Простое управление настройками django приложения

Денис Дудник / Евгений Кацевман, SigmaDevs

Что сейчас будет

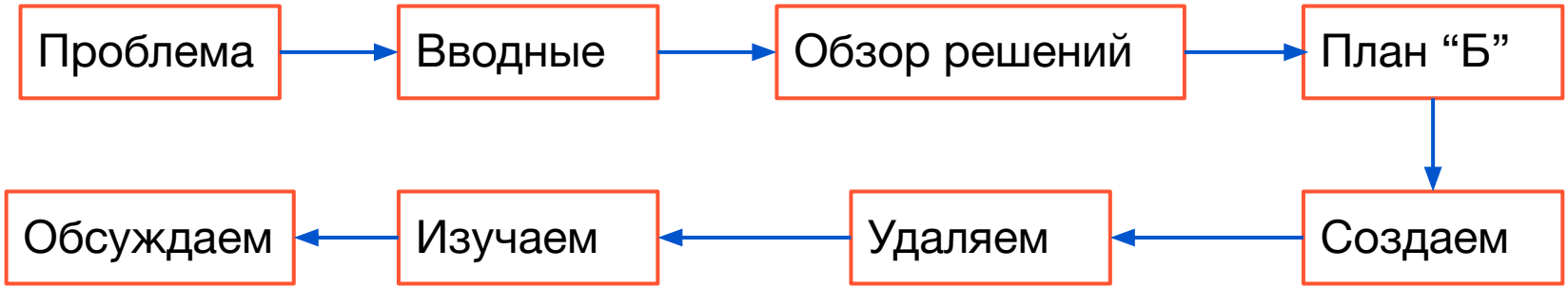
Спикер:	Денис Дудник	Евгений Кацевман
Позиция:	Backend Developer	Backend Team Lead
Скилы:	Python	Python

Время доклада: 30 минут

Вопросы: 15 минут после доклада

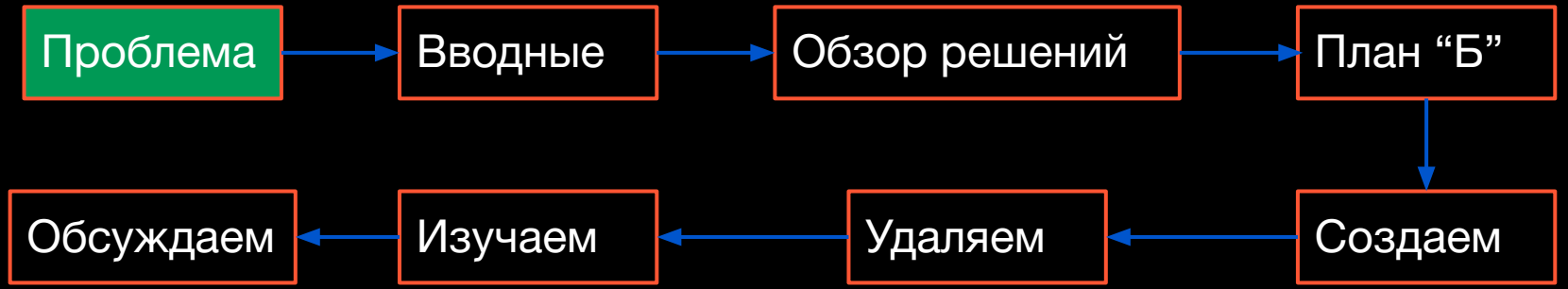
Репозиторий тут: https://github.com/liveconfigs/django-liveconfigs

План доклада



Репозиторий: https://github.com/liveconfigs/django-liveconfigs

Проблема


Проблема
Вводные
Обзор решений
План “Б”
Создаем
Удаляем
Изучаем
Обсуждаем

Проблема

1. Много стендов

Проблема

1. Много стендов
2. Настраиваемый снаружи код

Проблема

1. Много стендов
2. Настраиваемый снаружи код
3. Создание и обслуживание настроек

Проблема

1. Много стендов
2. Настраиваемый снаружи код
3. Создание и обслуживание настроек
4. Передача настроек на сервера

Проблема

1. Много стендов
2. Настраиваемый снаружи код
3. Создание и обслуживание настроек
4. Передача настроек на сервера
5. Хотелось бы быстрее, легче и красивее

Проблема

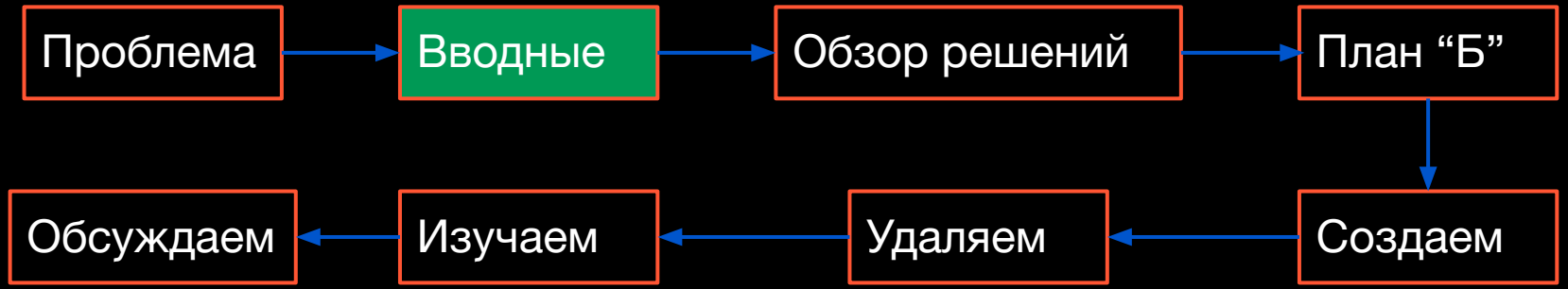
1. Много стендов
2. Настраиваемый снаружи код
3. Создание и обслуживание настроек
4. Передача настроек на сервера
5. Хотелось бы быстрее, легче и красивее

Сегодня расскажем как это делаем мы!

Как это делаем мы

```
class MyConfig(liveconfigs.BaseConfig):
    IS_FEATURE_ENABLED: bool = False
    NEW_FEATURE_VALUE: float = 20.4

if MyConfig.IS_FEATURE_ENABLED:
    print(
        "feature is enabled and feature value is",
        MyConfig.NEW_FEATURE_VALUE
    )
```

Вводные


Проблема
Вводные
Обзор решений
План “Б”
Создаем
Удаляем
Изучаем
Обсуждаем

О каких настройках поговорим

- Говорим тут только о бизнес-настройках приложения и немного о технических

О каких настройках поговорим

- Говорим тут только о бизнес-настройках приложения и немного о технических
- Не говорим о большой массе технических настроек, которые могут лежать в переменных окружения

О каких настройках поговорим

- Говорим тут только о бизнес-настройках приложения и немного о технических
- Не говорим о большой массе технических настроек, которые могут лежать в переменных окружения
- Не говорим о настройках пользователя

Жизненный цикл настроек

- Придумать новую настройку

Жизненный цикл настроек

- Придумать новую настройку
- Добавить новую настройку с документацией и валидаторами

Жизненный цикл настроек

- Придумать новую настройку
- Добавить новую настройку с документацией и валидаторами
- Использовать новую настройку в коде

Жизненный цикл настроек

- Придумать новую настройку
- Добавить новую настройку с документацией и валидаторами
- Использовать новую настройку в коде
- Найти настройки

Жизненный цикл настроек

- Придумать новую настройку
- Добавить новую настройку с документацией и валидаторами
- Использовать новую настройку в коде
- Найти настройки
- Поменять настройки допустимым образом

Жизненный цикл настроек

- Придумать новую настройку
- Добавить новую настройку с документацией и валидаторами
- Использовать новую настройку в коде
- Найти настройки
- Поменять настройки допустимым образом
- Видеть, кто, как и когда менял настройки

Жизненный цикл настроек

- Придумать новую настройку
- Добавить новую настройку с документацией и валидаторами
- Использовать новую настройку в коде
- Найти настройки
- Поменять настройки допустимым образом
- Видеть, кто, как и когда менял настройки
- Экспортировать и импортировать настройки, переносить их со стенда на стенд

Жизненный цикл настроек

- Придумать новую настройку
- Добавить новую настройку с документацией и валидаторами
- Использовать новую настройку в коде
- Найти настройки
- Поменять настройки допустимым образом
- Видеть, кто, как и когда менял настройки
- Экспортировать и импортировать настройки, переносить их со стенда на стенд
- Найти и удалить неиспользуемые настройки

История и предпосылки

- Командой из 3-х человек

История и предпосылки

- Командой из 3-х человек
- На python (django)

История и предпосылки

- Командой из 3-х человек
- На python (django)
- Разрабатывали и поддерживали чат-бота-секретаря с обработкой естественного языка

История и предпосылки

- Командой из 3-х человек
- На python (django)
- Разрабатывали и поддерживали чат-бота-секретаря с обработкой естественного языка
- И со множеством сценариев поведения

История и предпосылки - компоненты

- Django и Celery из одного образа

История и предпосылки - компоненты

- Django и Celery из одного образа
- Postgres

История и предпосылки - компоненты

- Django и Celery из одного образа
- Postgres
- Redis как брокер и кеш

История и предпосылки - компоненты

- Django и Celery из одного образа
- Postgres
- Redis как брокер и кеш
- Достаточно объемные языковые модели

История и предпосылки - компоненты

- Django и Celery из одного образа
- Postgres
- Redis как брокер и кеш
- Достаточно объемные языковые модели
- Плохое покрытие юнит-тестами

История и предпосылки - ограничения

- Сервис долго стартует

История и предпосылки - ограничения

- Сервис долго стартует
- Ограничены ресурсы на выполнение

История и предпосылки - ограничения

- Сервис долго стартует
- Ограничены ресурсы на выполнение
- Быстро исправить пункты выше не можем

История и предпосылки - постановка

- Изменения настроек должны подхватываться «сразу»

История и предпосылки - постановка

- Изменения настроек должны подхватываться «сразу»
- Добавляет настройки разработчик

История и предпосылки - постановка

- Изменения настроек должны подхватываться «сразу»
- Добавляет настройки разработчик
- В настройках может храниться любой json, не только флаги

История и предпосылки - постановка

- Изменения настроек должны подхватываться «сразу»
- Добавляет настройки разработчик
- В настройках может храниться любой json, не только флаги
- Должна быть валидация настроек при сохранении (не только по типу)

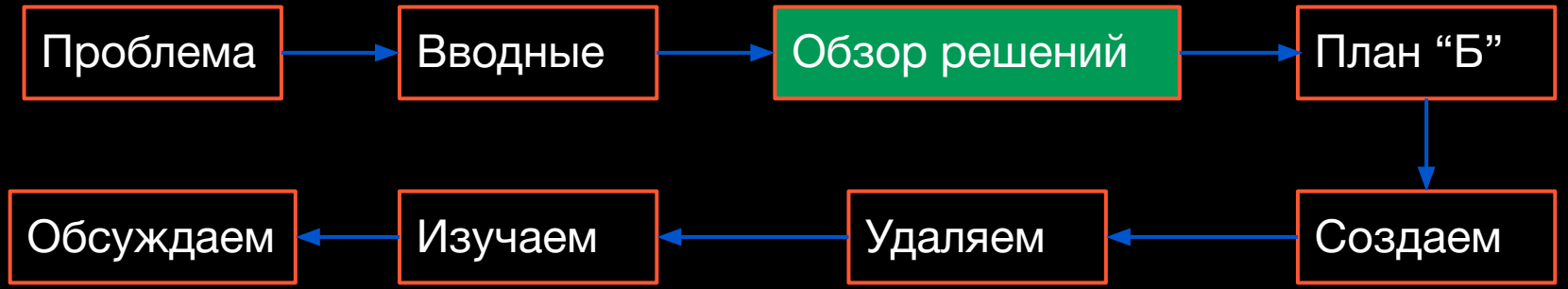
История и предпосылки - постановка

- Изменения настроек должны подхватываться «сразу»
- Добавляет настройки разработчик
- В настройках может храниться любой json, не только флаги
- Должна быть валидация настроек при сохранении (не только по типу)
- Должна быть история изменений

История и предпосылки - постановка

- Изменения настроек должны подхватываться «сразу»
- Добавляет настройки разработчик
- В настройках может храниться любой json, не только флаги
- Должна быть валидация настроек при сохранении (не только по типу)
- Должна быть история изменений
- Ожидаем, что разных настроек будет много

Это же django!
Возьми готовое!



Зачем велосипеды?

Почему не

growthbook

flipt unleash

django-waffle flagsmith

django-constance

django-dynamic-preferences

?

Примеры работы с этими решениями

Дополнительные сервисы

- growthbook
- flipt
- unleash
- flagsmith

Библиотеки django

- django-constance
- django-waffle
- django-dynamic-preferences

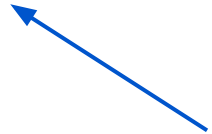
Примеры работы с этими решениями

Дополнительные сервисы

- growthbook
- flipt
- unleash
- flagsmith

Библиотеки django

- django-constance
- django-waffle
- django-dynamic-preferences



- Все устроены примерно одинаково
- Настройки на сервер нужно как-то занести
- Нет валидации значений или ее сложно настраивать
- Клиенты местами очень странные

Примеры работы с этими решениями

unleash

```
if unleash.isEnabled("AwesomeFeature"):
   do_new_stuff()
else:
   do_old_stuff()
```

flipt

```
bool_flag = flipt_client.evaluation.boolean(
   EvaluationRequest(
       namespace_key="default",
       flag_key="flag_boolean",
       entity_id="entity",
       context={"fizz": "buzz"},
   )
)
```

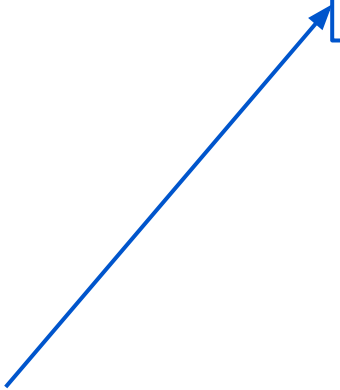
Примеры работы с этими решениями

Дополнительные сервисы

- growthbook
- flipt
- unleash
- flagsmith

Библиотеки django

- django-constance
- django-waffle
- django-dynamic-preferences



Предназначен для “динамизации” settings

Примеры работы с этими решениями

Django-constance

```
CONSTANCE_CONFIG = OrderedDict([
    ("SITE_NAME", ("My Title", "Website title")),
    ("SITE_DESCRIPTION", ("", "Website description")),
    ("THEME", ("light-blue", "Website theme")),
    ("THE_ANSWER", (42, "The answer")),
])

from constance import config
if config.THE_ANSWER == 42:
    answer_the_question()
```

Примеры работы с этими решениями

Django-constance

```
CONSTANCE_CONFIG = OrderedDict([
    ("SITE_NAME", ("My Title", "Website title")),
    ("SITE_DESCRIPTION", ("", "Website description")),
    ("THEME", ("light-blue", "Website theme")),
    ("THE_ANSWER", (42, "The answer")),
])

from constance import config
if config.THE_ANSWER == 42:
    answer_the_question()
```

Тип значения?

Валидаторы?

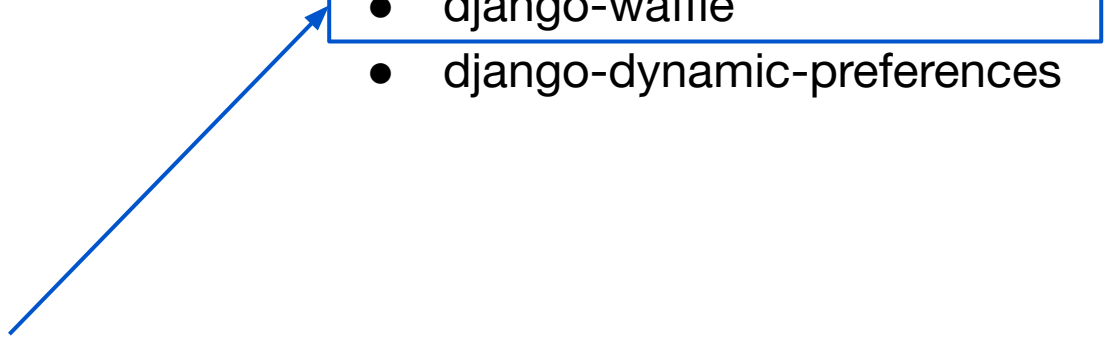
Примеры работы с этими решениями

Дополнительные сервисы

- growthbook
- flipt
- unleash
- flagsmith

Библиотеки django

- django-constance
- django-waffle
- django-dynamic-preferences



Не поддерживает ничего, кроме on/off флагов

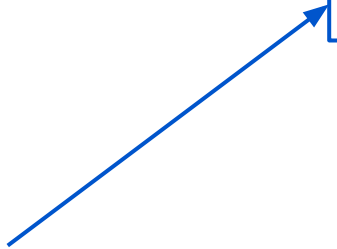
Примеры работы с этими решениями

Дополнительные сервисы

- growthbook
- flipt
- unleash
- flagsmith

Библиотеки django

- django-constance
- django-waffle
- django-dynamic-preferences



Самый близкий к нам, но слишком многословный, и есть не все фичи

Примеры работы с этими решениями

Один из паттернов в решениях

```
config.get_value('feature_name')
```

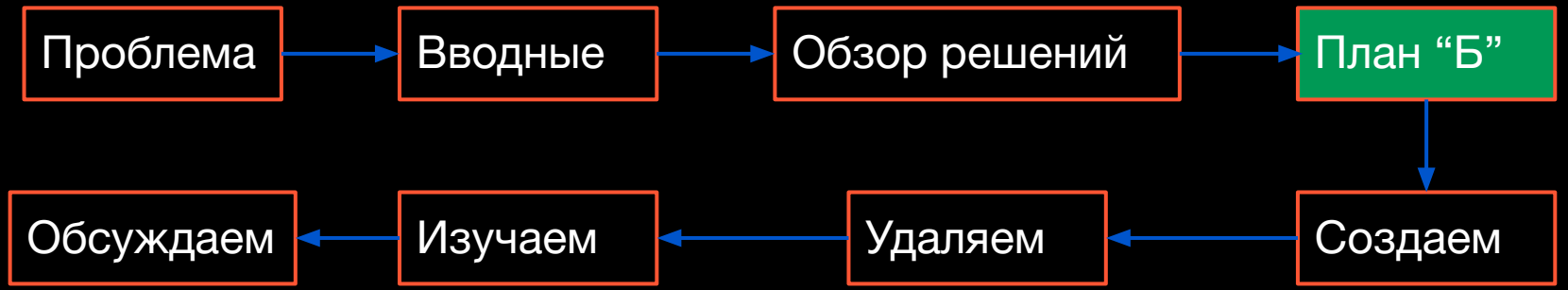
Примеры работы с этими решениями

Один из паттернов в решениях

```
config.get_value('feature_name')
```

Какой тип у этой настройки?
Где хранится ее значение по-умолчанию, документация и кто за них отвечает?
Как узнать названия всех доступных настроек?
Сколько нужно этапов, чтобы добавить настройку и донести ее до прода?

План “Б”. Делаем своё!
django-liveconfigs



Что в итоге решили

- Значения настроек и метаданные хранятся в одной отдельной таблице в json-поле

Что в итоге решили

- Значения настроек и метаданные хранятся в одной отдельной таблице в json-поле
- Валидация выполняется в админке django при попытке сохранения

Что в итоге решили

- Значения настроек и метаданные хранятся в одной отдельной таблице в json-поле
- Валидация выполняется в админке django при попытке сохранения
- Описание настроек хранится в **коде** (образ один) и выкатывается на стенд вместе с ним.

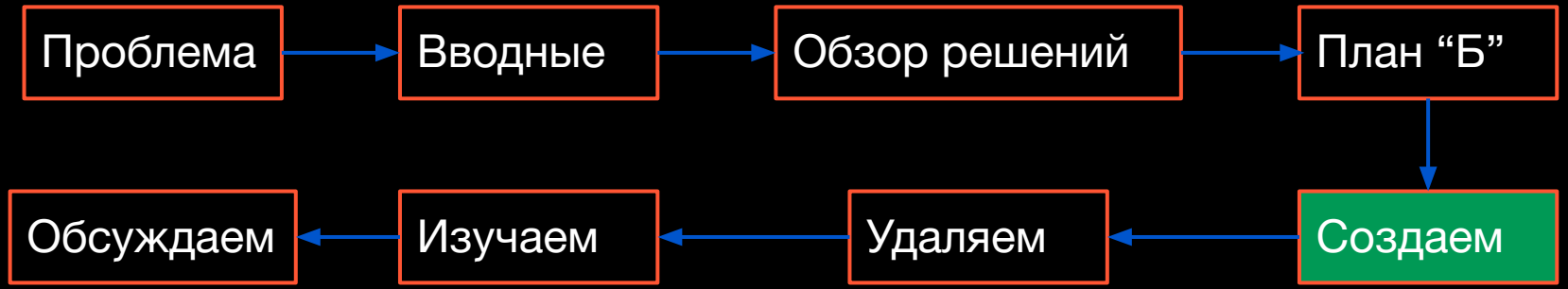
Что в итоге решили

- Значения настроек и метаданные хранятся в одной отдельной таблице в json-поле
- Валидация выполняется в админке django при попытке сохранения
- Описание настроек хранится в **коде** (образ один) и выкатывается на стенд вместе с ним.
- Описание настроек попадает в БД на старте сервиса или при первом обращении к настройке

Что в итоге решили

- Значения настроек и метаданные хранятся в одной отдельной таблице в json-поле
- Валидация выполняется в админке django при попытке сохранения
- Описание настроек хранится в **коде** (образ один) и выкатывается на стенд вместе с ним.
- Описание настроек попадает в БД на старте сервиса или при первом обращении к настройке
- Значение настройки кешируется в сервисе

Поехали! Создадим первый конфиг вместе!



```
INSTALLED_APPS = [
    ...
    "import_export",
    "liveconfigs",
]
...
# необязательные настройки liveconfigs
LC_MAX_STR_LENGTH_DISPLAYED_AS_TEXTINPUT = 50
LC_ENABLE_PRETTY_INPUT = True
LIVECONFIGS_SYNCWRITE = True
LC_CACHE_TTL = 1
LC_MAX_VISUAL_VALUE_LENGTH = 50
```

```python
class MyConfig(liveconfigs.BaseConfig):
    IS_FEATURE_ENABLED: bool = False
    NEW_FEATURE_VALUE: float | str | None = 20.4


if MyConfig.IS_FEATURE_ENABLED:
    print(
        "feature is enabled and feature value is",
        MyConfig.NEW_FEATURE_VALUE
    )
```

Есть поддержка Union!

IDE знает, подскажет и проверит имя и тип настройки!

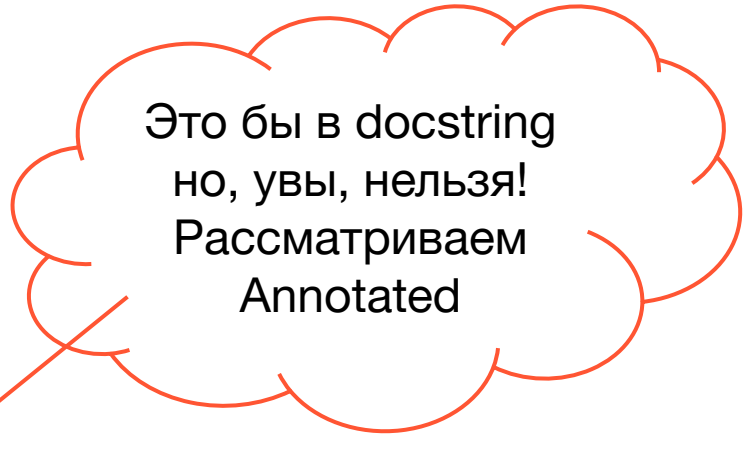
```python
class MyConfig(liveconfigs.BaseConfig):
    BOOL_VALUE: bool = False
    INT_VALUE: int = 20
    FLOAT_VALUE: float = 20.4
    STR_VALUE: str = “20.4”
    LIST_VALUE: list = []
    DICT_VALUE: dict = {}
    NESTED_VALUE: dict[str | int, list[float]] | None = None
```

```
class FirstExample(models.BaseConfig):
  __topic__ = "Описание первой группы настроек"
  __exported__ = ["FUEL_PRICES",]

  FUEL_PRICES: dict[str, float] = {
    "92": 50.5,
    "95": 55.8,
  }
  FUEL_PRICES_DESCRIPTION = "Стоимость видов топлива"
  FUEL_PRICES_TAGS = [ConfigTags.basic]
  FUEL_PRICES_VALIDATORS = [dict_values_are(greater_than(0))]
```

```python
class FirstExample(models.BaseConfig):
  __topic__ = "Описание первой группы настроек"
  __exported__ = ["FUEL_PRICES",]

  FUEL_PRICES: dict[str, float] = {
    "92": 50.5,
    "95": 55.8,
  }
  FUEL_PRICES_DESCRIPTION = "Стоимость видов топлива"
  FUEL_PRICES_TAGS = [ConfigTags.basic]
  FUEL_PRICES_VALIDATORS = [dict_values_are(greater_than(0))]
```

```python
class FirstExample(models.BaseConfig):
  __topic__ = "Описание первой группы настроек"
  __exported__ = ["FUEL_PRICES",]

  FUEL_PRICES: dict[str, float] = {
    "92": 50.5,
    "95": 55.8,
  }
  FUEL_PRICES_DESCRIPTION = "Стоимость видов топлива"
  FUEL_PRICES_TAGS = [ConfigTags.basic]
  FUEL_PRICES_VALIDATORS = [dict_values_are(greater_than(0))]
```

```python
class FirstExample(models.BaseConfig):
    __topic__ = "Описание первой группы настроек"
    __exported__ = ["FUEL_PRICES",]

    FUEL_PRICES: dict[str, float] = {
        "92": 50.5,
        "95": 55.8,
    }
    FUEL_PRICES_DESCRIPTION = "Стоимость видов топлива"
    FUEL_PRICES_TAGS = [ConfigTags.basic]
    FUEL_PRICES_VALIDATORS = [dict_values_are(greater_than(0))]
```



```python
class FirstExample(models.BaseConfig):
  __topic__ = "Описание первой группы настроек"
  __exported__ = ["FUEL_PRICES",]

  FUEL_PRICES: dict[str, float] = {
    "92": 50.5,
    "95": 55.8,
  }
  FUEL_PRICES_DESCRIPTION = "Стоимость видов топлива"
  FUEL_PRICES_TAGS = [ConfigTags.basic]
  FUEL_PRICES_VALIDATORS = [dict_values_are(greater_than(0))]
```

```python
class FirstExample(models.BaseConfig):
  __topic__ = "Описание первой группы настроек"
  __exported__ = ["FUEL_PRICES",]

  FUEL_PRICES: dict[str, float] = {
    "92": 50.5,
    "95": 55.8,
  }
  FUEL_PRICES_DESCRIPTION = "Стоимость видов топлива"
  FUEL_PRICES_TAGS = [ConfigTags.basic]
  FUEL_PRICES_VALIDATORS = [dict_values_are(greater_than(0))]
```

```python
class FirstExample(models.BaseConfig):
  __topic__ = "Описание первой группы настроек"
  __exported__ = ["FUEL_PRICES",]

  FUEL_PRICES: dict[str, float] = {
    "92": 50.5,
    "95": 55.8,
  }
  FUEL_PRICES_DESCRIPTION = "Стоимость видов топлива"
  FUEL_PRICES_TAGS = [ConfigTags.basic]
  FUEL_PRICES_VALIDATORS = [dict_values_are(greater_than(0))]
```

```python
class FirstExample(models.BaseConfig):
  __topic__ = "Описание первой группы настроек"
  __exported__ = ["FUEL_PRICES",]

  FUEL_PRICES: dict[str, float] = {
    "92": 50.5,
    "95": 55.8,
  }
  FUEL_PRICES_DESCRIPTION = "Стоимость видов топлива"
  FUEL_PRICES_TAGS = [ConfigTags.basic]
  FUEL_PRICES_VALIDATORS = [dict_values_are(greater_than(0))]
```

Select config row to change

IMPORT EXPORT ADD CONFIG ROW +

Search

Action: --------- Go 0 of 7 selected

NAME	VALUE	IS CHANGED	DESCRIPTION	TOPIC	TAGS	LAST READ	LAST SET
USE_CALENDAR	false	☐	Использовать производственнный календарь	Описание первой группы настроек	["Основные"]	Aug. 13, 2024, 3:41 p.m.	June 20, 2024, 8:14 a.m.
TEST	[]	☑	-	Описание первой группы настроек	-	-	June 20, 2024, 8:14 a.m.
SECRET_SWITCH	false	☐	Настройка не для экспорта!	Описание первой группы настроек	["Фичи"]	Aug. 13, 2024, 3:41 p.m.	June 20, 2024, 8:14 a.m.
FUEL_PRICES	{"92": 50.5, "95": 55.8}	☐	Стоимость различных видов топлива	Описание первой группы настроек	["Основные"]	Aug. 13, 2024, 3:41 p.m.	June 20, 2024, 8:14 a.m.
FIRST_DAY_OF_WEEK	null	☐	Первый день недели	Описание первой группы настроек	["Фичи"]	Aug. 13, 2024, 3:41 p.m.	Aug. 13, 2024, 3:30 p.m.

FILTER

Show counts

↓ By topic

All

Описание первой группы настроек

↓ By tags

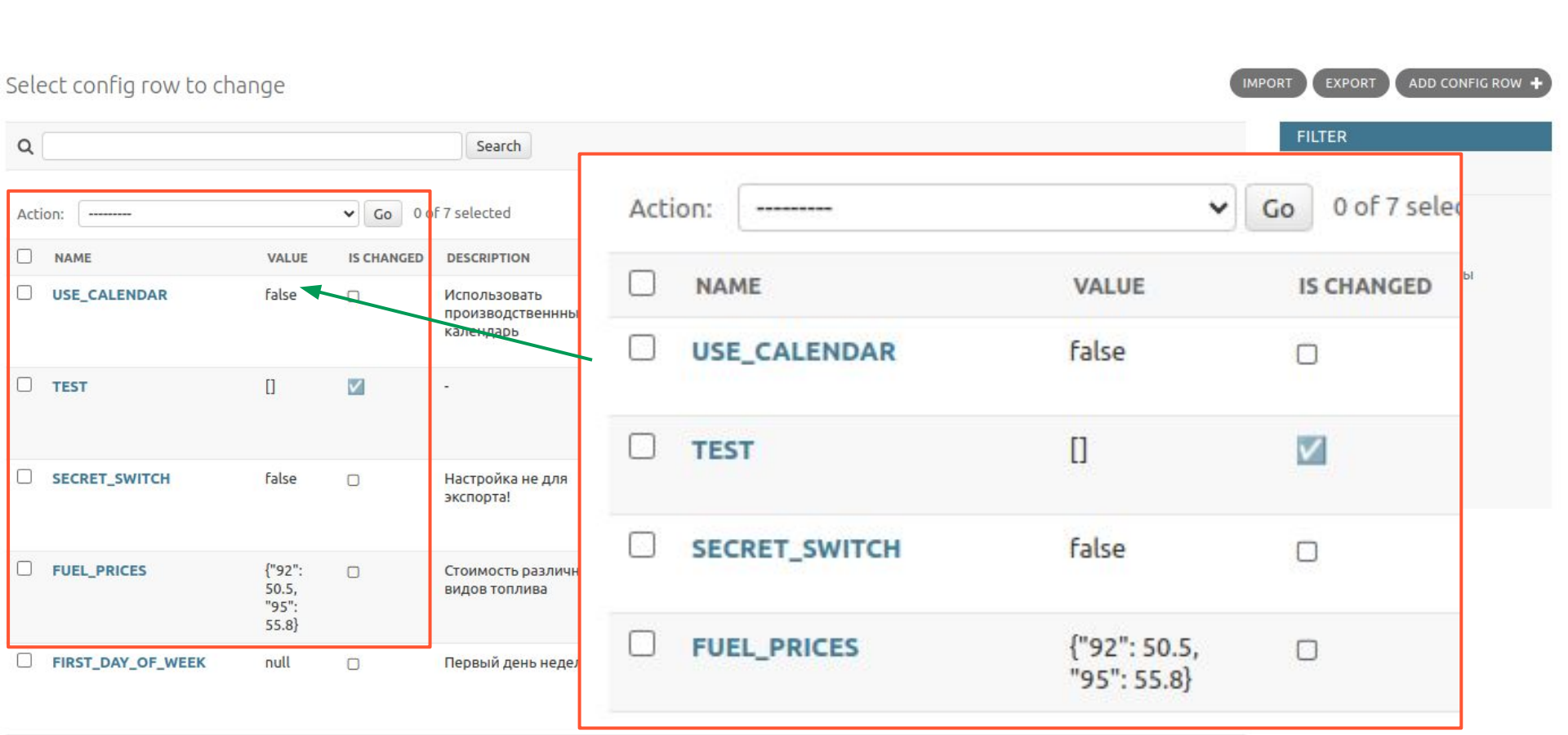
All

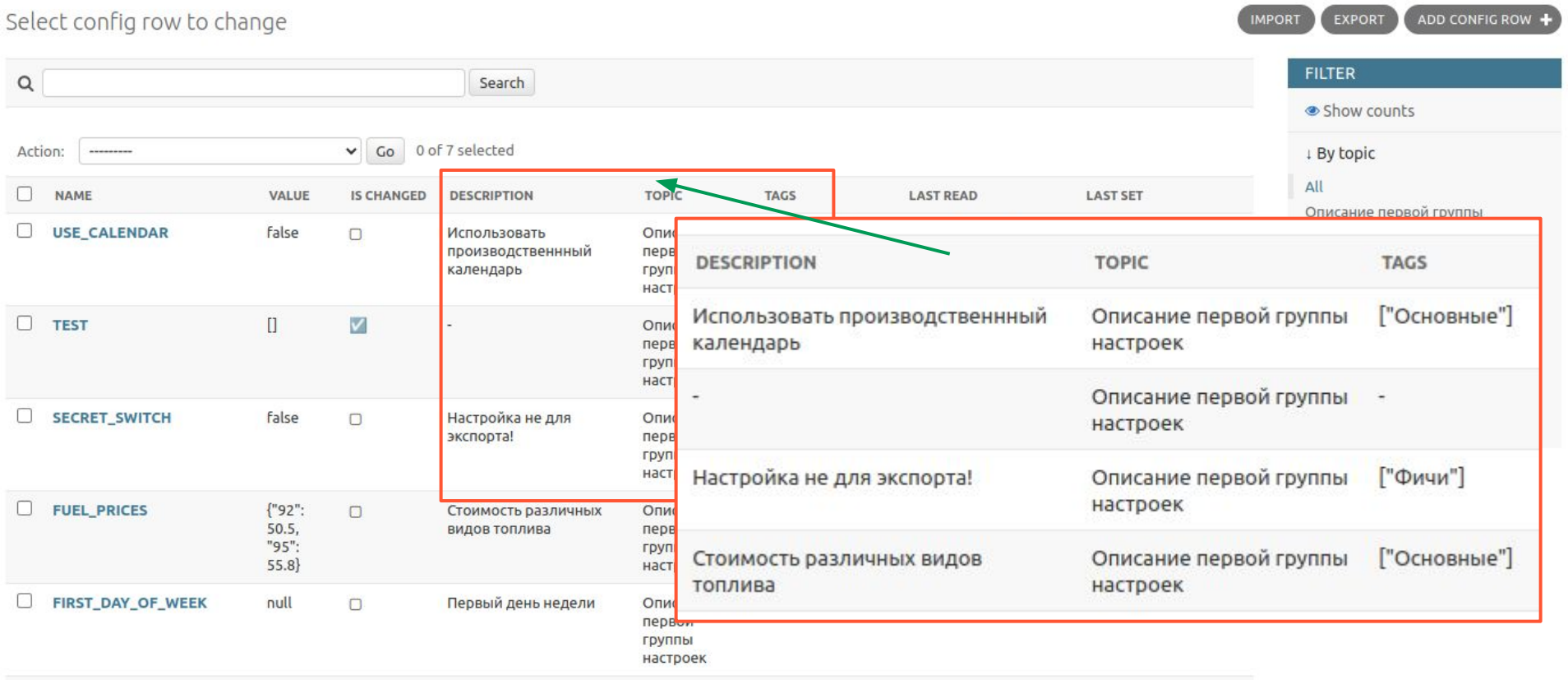
-

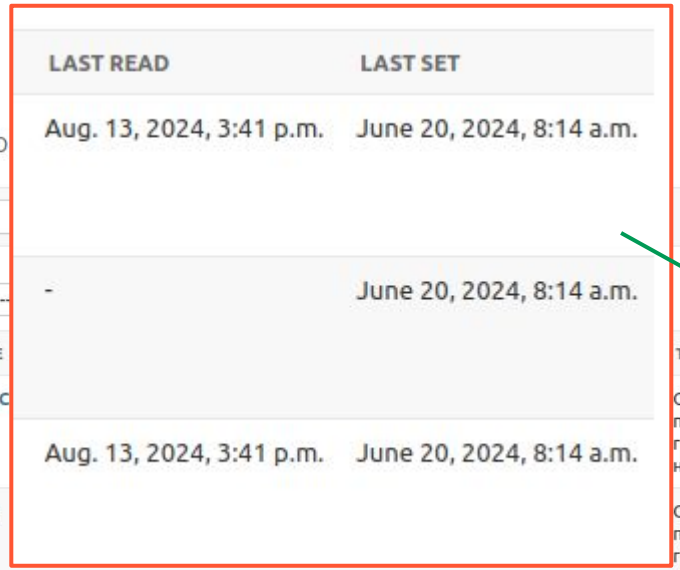
Настройки для фронта

Основные

Фичи


Select config row to change
IMPORT
EXPORT
ADD CONFIG ROW
Search
FILTER
Action:
Go
0 of 7 selected
NAME
VALUE
IS CHANGED
DESCRIPTION
USE_CALENDAR
false
Использовать производственны календарь
TEST
[]
-
SECRET_SWITCH
false
Настройка не для экспорта!
FUEL_PRICES
{"92": 50.5, "95": 55.8}
Стоимость различн видов топлива
FIRST_DAY_OF_WEEK
null
Первый день недел


Select config row to change
IMPORT
EXPORT
ADD CONFIG ROW +
Search
FILTER
Show counts
By topic
All
Описание первой группы
Action:
Go
0 of 7 selected
NAME
VALUE
IS CHANGED
DESCRIPTION
TOPIC
TAGS
LAST READ
LAST SET
USE_CALENDAR
false
Использовать производственнный календарь
TEST
[]
-
SECRET_SWITCH
false
Настройка не для экспорта!
FUEL_PRICES
{"92": 50.5, "95": 55.8}
Стоимость различных видов топлива
FIRST_DAY_OF_WEEK
null
Первый день недели
DESCRIPTION
TOPIC
TAGS
Использовать производственнный календарь
Описание первой группы настроек
["Основные"]
-
Описание первой группы настроек
-
Настройка не для экспорта!
Описание первой группы настроек
["Фичи"]
Стоимость различных видов топлива
Описание первой группы настроек
["Основные"]



Select co

IMPORT EXPORT ADD CONFIG ROW +

Action: --

NAME				TOPIC	TAGS	LAST READ	LAST SET
USE_C				Описание первой группы настроек	["Основные"]	Aug. 13, 2024, 3:41 p.m.	June 20, 2024, 8:14 a.m.
TEST				Описание первой группы настроек	-	-	June 20, 2024, 8:14 a.m.
SECRET_SWITCH	false		Настройка не для экспорта!	Описание первой группы настроек	["Фичи"]	Aug. 13, 2024, 3:41 p.m.	June 20, 2024, 8:14 a.m.
FUEL_PRICES	{"92": 50.5, "95": 55.8}		Стоимость различных видов топлива	Описание первой группы настроек	["Основные"]	Aug. 13, 2024, 3:41 p.m.	June 20, 2024, 8:14 a.m.
FIRST_DAY_OF_WEEK	null		Первый день недели	Описание первой группы настроек	["Фичи"]	Aug. 13, 2024, 3:41 p.m.	Aug. 13, 2024, 3:30 p.m.

FILTER

Show counts

↓ By topic

All

Описание первой группы настроек

↓ By tags

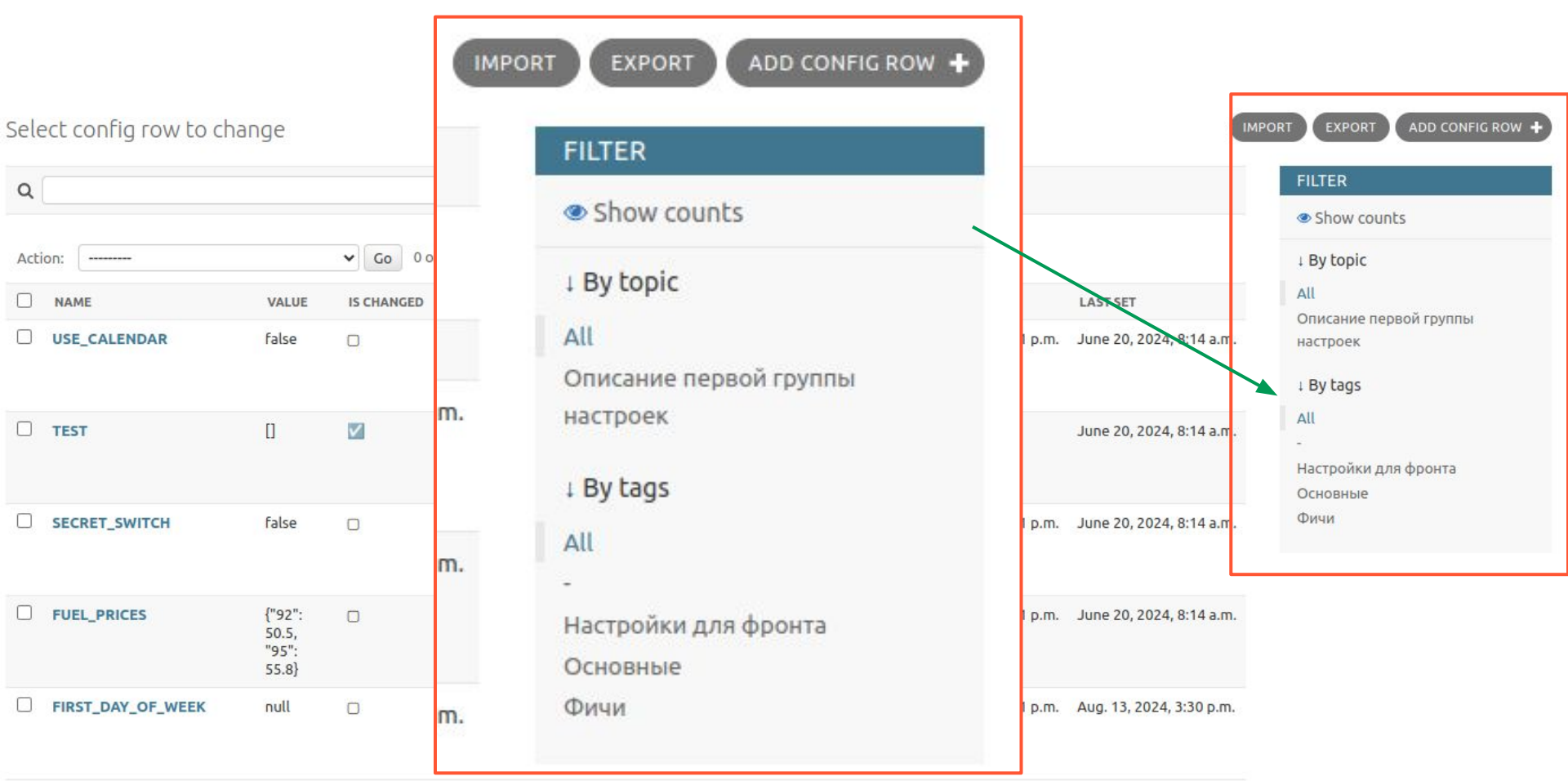
All

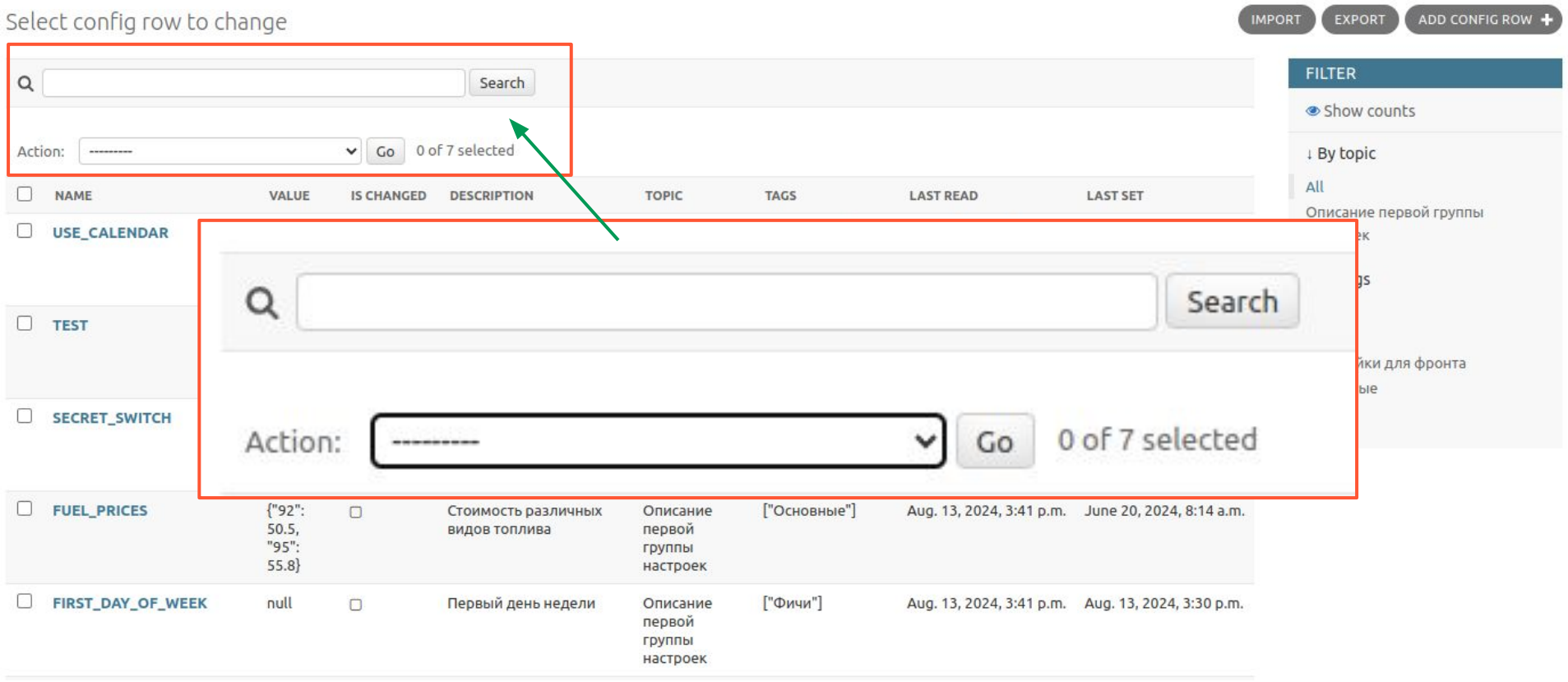
-

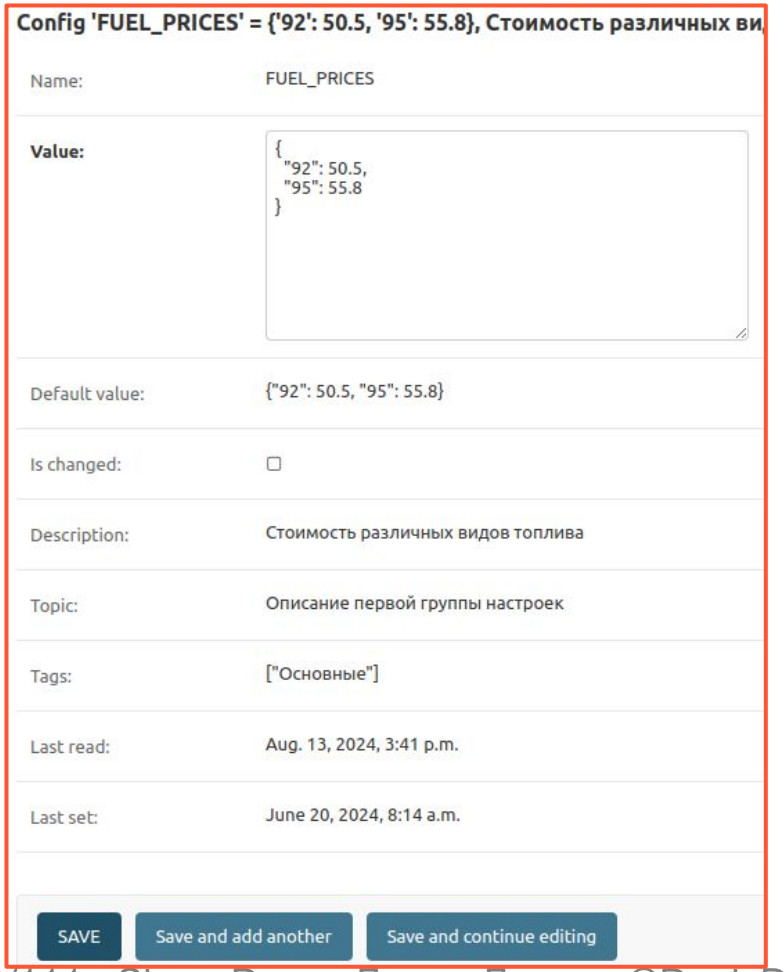
Настройки для фронта

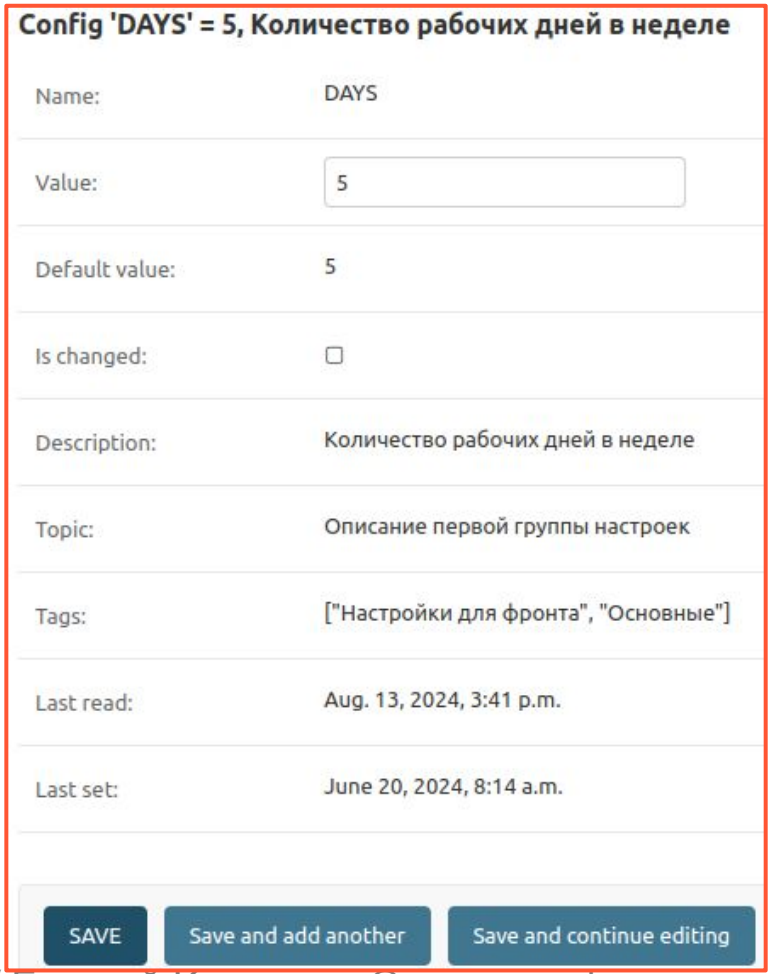
Основные

Фичи


IMPORT
EXPORT
ADD CONFIG ROW +
Select config row to change
Action:
Go
NAME
VALUE
IS CHANGED
LAST SET
USE_CALENDAR
false
June 20, 2024, 8:14 a.m.
TEST
[]
June 20, 2024, 8:14 a.m.
SECRET_SWITCH
false
June 20, 2024, 8:14 a.m.
FUEL_PRICES
{"92": 50.5, "95": 55.8}
June 20, 2024, 8:14 a.m.
FIRST_DAY_OF_WEEK
null
Aug. 13, 2024, 3:30 p.m.
FILTER
Show counts
↓ By topic
All
Описание первой группы настроек
↓ By tags
All
-
Настройки для фронта
Основные
Фичи


Select config row to change
IMPORT
EXPORT
ADD CONFIG ROW +
Search
Action:
Go
0 of 7 selected
FILTER
Show counts
By topic
All
Описание первой группы
NAME
VALUE
IS CHANGED
DESCRIPTION
TOPIC
TAGS
LAST READ
LAST SET
USE_CALENDAR
TEST
SECRET_SWITCH
FUEL_PRICES
{"92": 50.5, "95": 55.8}
Стоимость различных видов топлива
Описание первой группы настроек
["Основные"]
Aug. 13, 2024, 3:41 p.m.
June 20, 2024, 8:14 a.m.
FIRST_DAY_OF_WEEK
null
Первый день недели
Описание первой группы настроек
["Фичи"]
Aug. 13, 2024, 3:41 p.m.
Aug. 13, 2024, 3:30 p.m.


Config 'FUEL_PRICES' = {'92': 50.5, '95': 55.8}, Стоимость различных ви
Name:
FUEL_PRICES
Value:
{
"92": 50.5,
"95": 55.8
}
Default value:
{"92": 50.5, "95": 55.8}
Is changed:
Description:
Стоимость различных видов топлива
Topic:
Описание первой группы настроек
Tags:
["Основные"]
Last read:
Aug. 13, 2024, 3:41 p.m.
Last set:
June 20, 2024, 8:14 a.m.
SAVE
Save and add another
Save and continue editing


Config 'DAYS' = 5, Количество рабочих дней в неделе
Name:
DAYS
Value:
5
Default value:
5
Is changed:
Description:
Количество рабочих дней в неделе
Topic:
Описание первой группы настроек
Tags:
["Настройки для фронта", "Основные"]
Last read:
Aug. 13, 2024, 3:41 p.m.
Last set:
June 20, 2024, 8:14 a.m.
SAVE
Save and add another
Save and continue editing

Select history event to change

Search

Action: --------- Go 0 of 6 selected

	NAME	VALUE	EDIT AT	EDIT BY
☐	TEST	[]	June 20, 2024, 8:15 a.m.	user
☐	TEST	""	June 20, 2024, 8:15 a.m.	user
☐	TEST	{}	June 20, 2024, 8:15 a.m.	user
☐	TEST	"qqqqqqqqqqqqqqqqqqqqqqqq ... qqqqqqqqqqqqqqqqqqqqqqqq"	June 20, 2024, 8:15 a.m.	user
☐	TEST	"111"	June 20, 2024, 8:14 a.m.	user
☐	TEST	[]	June 20, 2024, 8:14 a.m.	user

6 history events

FILTER

Show counts

↓ By name

All

TEST

↓ By edit at

Any date

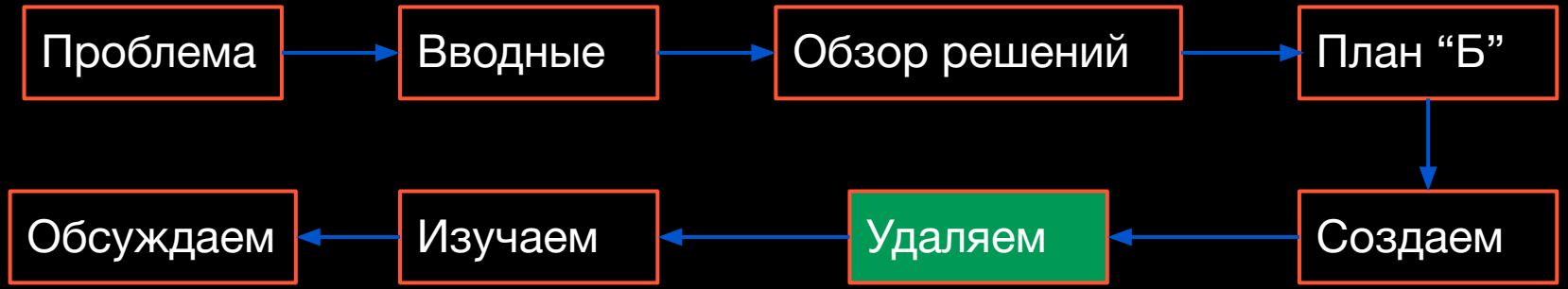
Today

Past 7 days

This month

This year

Удаление настроек



План удаления настройки

- Найти устаревшие настройки

План удаления настройки

- Найти устаревшие настройки
- Проверить код и подтвердить неактуальность настройки

План удаления настройки

- Найти устаревшие настройки
- Проверить код и подтвердить неактуальность настройки
- Удалить настройку из кода

План удаления настройки

- Найти устаревшие настройки
- Проверить код и подтвердить неактуальность настройки
- Удалить настройку из кода
- Удалить настройку из файла конфига

План удаления настройки

- Найти устаревшие настройки
- Проверить код и подтвердить неактуальность настройки
- Удалить настройку из кода
- Удалить настройку из файла конфига
- Удалить настройку из БД

LAST READ	LAST SET
Aug. 13, 2024, 3:41 p.m.	June 20, 2024, 8:14 a.m.
-	June 20, 2024, 8:14 a.m.
Aug. 13, 2024, 3:41 p.m.	June 20, 2024, 8:14 a.m.

Select co

IMPORT EXPORT ADD CONFIG ROW +

Action: --

NAME				TOPIC	TAGS	LAST READ	LAST SET
USE_C				Описание первой группы настроек	["Основные"]	Aug. 13, 2024, 3:41 p.m.	June 20, 2024, 8:14 a.m.
TEST				Описание первой группы настроек	-	-	June 20, 2024, 8:14 a.m.
SECRET_SWITCH	false		Настройка не для экспорта!	Описание первой группы настроек	["Фичи"]	Aug. 13, 2024, 3:41 p.m.	June 20, 2024, 8:14 a.m.
FUEL_PRICES	{"92": 50.5, "95": 55.8}		Стоимость различных видов топлива	Описание первой группы настроек	["Основные"]	Aug. 13, 2024, 3:41 p.m.	June 20, 2024, 8:14 a.m.
FIRST_DAY_OF_WEEK	null		Первый день недели	Описание первой группы настроек	["Фичи"]	Aug. 13, 2024, 3:41 p.m.	Aug. 13, 2024, 3:30 p.m.

FILTER

Show counts

↓ By topic

All

Описание первой группы настроек

↓ By tags

All

-

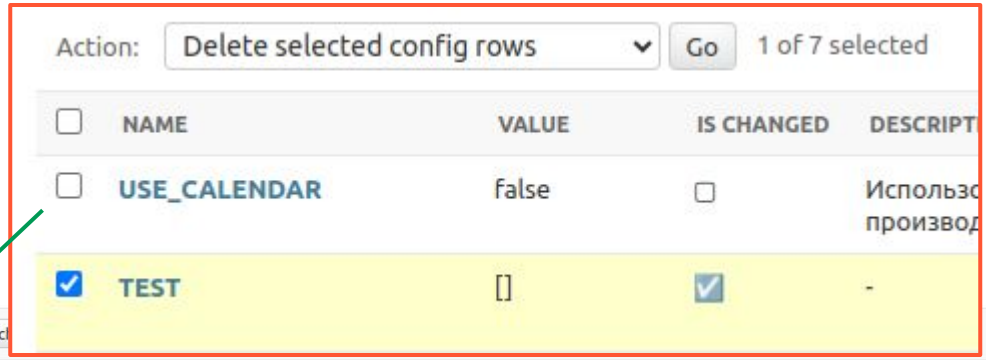
Настройки для фронта

Основные

Фичи

```python
class FirstExample(models.BaseConfig):
    ...
    TEST: float = 5.0

    FUEL_PRICES: dict[str, float] = {
        "92": 50.5,
        "95": 55.8,
    }
    FUEL_PRICES_DESCRIPTION = "Стоимость видов топлива"
    FUEL_PRICES_TAGS = [ConfigTags.basic]
    FUEL_PRICES_VALIDATORS = [dict_values_are(greater_than(0))]
```

Удаляем настройку
из файла конфигов



Select config row to change

IMPORT EXPORT ADD CONFIG ROW +

Search

Action: Delete selected config rows Go 1 of 7 selected

NAME	VALUE	IS CHANGED	DESCRIPTION	TOPIC	TAGS	LAST READ	LAST SET
USE_CALENDAR	false	☐	Использовать производственный календарь	Описание первой группы настроек	["Основные"]	Aug. 13, 2024, 3:41 p.m.	June 20, 2024, 8:14 a.m.
TEST	[]	☑	-	Описание первой группы настроек	-	-	June 20, 2024, 8:14 a.m.
SECRET_SWITCH	false	☐	Настройка не для экспорта!	Описание первой группы настроек	["Фичи"]	Aug. 13, 2024, 3:41 p.m.	June 20, 2024, 8:14 a.m.
FUEL_PRICES	{"92": 50.5, "95": 55.8}	☐	Стоимость различных видов топлива	Описание первой группы настроек	["Основные"]	Aug. 13, 2024, 3:41 p.m.	June 20, 2024, 8:14 a.m.
FIRST_DAY_OF_WEEK	null	☐	Первый день недели	Описание первой группы настроек	["Фичи"]	Aug. 13, 2024, 3:41 p.m.	Aug. 22, 2024, 10:11 a.m.

FILTER

Show counts

↓ By topic

All

Описание первой группы настроек

↓ By tags

All

-

Настройки для фронта

Основные

Фичи

```
$ docker compose exec django bash
root@5f0928858731:/app# ./manage.py delete_unused_configs
Будут удалены 1 конфигов:
TEST
Вы точно хотите удалить конфиги из списка выше? (y/n)
y
Неиспользуемые конфиги удалены
root@5f0928858731:/app#
```

Select config row to change

IMPORT EXPORT ADD CONFIG ROW +

Action: --------- Go 0 of 6 selected

NAME	VALUE	IS CHANGED	DESCRIPTION	TOPIC	TAGS	LAST READ	LAST SET
USE_CALENDAR	false	☐	Использовать производственнный календарь	Описание первой группы настроек			
SECRET_SWITCH	false	☐	Настройка не для экспорта!	Описа... групп...			
FUEL_PRICES	{"92": 50.5, "95": 55.8}	☐	Стоимость различных видов топлива				
FIRST_DAY_OF_WEEK	null	☐	Первый день недели				

FILTER

Show counts

↓ By topic

All

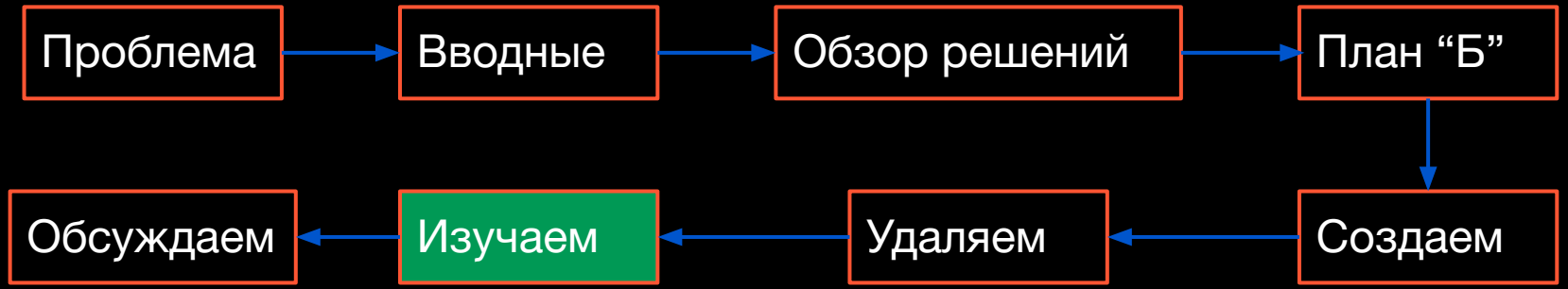
Описание первой группы настроек

...ags

...йки для фронта

Настройка удалена из БД

Заглянем под капот!



Какой файл смотрим

https://github.com/liveconfigs/django-liveconfigs/blob/main/src/liveconfigs/models/descriptors.py


ConfigMeta
удаляет "лишние" атрибуты класса
навешивает дескрипторы на атрибуты класса
ConfigRowDescriptor
получает значение из БД и кеширует их на время
записывает значение по-умолчанию и метаданные в БД при выкатке
Database
BaseConfig
служит базовым классом для групп настроек
MyConfig
USE_FEATURE1: bool
USE_FEATURE2: bool
SOME_FEATURE1_VALUE: float

```python
class ConfigRowDescriptor:
  """ Кеширующий дескриптор для работы с конфигами """

  def __init__(self, config_name, ...):
    ...

  def __get__(self, obj, klass=None):
    ...
    db_row = ConfigRow.objects.get(name=self.config_name)
    ...
    config_row_update_signal.send(...)
    return self.last_value
```

```
class ConfigMeta(type):
  """ Метакласс. Подменяет все атрибуты на дескрипторы """

  def __new__(cls, name, bases, dct):
    for n, v in dct.items():
      # если этот атрибут - настройка, а не ее описание, то
        dct[n] = ConfigRowDescriptor(n, ...)
    ...
    # удаляем вспомогательные атрибуты вроде *_DESCRIPTION
    ...
    c = super().__new__(cls, name, bases, dct)
    return c
```

```
class BaseConfig(metaclass=ConfigMeta):
    """От этого класса можно наследовать конфиги.
    За значениями этих конфигов система будет обращаться к БД
    и фоллбечиться на значения атрибутов, указанные в самом
    классе"""
```

```
class FirstExample(models.BaseConfig):
  __topic__ = "Описание первой группы настроек"
  __exported__ = ["FUEL_PRICES",]

  FUEL_PRICES: dict[str, float] = {
    "92": 50.5,
    "95": 55.8,
  }
  FUEL_PRICES_DESCRIPTION = "Стоимость видов топлива"
  FUEL_PRICES_TAGS = [ConfigTags.basic]
  FUEL_PRICES_VALIDATORS = [dict_values_are(greater_than(0))]
```

Итоги

Итоги

- Добавление новой настройки занимает минуту

Итоги

- Добавление новой настройки занимает минуту
- Всем понятно, где ее искать

Итоги

- Добавление новой настройки занимает минуту
- Всем понятно, где ее искать
- В пике на одном из сервисов было 220 настроек

Выводы

- Если настройки добавлять легко, их будут добавлять

Выводы

- Если настройки добавлять легко, их будут добавлять
- Переносить принятие решений ближе к заказчику - хорошо

Выводы

- Если настройки добавлять легко, их будут добавлять
- Переносить принятие решений ближе к заказчику - хорошо
- Хранить описание настроек в самом сервисе — хорошо

Выводы

- Если настройки добавлять легко, их будут добавлять
- Переносить принятие решений ближе к заказчику - хорошо
- Хранить описание настроек в самом сервисе — хорошо
- Хранить описание настроек в виде кода — хорошо

Выводы

- Если настройки добавлять легко, их будут добавлять
- Переносить принятие решений ближе к заказчику - хорошо
- Хранить описание настроек в самом сервисе — хорошо
- Хранить описание настроек в виде кода — хорошо
- Писать велосипеды иногда полезно

Куда дальше?

- «Заморозка» значений

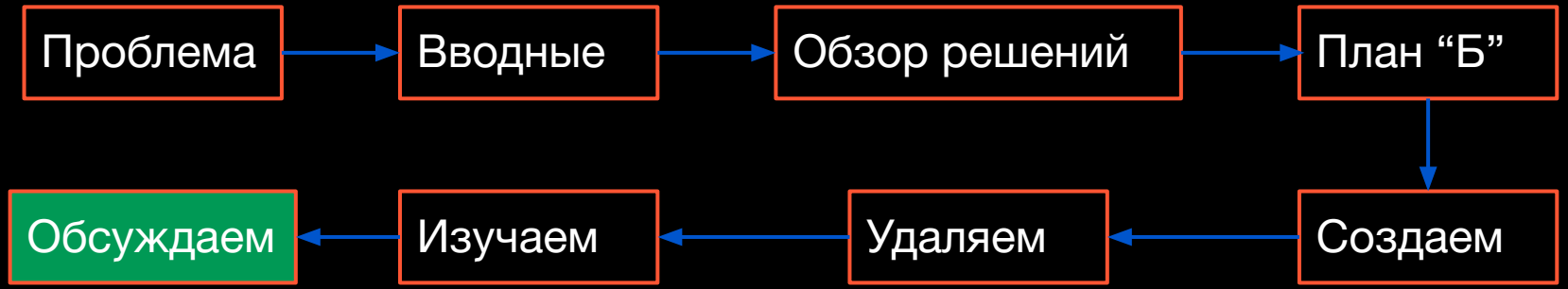
Куда дальше?

- «Заморозка» значений
- Асинхронная работа

Куда дальше?

- «Заморозка» значений
- Асинхронная работа
- Использовать Annotated вместо _DESCRIPTION и подобного

Спасибо за внимание!



Денис Дудник

- https://t.me/DenisDudnik

Евгений Кацевман

- https://t.me/eugene_mk

- https://bit.ly/django-liveconfigs-piterpy
- https://github.com/liveconfigs/django-liveconfigs
- https://pypi.org/project/django-liveconfigs/