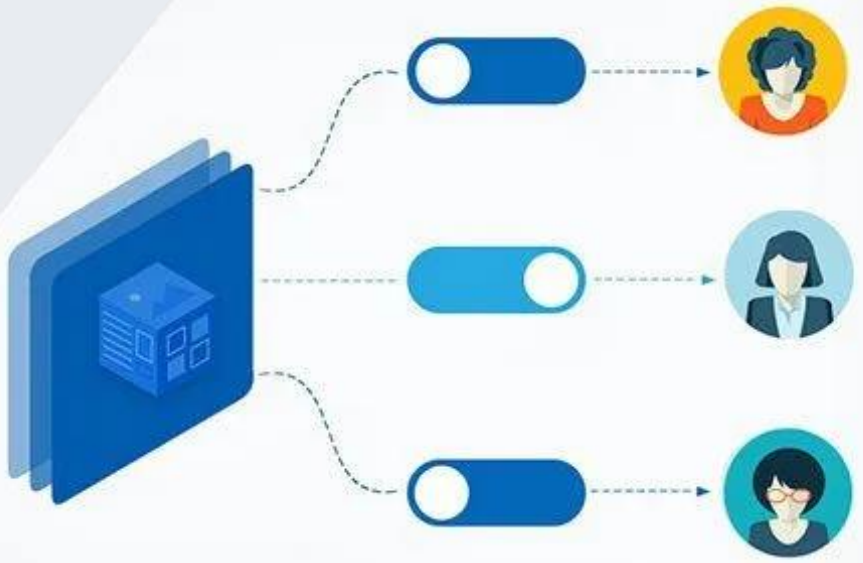
RUDN university

СТАТИЧЕСКИЕ И ДИНАМИЧЕСКИЕ ПАРАМЕТРЫ

КОНФИГУРАЦИЯ, ОКРУЖЕНИЕ, FEATURE TOGGLE



Config Management

СТАТИЧЕСКИЕ ПАРАМЕТРЫ

```
41    # Максимальная длина текста вопроса пользователя (в символах)
42    MAX_PROMPT_CHARS_DEFAULT = 600
43
44    # Показывать ли в ответе модели доп.инфо
45    SHOW_MODEL_FOOTER_DEFAULT = True
```

В файл **config.py** добавляем два статических параметра:

MAX_PROMPT_CHARS_DEFAULT – *максимальная длина текста вопроса пользователя*

SHOW_MODEL_FOOTER_DEFAULT – *показывать ли в ответе модели доп.инфо*

СТАТИЧЕСКИЕ ПАРАМЕТРЫ

В файле **main.py** для команды /ask используем созданные ранее параметры:

- MAX_PROMPT_CHARS_DEFAULT
- SHOW_MODEL_FOOTER_DEFAULT

```python
from config import TOKEN, MAX_PROMPT_CHARS_DEFAULT, SHOW_MODEL_FOOTER_DEFAULT

@bot.message_handler(commands=["ask"])
def cmd_ask(message: types.Message) -> None:
    """
    Задать вопрос LLM модели
    """
    metric.counter("commands_total").inc()
    metric.counter("ask_requests_total").inc()

    user_id = message.from_user.id
    q = message.text.replace("/ask", "", 1).strip()
    if not q:
        bot.reply_to(message, "Использование: /ask <вопрос>")
        return

    msgs = _build_messages(user_id, q[:MAX_PROMPT_CHARS_DEFAULT])
    model_key = get_active_model()["key"]

    log.info("Команда /ask от user_id=%s, вопрос=%.80s", user_id, q)

    try:
        text, ms = chat_once(msgs, model=model_key, temperature=0.2, max_tokens=400)

    except OpenRouterError as e:
        metric.counter("openrouter_errors_total").inc()

        log.error("OpenRouterError при /ask от user_id=%s: %s", user_id, e)
        write_error_log(
            level="ERROR",
            logger_name=__name__,
            message=str(e),
            user_id=user_id,
            command="/ask",
            details=None,
        )
        bot.reply_to(message, f"Ошибка: {e}")
        return
    except Exception:
        log.exception("Непредвиденная ошибка при /ask от user_id=%s", user_id)
        write_error_log(
            level="ERROR",
            logger_name=__name__,
            message=f"Unhandled error in /ask: {e}",
            user_id=user_id,
            command="/ask",
            details=None,
        )
        bot.reply_to(message, "Непредвиденная ошибка.")
        return

    metric.latency("openrouter_latency_ms").observe(ms)

    out = (text or "").strip()[:4000]  # не переполняем сообщение Telegram

    if SHOW_MODEL_FOOTER_DEFAULT:
        bot.reply_to(message, f"{out}\n\n({ms} мс; модель: {model_key})")
```

ДИНАМИЧЕСКИЕ ПАРАМЕТРЫ

```
-- Динамические параметры
CREATE TABLE IF NOT EXISTS settings (
        key   TEXT PRIMARY KEY,
        value TEXT NOT NULL
    );
```

В файл **db.py** добавляем запрос создания таблицы динамических параметров

ДИНАМИЧЕСКИЕ ПАРАМЕТРЫ

В файл **db.py** добавляем методы получения динамических параметров

```python
def get_setting_or_default(key: str, default: str) -> str:  2 usages  new *
    """
    Вернуть динамический параметр по ключу.
    Если параметра нет — вернуть default.
    """
    with _connect() as conn:
        row = conn.execute(
            sql: "SELECT value FROM settings WHERE key = ?",
            parameters: (key,),
        ).fetchone()
    if row is None:
        return default
    return row["value"]


def get_int_setting(key: str, default: int) -> int:  3 usages  new *
    """
    Вернуть динамический параметр по ключу в виде int.
    """
    raw = get_setting_or_default(key, str(default))
    try:
        return int(raw)
    except ValueError:
        return default


def get_bool_setting(key: str, default: bool) -> bool:  3 usages  new *
    """
    Вернуть динамический параметр по ключу в виде bool.
    """
    raw = get_setting_or_default(key, "true" if default else "false")
    raw_low = raw.lower()
    if raw_low in ("1", "true", "yes", "on"):
        return True
    if raw_low in ("0", "false", "no", "off"):
        return False
    return default
```

ДИНАМИЧЕСКИЕ ПАРАМЕТРЫ

В файле **main.py** для команды /ask вносим изменения, заменив статические параметры на динамические

```python
from db import (get_bool_setting)

def cmd_ask(message: types.Message) -> None:
    """
    Задать вопрос LLM модели
    """
    metric.counter("commands_total").inc()
    metric.counter("ask_requests_total").inc()

    user_id = message.from_user.id
    q = message.text.replace("/ask", "", 1).strip()
    if not q:
        bot.reply_to(message, "Использование: /ask <вопрос>")
        return

    max_len = get_int_setting("max_prompt_chars", MAX_PROMPT_CHARS_DEFAULT)

    msgs = _build_messages(user_id, q[:max_len])
    model_key = get_active_model()["key"]

    log.info("Команда /ask от user_id=%s, вопрос=%.80s", user_id, q)

    try:
        text, ms = chat_once(msgs, model=model_key, temperature=0.2, max_tokens=400)

    except OpenRouterError as e:
        metric.counter("openrouter_errors_total").inc()

        log.error("OpenRouterError при /ask от user_id=%s: %s", user_id, e)
        write_error_log(
            level="ERROR",
            logger_name=__name__,
            message=str(e),
            user_id=user_id,
            command="/ask",
            details=None,
        )
        bot.reply_to(message, f"Ошибка: {e}")
        return
    except Exception as e:
        log.exception("Непредвиденная ошибка при /ask от user_id=%s", user_id)
        write_error_log(
            level="ERROR",
            logger_name=__name__,
            message=f"Unhandled error in /ask: {e}",
            user_id=user_id,
            command="/ask",
            details=None,
        )
        bot.reply_to(message, "Непредвиденная ошибка.")
        return

    metric.latency("openrouter_latency_ms").observe(ms)

    out = (text or "").strip()[:4000]  # не переполняем сообщение Telegram

    show_footer = get_bool_setting("show_model_footer", SHOW_MODEL_FOOTER_DEFAULT)
    add_info = f"\n\n({ms} мс; модель: {model_key})" if show_footer else ""
    bot.reply_to(message, f"{out}{add_info}")
```

FEATURE TOGGLES

```
47    # ---------- Feature Toggles ------------
48
49    # Показать настройки бота
50    DEBUG_SETTINGS_SHOW = False
51
52    # Использование команды смены модели
53    CMD_MODEL_ID_ENABLED = True
```

В файл **config.py** добавляем два статических параметра:

DEBUG_SETTINGS_SHOW – *показать настройки бота*

CMD_MODEL_ID_ENABLED – *использовать команду смены модели /model <ID>*

FEATURE TOGGLES

```
-- Feature Toggles
CREATE TABLE IF NOT EXISTS feature_toggles (
        name    TEXT PRIMARY KEY,
        enabled INTEGER NOT NULL CHECK (enabled IN (0, 1))
    );
```

В файл **db.py** добавляем запрос создания таблицы Feature Toggles

FEATURE TOGGLES

В файл **db.py** добавляем метод получения значения фиче-тогглов

```
569  def is_feature_enabled(name: str, default: bool) -> bool:  4 usages  new *
570      """
571      Вернуть состояние фиче-тоггла по имени.
572      Если записи нет — вернуть default.
573      """
574      with _connect() as conn:
575          row = conn.execute(
576              sql: "SELECT enabled FROM feature_toggles WHERE name = ?",
577              parameters: (name,),
578          ).fetchone()
579      if row is None:
580          return default
581      return bool(row["enabled"])
```

FEATURE TOGGLES

В файл **main.py** добавляем

- Импорты
- Команду отображения настроек

```python
from config import DEBUG_SETTINGS_SHOW, CMD_MODEL_ID_ENABLED
import db

from telebot import types
from openrouter_client import chat_once, OpenRouterError
from db import (is_feature_enabled)
```

```python
@bot.message_handler(commands=["debug_settings"])  new *
def cmd_debug_settings(message):
    """
    Показывает настройки, если фиче-тоггл debug_settings включен.
    Иначе сообщает, что команда отключена.
    """
    max_len = get_int_setting( key: "max_prompt_chars", MAX_PROMPT_CHARS_DEFAULT)
    show_footer = get_bool_setting( key: "show_model_footer", SHOW_MODEL_FOOTER_DEFAULT)
    model_cmds = is_feature_enabled( name: "model_commands", CMD_MODEL_ID_ENABLED)

    text = (
        f"max_prompt_chars = {max_len}\n"
        f"show_model_footer = {show_footer}\n"
        f"feature: model_commands = {model_cmds}\n"
    )
    bot.reply_to(message, text)
```

FEATURE TOGGLES

В файл **main.py** добавляем

- Условие проверки Feature toggle для команды отображения настроек
- Условие проверки Feature toggle для команды выбора модели

```python
def _setup_bot_commands() -> None:
    """
    Регистрирует команды в меню клиента Telegram (удобно для новичков).
    """
    cmds = [
        types.BotCommand("start", "Приветствие и помощь"),
        types.BotCommand("note_add", "Добавить заметку"),
        types.BotCommand("note_list", "Список заметок"),
        types.BotCommand("note_find", "Поиск заметок"),
        types.BotCommand("note_edit", "Изменить заметку"),
        types.BotCommand("note_del", "Удалить заметку"),
        types.BotCommand("note_count", "Сколько заметок"),
        types.BotCommand("note_export", "Экспорт заметок в .txt"),
        types.BotCommand("note_stats", "Статистика по датам"),
        types.BotCommand("model", "Установить активную модель"),
        types.BotCommand("models", "Получить список моделей"),
        types.BotCommand("ask", "Задать вопрос модели"),
        types.BotCommand("ask_random", "Задать вопрос случайной модели"),
        types.BotCommand("character", "Установить активного персонажа"),
        types.BotCommand("characters", "Получить список персонажей"),
        types.BotCommand("whoami", "Получить активную модель и активного персонажа"),
        types.BotCommand("stats", "Мониторинг бота"),
    ]
    if is_feature_enabled("debug_settings", DEBUG_SETTINGS_SHOW):
        cmds.append(types.BotCommand("debug_settings", "Показать настройки бота"))

    bot.set_my_commands(cmds)
```

```python
@bot.message_handler(commands=["model"])
def cmd_model(message: types.Message) -> None:
    """
    Установить активной LLM модель
    """
    if not is_feature_enabled("model_commands", CMD_MODEL_ID_ENABLED):
        bot.reply_to(message, "Команды выбора модели временно отключены.")
        return

    metric.counter("commands_total").inc()
    metric.counter("model_requests_total").inc()
    arg = message.text.replace("/model", "", 1).strip()
    if not arg:
        active = get_active_model()
        bot.reply_to(message, f"Текущая активная модель: {active['label']} [{active['key
        return
    if not arg.isdigit():
        bot.reply_to(message, "Использование: /model <ID из /models>")
        return
    try:
        active = set_active_model(int(arg))
        bot.reply_to(message, f"Активная модель переключена: {active['label']} [{active['
    except ValueError:
        bot.reply_to(message, "Неизвестный ID модели. Сначала /models.")
```

РЕДАКТИРОВАНИЕ НАСТРОЕК

В файл **db.py** добавляем методы установки значений динамических параметров и фиче-тогглов

```python
def set_setting(key: str, value: str) -> None:  2 usages  new *
    """
    Установить динамический параметр
    """
    with _connect() as conn:
        conn.execute(
            sql: "INSERT INTO settings (key, value) VALUES (?, ?) "
            "ON CONFLICT(key) DO UPDATE SET value = excluded.value",
            parameters: (key, value),
        )


def set_feature_toggle(name: str, enabled: bool) -> None:  2 usages  new *
    """
    Установить фиче-тоггл (enabled = 0/1).
    """
    with _connect() as conn:
        conn.execute(
            sql: """
            INSERT INTO feature_toggles (name, enabled)
            VALUES (?, ?)
            ON CONFLICT(name) DO UPDATE SET enabled = excluded.enabled
            """,
            parameters: (name, 1 if enabled else 0),
        )
```

РЕДАКТИРОВАНИЕ НАСТРОЕК

В файл **db.py** добавляем метод установки значений динамических параметров

```python
from db import (set_setting, set_feature_toggle)

@bot.message_handler(commands=["set_setting"])  new *
def cmd_set_setting(message: types.Message) -> None:
    """
    Админ-команда: установить динамический параметр в таблице settings.

    Формат:
      /set_setting ключ=значение

    Примеры:
      /set_setting max_prompt_chars=300
      /set_setting show_model_footer=false
    """

    parts = message.text.split(maxsplit=1)
    if len(parts) < 2 or "=" not in parts[1]:
        bot.reply_to(message, text: "Использование: /set_setting ключ=значение")
        return

    key, value = parts[1].split( sep: "=", maxsplit: 1)
    key = key.strip()
    value = value.strip()

    if not key:
        bot.reply_to(message, text: "Ключ параметра не может быть пустым.")
        return

    set_setting(key, value)
    bot.reply_to(message, text: f"Параметр {key} установлен в {value}")
```

РЕДАКТИРОВАНИЕ НАСТРОЕК

В файл **main.py** добавляем метод установки значений Feature Toggle

```python
@bot.message_handler(commands=["set_toggle"])  new *
def cmd_set_toggle(message: types.Message) -> None:
    """
    Админ-команда: включить/выключить фиче-тоггл в таблице feature_toggles.

    Формат:
      /set_toggle имя on|off

    Примеры:
      /set_toggle debug_settings on
      /set_toggle model_commands off
    """

    parts = message.text.split(maxsplit=2)
    if len(parts) < 3:
        bot.reply_to(message, text: "Использование: /set_toggle имя on|off")
        return

    name = parts[1].strip()
    state = parts[2].strip().lower()

    if state not in ("on", "off"):
        bot.reply_to(message, text: "Второй аргумент должен быть on или off.")
        return

    enabled = state == "on"
    set_feature_toggle(name, enabled)
    bot.reply_to(message, text: f"Feature-toggle {name} = {enabled}")
```