f37a",

"nameConditioner": "cond4",

"inventoryNumber": "number5",

"place": "KMK",

"startDate": "2020-07-26T10:07:37.490+00:00",

"maintenance": [

{

"uuidTypeMaintenance": "a19ad67c-8408-4be1-acfa-e2f11c3a70f5",

"nameMaintenance": "main2",

"peopleHours": "10",

"deleted": false

}

],

"workedHours": 120,

"deleted": false

}

### Примечание:

Если кондиционер или ТО удалены (то есть deleted= true), вернется сообщение, что такого кондиционера или ТО нет в БД.

# Планирование ТО

Пользователь (Администратор или менеджер) планирует ТО конкретного кондиционера

Зайдя в приложение, администратор может выбрать из таблицы for planning:

* [Просроченные ТО](#_Кондиционеры,_для_которых)
* [Кондиционеры, у которых ТО должно быть сегодня](#_Кондиционеры,_ТО_сегодня)
* [В ближайшую неделю](#_Кондиционеры,_ТО_в)
* [В ближайший месяц](#_Кондиционеры,_ТО_в)
* [В ближайший год](#_Кондиционеры,_ТО_в)
* [В дату от … до](#_Кондиционеры,_ТО_в) …

Формат даты: 2020-08-07T12:05:54.822

Фронт отправляет дату со временем – старт **00:00:00.000**

Фронт отправляет дату со временем - финиш: **23:59:59.000**

## Кондиционеры, для которых срок ТО уже прошел

### API : get all missed conditioners

### LINK

Link: …/planning/missed

### Response: GET

### Request:

[

{

"uuidRecords": "06f17ad0-30c9-482e-aafc-35576cda7236",

"uuidConditioner": "8d965c79-6105-47e6-877e-140d247b2746",

"nameConditioner": "conditioner5",

"inventoryNumber": "CON5r",

"place": "KMK",

"maintenance": {

"uuidTypeMaintenance": "cd42acb0-6abf-4fc7-94bb-017cd68e7bf5",

"nameMaintenance": "TO1",

"peopleHours": 10,

"deleted": false,

"hoursBeforeTypeMaintenance": 0

},

"lastTypeMaintenanceDate": "2020-07-27T14:41:05.572",

"nextTypeMaintenanceDate": "2020-07-27T14:41:05.572"

}

]

### Примечание:

Возвращает все кондиционеры, для которых сроки технического обслуживания прошли на момента «сегодня»

## Кондиционеры, ТО сегодня

### API : get all conditioners for today

### LINK

Link: …/planning

### Response: POST

#### DTO:

* LocalDateTime startDate;
* LocalDateTime finishDate;

#### JSON:

{}

### Request:

[

{

"uuidRecords": "06f17ad0-30c9-482e-aafc-35576cda7236",

"uuidConditioner": "8d965c79-6105-47e6-877e-140d247b2746",

"nameConditioner": "conditioner5",

"inventoryNumber": "CON5r",

"place": "KMK",

"maintenance": {

"uuidTypeMaintenance": "cd42acb0-6abf-4fc7-94bb-017cd68e7bf5",

"nameMaintenance": "TO1",

"peopleHours": 10,

"deleted": false,

"hoursBeforeTypeMaintenance": 0

},

"lastTypeMaintenanceDate": "2020-07-27T14:41:05.572",

"nextTypeMaintenanceDate": "2020-07-27T14:41:05.572"

}

]

### Примечание:

Если на сегодня ТО не запланировано -= вернется просто пустой объект

## Кондиционеры, ТО в будущем

### API : get all conditioners for future

### LINK

Link: …/planning

### Response: GET

#### DTO:

* LocalDateTime startDate; //с какой даты проверяем
* LocalDateTime finishDate;//до какой даты проверяем
  + - * !!! ФОРМАТ: 2020-08-07T00:00:00.000

#### JSON:

{

"startDate":"2020-08-25T11:44:44.797",

"finishDate":"2021-10-25T11:44:44.797"

}

### Request:

[

{

"uuidRecords": "06f17ad0-30c9-482e-aafc-35576cda7236",

"uuidConditioner": "8d965c79-6105-47e6-877e-140d247b2746",

"nameConditioner": "conditioner5",

"inventoryNumber": "CON5r",

"place": "KMK",

"maintenance": {

"uuidTypeMaintenance": "cd42acb0-6abf-4fc7-94bb-017cd68e7bf5",

"nameMaintenance": "TO1",

"peopleHours": 10,

"deleted": false,

"hoursBeforeTypeMaintenance": 0

},

"lastTypeMaintenanceDate": "2020-07-27T14:41:05.572",

"nextTypeMaintenanceDate": "2020-07-27T14:41:05.572"

}

]

### Примечание:

Даты из срока берутся из Json. Время -> время начала 00:00:00

Время окончания -> 23:59:59 то есть данные берутся за весь рабочий день, без учета времени подачи запроса

# Отправка в работу кондиционеров для ТО

Когда пользователь получает список всех кондиционеров, которым необходимо ТО в указанное пользователем время, он может нажать на любой кондиционер и отправить его в работу.

## Отправка кондиционера вы работу

### API : make type maint to plan

### LINK…/planning/plan

### Response: POST

#### DTO:

* String inventoryNumber; //инвентарный номер кондиуионера
* String nameConditioner;//имя кондиционера
* LocalDateTime startTime;//планируемое время начала проведения ТО
* List<WorkerDto> workers;//Список работников
* String place;//место, где проходит ТО и расположен кондиционер
* TypeMaintenanceForDto typeMaintenance;//вид ТО
* String planningRecordUuid;//id записи

#### JSON:

{

"inventoryNumber": "CON5r",

"nameConditioner": "conditioner5",

"startTime" : "2020-07-27T15:00:00.572",

"workers":

[

{

"userUuid": "ed52c7a6-8de6-4675-8610-180be4909618",

"firstName": "user1",

"lastName": "manager",

"email": "smilyk1982@gmail.com"

},

{

"userUuid": "15bda89a-0a7c-40cf-b58d-f809cc603f56",

"firstName": "user2",

"lastName": "manager2",

"email": "smilyk1982@gmail.com7"

}

],

"place": "KMK",

"typeMaintenance" : {

"uuidTypeMaintenance": "cd42acb0-6abf-4fc7-94bb-017cd68e7bf5"

},

"planningRecordUuid" : "d63fc85c-1687-438d-9928-78a8d5fd6546"

}

### Request:

next type maintenance for conditioner conditioner5 with inventory number CON5r was planning on date 2020-08-05T13:00

### Примечание:

API возвращает стринг, в котором указано, каое ТО в какое время для какого кондиционера запланировано.

Когда пользователь планирует выполнение ТО, ему необходимо назначить работников.

В процессе работы:

1. В таблицу где ведется учет занятых работников добавляется запись, где сказано, в какое время какой сотрудник занят. (ТАБЛИЦА “[busy\_workers](#_busy_workers)”)
2. В таблице для планирования обновляется запись, которая «запланирована». А именно – в поле «дата последнего ТО» добавляется то время, в которое ЗАКОНЧИТСЯ ТО, в поле дата следующего ТО добавляется дата = дата последнего ТО + то количество часов, которое кондиционер должен отработать (ТАБЛИЦА «[for\_planning](#_\«for_planning\»))»)
3. Добавляется запись в таблицу, где хранятся записи тех работ, которые уже спланированы на конкретный день(ТАБЛИЦА “[in\_work](#_in_work)”)
   1. При сохранении записи в БД поле «В работе» = false. Изменится на true в момент, когда рабочий возьмет в работу это запись.
   2. После того, как работа будет выполнена, поле done станет true
   3. После того, как работа будет выполнена, будет сохранена запись в таблицу «сравнение планируемого и реального результата» (версия 2)
   4. После того, как работа будет выполнена, запись будет удалена????????? Из таблицы, а в таблице «архив» появится запись о проделанной работе со всеми подробностями.

Возвращает все кондиционеры, для которых сроки технического обслуживания прошли на момента «сегодня»

## Получение всех свободных работников в конкретное время

### API : getting all not busy workers

### LINK…//users/all-not-busy

### Response: GET

#### DTO:

* LocalDateTime startDate;//старт даты, когда начинается ТО
* String uuidTypeMaintenance;//тип ТО
* Integer countOfWorkers;//сколько работников

#### JSON:

{

"startDate":"2020-05-24T14:00:00.000",

"uuidTypeMaintenance":"cd42acb0-6abf-4fc7-94bb-017cd68e7bf5",

"countOfWorkers":2

}

### Request:

[

{

"userUuid": "15bda89a-0a7c-40cf-b58d-f809cc603f56",

"firstName": "user2",

"lastName": "manager2",

"email": "smilyk1982@gmail.com7"

}

]

### Примечание:

По запросу возвращаются те работники, которые в указанные период времени НЕ заняты

# РАБОТНИКИ ПРИНИМАЮТ В РАБОТУ ТО И ПРОВОДЯТ ТО

Когда ремонтник открывает свое приложение он видит, какие ТО он должен провести.

Он может выбрать ТО на:

* [Сегодня](#_ТО_которых_сегодня)
* Завтра
* За определенную дату
* [ТО срок которых уже прошел](#_ТО_срок_которых)

## ТО срок которых прошел

### API : worker get all missed Type maintenance

LINK…/ /workers/missed/{workerUuid}

### Response: GET

Request:

#### DTO:

**результат**:

* String planningRecordsUuid;//ай ди записи запланированного ТО
* String nameConditioner;//название кондиуионера
* String inventoryNumber;//уникальные номер кондиционера
* String place;//место, где расположен кондиционер
* String typeMaintenanceName;//название ТО
* Integer typeMaintenancePeopleHorse;//сколько времени (всего) на работу
* LocalDateTime startTimes;//запланированное начальником время начала ТО
* List<WorkerDto> workers;//список работников, которые должны делать ТО

**WorkerDto:**

* String firstName;//имя сотрудника
* String lastName;//фамилия сотрудника

### String email;//почта сотрудника

[

{

"planningRecordsUuid": "70a36559-a0c7-464f-a5ed-14badae1e311",

"nameConditioner": "conditioner5",

"inventoryNumber": "CON5r",

"place": "KMK",

"typeMaintenanceName": "cd42acb0-6abf-4fc7-94bb-017cd68e7bf5",

"typeMaintenancePeopleHorse": 70,

"startTimes": "2020-07-27T14:00:00.572",

"workers": [

{

"userUuid": null,

"firstName": "user1",

"lastName": "manager",

"email": "smilyk1982@gmail.com"

},

{

"userUuid": null,

"firstName": "user2",

"lastName": "manager2",

"email": "smilyk1982@gmail.com7"

}

]

},

]

### Примечание:

По запросу возвращаются те ТО, которые были запланированы на старт раньше, чем сегодняшняя дата

## ТО срок которых сегодня

### API : worker get Type maintenance for today

### LINK…/ workers/workerUuid}

### Response: GET

#### DTO:

* LocalDateTime startDate;
* LocalDateTime finishDate;

#### JSON:

{}

### Request:

**результат**:

* String planningRecordsUuid;//ай ди записи запланированного ТО
* String nameConditioner;//название кондиуионера
* String inventoryNumber;//уникальные номер кондиционера
* String place;//место, где расположен кондиционер
* String typeMaintenanceName;//название ТО
* Integer typeMaintenancePeopleHorse;//сколько времени (всего) на работу
* LocalDateTime startTimes;//запланированное начальником время начала ТО
* List<WorkerDto> workers;//список работников, которые должны делать ТО

**WorkerDto:**

* String firstName;//имя сотрудника
* String lastName;//фамилия сотрудника

[

{

"planningRecordsUuid": "c03ab031-9603-4042-ba03-7729b1ea4952",

"nameConditioner": "conditioner5",

"inventoryNumber": "CON5r",

"place": "KMK",

"typeMaintenanceName": "cd42acb0-6abf-4fc7-94bb-017cd68e7bf5",

"typeMaintenancePeopleHorse": 70,

"startTimes": "2020-08-14T14:00:00.572",

"workers": [

{

"userUuid": null,

"firstName": "user1",

"lastName": "manager",

"email": "smilyk1982@gmail.com"

},

{

"userUuid": null,

"firstName": "user2",

"lastName": "manager2",

"email": "smilyk1982@gmail.com7"

}

]

}

]

### Примечание:

По запросу возвращаются те ТО, которые были запланированы на на сегодняшнюю дату

## ТО срок которых в будущем

### API : worker get Type maintenance for today

### LINK: …/ workers/{workerUuid}

Link: …/planning

### Response: GET

#### DTO:

* LocalDateTime startDate; //с какой даты проверяем
* LocalDateTime finishDate;//до какой даты проверяем
  + - * !!! ФОРМАТ: 2020-08-07T00:00:00.000

#### JSON:

{

"startDate":"2020-08-25T00:00:00.000",

"finishDate":"2021-10-25T00:00:00.000"

}

### Request:

[

{

"planningRecordsUuid": "70a36559-a0c7-464f-a5ed-14badae1e311",

"nameConditioner": "conditioner5",

"inventoryNumber": "CON5r",

"place": "KMK",

"typeMaintenanceName": "cd42acb0-6abf-4fc7-94bb-017cd68e7bf5",

"typeMaintenancePeopleHorse": 70,

"startTimes": "2021-07-27T14:00:00.572",

"workers": [

{

"userUuid": null,

"firstName": "user1",

"lastName": "manager",

"email": "smilyk1982@gmail.com"

},

{

"userUuid": null,

"firstName": "user2",

"lastName": "manager2",

"email": "smilyk1982@gmail.com7"

}

]

}

]

### Примечание:

Даты из срока берутся из Json. Время -> время начала 00:00:00

Время окончания -> 23:59:59 то есть данные берутся за весь рабочий день, без учета времени подачи запроса

## Работник берет в работу запланированное ТО

Когда работник берет в работу запанированное ранее ТО, то в таблице [in\_work](#_in_work) колонка in\_work = true.

* startTime меняется на «сейчас»

Если запланированное ранее задание в статусе «в работе», то при планировании передвигать его сроки запрещено (вер2)

### API : make planning type maintenance done

### LINK: …/ workers/{recordUuid}/{workerUuid}

### Response: PUT

#### DTO:

#### JSON:

### Request:

[

{

"planningRecordsUuid": "70a36559-a0c7-464f-a5ed-14badae1e311",

"nameConditioner": "conditioner5",

"inventoryNumber": "CON5r",

"place": "KMK",

"typeMaintenanceName": "cd42acb0-6abf-4fc7-94bb-017cd68e7bf5",

"typeMaintenancePeopleHorse": 70,

"startTimes": "2021-07-27T14:00:00.572",

"workers": [

{

"userUuid": null,

"firstName": "user1",

"lastName": "manager",

"email": "smilyk1982@gmail.com"

},

{

"userUuid": null,

"firstName": "user2",

"lastName": "manager2",

"email": "smilyk1982@gmail.com7"

}

]

}

]

### Примечание:

## Работник заканчивает работу с запланированным ТО

Когда работник заканчивает работу с ТО, то в таблице [in\_work](#_in_work) колонка done = true.

* Когда сотрудник заканчивает работу с ТО, в таблице [for\_planning](#_for_planning) меняется время – на «сегодня» + «количество часов ТО»
* Из таблицы [in\_work](#_in_work) запись перемешается в таблицу [архив](#_arhiv)

### API : take planning type maintenance done

### LINK: …/ workers/{recordUuid}/{workerUuid}

### Response: PUT

#### DTO:

#### JSON:

### Request:

[

{

"planningRecordsUuid": "70a36559-a0c7-464f-a5ed-14badae1e311",

"nameConditioner": "conditioner5",

"inventoryNumber": "CON5r",

"place": "KMK",

"typeMaintenanceName": "cd42acb0-6abf-4fc7-94bb-017cd68e7bf5",

"typeMaintenancePeopleHorse": 70,

"startTimes": "2021-07-27T14:00:00.572",

"workers": [

{

"userUuid": null,

"firstName": "user1",

"lastName": "manager",

"email": "smilyk1982@gmail.com"

},

{

"userUuid": null,

"firstName": "user2",

"lastName": "manager2",

"email": "smilyk1982@gmail.com7"

}

]

}

]

### Примечание:

# ТАБЛИЧКИ

## busy\_workers

## for\_planning

## in\_work

## arhiv