```
import logging
import random
from datetime import datetime, timedelta
from telegram import (
    Update,
    InlineKeyboardButton,
    InlineKeyboardMarkup,
    InputMediaPhoto,
)
from telegram.ext import (
    ApplicationBuilder,
    CommandHandler,
    CallbackQueryHandler,
    ContextTypes,
)
from tinydb import TinyDB, Query


# Настройка логирования
logging.basicConfig(format='%(asctime)s - %(message)s',
level=logging.INFO)

# База данных
db = TinyDB('bot_data.json')
players = Query()

# Константы
CARDS = [
    #обычные
    {"name": "Обычный банан", "rarity": "❄Обычная❄",
"drop_chance": 10, "rubies": 5, "image":
"/Users/mizincik/Desktop/ruby_bot/images/5.jpg"},
    {"name": "Обычный бибизян", "rarity": "❄Обычная",
"drop_chance": 10, "rubies": 5, "image":
"/Users/mizincik/Desktop/ruby_bot/images/8.jpg"},
    {"name": "Банан на крючке", "rarity": "❄Обычная",
"drop_chance": 10, "rubies": 5, "image":
"/Users/mizincik/Desktop/ruby_bot/images/7.jpg"},
    {"name": "Бибизян рыбак", "rarity": "❄Обычная", "drop_chance":
10, "rubies": 5, "image":
"/Users/mizincik/Desktop/ruby_bot/images/6.jpg"},
    {"name": "Угольный банан", "rarity": "❄Обычная",
"drop_chance": 10, "rubies": 5, "image":
"/Users/mizincik/Desktop/ruby_bot/images/1.jpg"},
    {"name": "Бибизян шахтер", "rarity": "❄Обычная",
"drop_chance": 10, "rubies": 5, "image":
"/Users/mizincik/Desktop/ruby_bot/images/2.jpg"},
```

```
    {"name": "Банан правопорядка", "rarity": "❄️Обычная",
"drop_chance": 10, "rubies": 5, "image":
"/Users/mizincik/Desktop/ruby_bot/images/4.jpg"},
    {"name": "Бибизян полицейский", "rarity": "❄️Обычная",
"drop_chance": 10, "rubies": 5, "image":
"/Users/mizincik/Desktop/ruby_bot/images/3.jpg"},
    #редкие
    {"name": "Пиратский банан", "rarity": "💎Редкая",
"drop_chance": 25, "rubies": 50, "image":
"/Users/mizincik/Desktop/ruby_bot/images/1.1.jpg"},
    {"name": "Бибизян пират", "rarity": "💎Редкая", "drop_chance":
25, "rubies": 50, "image":
"/Users/mizincik/Desktop/ruby_bot/images/1.4.jpg"},
    {"name": "Важный банан", "rarity": "💎Редкая", "drop_chance":
25, "rubies": 50, "image":
"/Users/mizincik/Desktop/ruby_bot/images/1.2.jpg"},
    {"name": "Бибизян политик", "rarity": "💎Редкая",
"drop_chance": 25, "rubies": 50, "image":
"/Users/mizincik/Desktop/ruby_bot/images/1.3.jpg"},
    {"name": "Воинственный банан", "rarity": "💎Редкая",
"drop_chance": 25, "rubies": 50, "image":
"/Users/mizincik/Desktop/ruby_bot/images/1.5.jpg"},
    {"name": "Бибизян рыцарь", "rarity": "💎Редкая", "drop_chance":
25, "rubies": 50, "image":
"/Users/mizincik/Desktop/ruby_bot/images/1.6.jpg"},
    #эпические
    {"name": "Бибизян мафиози", "rarity": "🔮Эпическая",
"drop_chance": 15, "rubies": 300, "image":
"/Users/mizincik/Desktop/ruby_bot/images/2.1.jpg"},
    {"name": "Банан-револьвер", "rarity": "🔮Эпическая",
"drop_chance": 15, "rubies": 300, "image":
"/Users/mizincik/Desktop/ruby_bot/images/2.2.jpg"},
    {"name": "Богатый бибизян", "rarity": "🔮Эпическая",
"drop_chance": 15, "rubies": 300, "image":
"/Users/mizincik/Desktop/ruby_bot/images/2.3.jpg"},
    {"name": "Дорогой банан", "rarity": "🔮Эпическая",
"drop_chance": 15, "rubies": 300, "image":
"/Users/mizincik/Desktop/ruby_bot/images/2.4.jpg"},
    #легендарные
    {"name": "Банан огненного самуря", "rarity": "🏅Легендарная",
"drop_chance": 2, "rubies": 1000, "image":
"/Users/mizincik/Desktop/ruby_bot/images/3.3.jpg"},
    {"name": "Огненный самурай бибизян", "rarity": "🏅Легендарная",
"drop_chance": 2, "rubies": 1000, "image":
"/Users/mizincik/Desktop/ruby_bot/images/3.8.jpg"},
    {"name": "Обсидиановый банан", "rarity": "🏅Легендарная",
"drop_chance": 2, "rubies": 1000, "image":
"/Users/mizincik/Desktop/ruby_bot/images/3.1.jpg"},
```

```
    {"name": "Обсидиановый бибизян", "rarity": "🏅Легендарная",
"drop_chance": 2, "rubies": 1000, "image":
"/Users/mizincik/Desktop/ruby_bot/images/3.2.jpg"},
    {"name": "Алмазный банан", "rarity": "🏅Легендарная",
"drop_chance": 2, "rubies": 1000, "image":
"/Users/mizincik/Desktop/ruby_bot/images/3.4.jpg"},
    {"name": "Алмазный бибизян", "rarity": "🏅Легендарная",
"drop_chance": 2, "rubies": 1000, "image":
"/Users/mizincik/Desktop/ruby_bot/images/3.6.jpg"},
    {"name": "Рубиновый банан", "rarity": "🏅Легендарная",
"drop_chance": 2, "rubies": 1000, "image":
"/Users/mizincik/Desktop/ruby_bot/images/3.5.jpg"},
    {"name": "Рубиновый бибизян", "rarity": "🏅Легендарная",
"drop_chance": 2, "rubies": 1000, "image":
"/Users/mizincik/Desktop/ruby_bot/images/3.7.jpg"},
    #рубиновые
    {"name": "Богоподобный бибизян", "rarity": "💥Рубиновая",
"drop_chance": 1, "rubies": 25000, "image":
"/Users/mizincik/Desktop/ruby_bot/images/4.1.jpg"},

    #эксклюзивные
    {"name": "Рубин таинственных джунглей", "rarity":
"🏵️Эксклюзивная", "drop_chance": 0, "rubies": 50000, "image":
"/Users/mizincik/Desktop/ruby_bot/images/5.1.jpg"},
]
RARITY_ORDER = ["Обычная", "Редкая", "Эпическая", "Легендарная",
"Рубиновая", "Эксклюзивная"]
FARM_COOLDOWN = 60 * 60  # В секундах (3 часа)


def get_time_remaining(last_time):
    """Вычислить оставшееся время кулдауна."""
    now = datetime.now()
    next_time = last_time + timedelta(seconds=FARM_COOLDOWN)
    return max(0, (next_time - now).total_seconds())


# Вспомогательные функции
def get_player_data(user_id, chat_id, username=None):
    """Получить данные игрока из базы или создать их, если их нет."""
    user_data = db.get((players.user_id == user_id) &
(players.chat_id == chat_id))
    if not user_data:
        user_data = {
            "user_id": user_id,
            "chat_id": chat_id,
            "username": username,  # Сохраняем username для
удобства
```

```
            "rubies": 0,
            "cards": [],
            "last_farm_time": None
        }
        db.insert(user_data)
    return user_data


def update_player_data(user_id, chat_id, data):
    """Обновить данные игрока в базе."""
    db.update(data, (players.user_id == user_id) & (players.chat_id
== chat_id))



def pick_card():
    """Выбрать случайную карту в соответствии с шансами выпадения."""
    roll = random.uniform(0, 100)
    cumulative = 0
    for card in CARDS:
        cumulative += card["drop_chance"]
        if roll <= cumulative:
            return card
    return CARDS[0]  # По умолчанию возвращаем обычную карту


# Команды
async def start(update: Update, context:
ContextTypes.DEFAULT_TYPE):
    """Приветствие при старте бота."""
    await update.message.reply_text(
        "🐵Добро пожаловать в <strong>банового
бота!</strong>\n\nИспользуйте /banana, чтобы фармить
бананы\n/profile, чтобы просматривать профиль\n/leaderboard, чтобы
просматривать рейтинг чата.",
        parse_mode="HTML")


async def leaderboard(update: Update, context:
ContextTypes.DEFAULT_TYPE):
    """Показать рейтинг игроков чата по количеству рубинов и
эксклюзивных карт."""
    chat_id = update.effective_chat.id
    all_players = db.search(players.chat_id == chat_id)  #
Фильтруем по текущему чату

    sorted_players = sorted(
        all_players,
```

```
        key=lambda p: (p["rubies"], sum(1 for card in p["cards"] if
card["rarity"] == "Эксклюзивная")),
        reverse=True
    )

    leaderboard_message = "🏆 <b>Таблица лидеров:</b>\n\n"
    for idx, player in enumerate(sorted_players[:10]):
        username = player.get("username")
        user_id = player["user_id"]
        rubies = player["rubies"]
        exclusive_cards = sum(1 for card in player["cards"] if
card["rarity"] == "Эксклюзивная")

        # Формируем гиперссылку на профиль
        if username:
            player_link = f"<a
href='tg://user?id={user_id}'>{username}</a>"
        else:
            player_link = f"<a href='tg://user?id={user_id}'>ID
{user_id}</a>"

        leaderboard_message += (
            f"{idx + 1}. {player_link} — ♦Рубины: {rubies}, "
            f"🎫Эксклюзивных карт: {exclusive_cards}\n"
        )

    await update.message.reply_text(leaderboard_message,
parse_mode="HTML")


async def profile(update: Update, context:
ContextTypes.DEFAULT_TYPE):
    """Просмотр профиля игрока."""
    user = update.effective_user
    user_id = user.id
    user_data = get_player_data(user_id, update.effective_chat.id)

    ruby_cards_count = sum(1 for card in user_data["cards"] if
card["rarity"] == "Рубиновая")

    photos = await context.bot.get_user_profile_photos(user_id)
    photo_file_id = photos.photos[0][0].file_id if
photos.total_count > 0 else None

    keyboard = [[InlineKeyboardButton("🀄Мои карточки",
callback_data="my_cards")]]
    reply_markup = InlineKeyboardMarkup(keyboard)
```

```
    profile_message = (
        f"🔖Ваш профиль:\n\n"
        f"♦Рубины: {user_data['rubies']}\n\n"
        f"🃏Всего карт: {len(user_data['cards'])}\n\n"
        f"🏵Карт редкости «Рубиновая»: {ruby_cards_count}\n\n"
    )

    if photo_file_id:
        await update.message.reply_photo(photo=photo_file_id,
caption=profile_message, reply_markup=reply_markup)
    else:
        await update.message.reply_text(text=profile_message,
reply_markup=reply_markup)


# Ваш Telegram ID
DEVELOPER_ID = 497660857  # Замените на ваш Telegram ID

async def give_card(update: Update, context:
ContextTypes.DEFAULT_TYPE):
    """Выдача карты пользователю (доступно только разработчику)."""
    user = update.effective_user
    if user.id != DEVELOPER_ID:
        await update.message.reply_text("У вас нет прав на
использование этой команды.")
        return

    # Проверяем наличие аргументов
    if len(context.args) < 2:
        await update.message.reply_text(
            "Использование: /give_card <user_id> <card_name>\n"
            "Пример: /give_card 123456789 'Банан-3'"
        )
        return

    try:
        target_user_id = int(context.args[0])  # ID целевого
пользователя
        card_name = " ".join(context.args[1:])  # Название карты
    except ValueError:
        await update.message.reply_text("Укажите корректный ID
пользователя.")
        return

    # Проверяем, существует ли карта с таким названием
    card = next((card for card in CARDS if card["name"] ==
card_name), None)
    if not card:
```

```
        await update.message.reply_text(f"Карта с названием
'{card_name}' не найдена.")
        return

    # Получаем данные целевого пользователя
    user_data = db.get(players.user_id == target_user_id)
    if not user_data:
        await update.message.reply_text(f"Пользователь с ID
{target_user_id} не найден.")
        return

    # Проверяем, есть ли карта уже у пользователя
    if any(existing_card["name"] == card_name for existing_card in
user_data["cards"]):
        await update.message.reply_text(f"У пользователя уже есть
карта '{card_name}'.")
        return

    # Добавляем карту и обновляем данные пользователя
    user_data["cards"].append({
        "name": card["name"],
        "rarity": card["rarity"],
        "rubies": card["rubies"],
        "image": card.get("image", ""),
    })
    db.update(user_data, players.user_id == target_user_id)

    await update.message.reply_text(
        f"Карта '{card_name}' успешно выдана пользователю с ID
{target_user_id}."
    )




async def banana(update: Update, context:
ContextTypes.DEFAULT_TYPE):
    """Функция фарма бананов."""
    user_id = update.effective_user.id
    chat_id = update.effective_chat.id
    username = update.effective_user.username

    user_data = get_player_data(user_id, chat_id, username)

    # Проверка кулдауна
    last_farm_time = user_data.get("last_farm_time")
    if last_farm_time:
```

```
        remaining_time =
get_time_remaining(datetime.fromisoformat(last_farm_time))
        if remaining_time > 0:
            await update.message.reply_text(
                f"Попробуйте снова через
{timedelta(seconds=remaining_time)}."
            )
            return

    card = pick_card()

    # Проверяем, есть ли карта в профиле пользователя
    existing_card = next((existing_card for existing_card in
user_data["cards"] if existing_card["name"] == card["name"]),
None)

    if existing_card:
        # Если карта уже есть, добавляем только рубины
        user_data["rubies"] += card["rubies"]
        update_player_data(user_id, chat_id, user_data)
        await update.message.reply_text(
            f"🎱Вы нашли карту снова:\n"
            f"🃏Название: {card['name']}\n"
            f"💠Редкость: {card['rarity']}\n"
            f"♦Рубины: {card['rubies']}\n\n"
            f"🎗Ваши рубины увеличились на {card['rubies']}!"
        )
    else:
        # Если карты нет в профиле, добавляем её в список карт
        print(f"Добавляем карту: {card['name']} с изображением
{card.get('image', 'Нет изображения')}")
        user_data["cards"].append({
            "name": card["name"],
            "rarity": card["rarity"],
            "rubies": card["rubies"],
            "image": card.get("image", ""),  # Путь к изображению
(если есть)
        })
        user_data["rubies"] += card["rubies"]
        update_player_data(user_id, chat_id, user_data)

        # Отправляем изображение карты и информацию
        await update.message.reply_photo(
            photo=open(card["image"], 'rb') if card.get("image")
else None,
            caption=(
                f"✨Вы нашли карту:\n\n"
                f"🃏Название: {card['name']}\n\n"
```

```
                f"💠Редкость: {card['rarity']}\n\n"
                f"♦Рубины: {card['rubies']}\n\n"
                f"[{user_id}](tg://user?id={user_id}\n\n)"
            ),
            parse_mode="Markdown"
        )

    # Обновляем время последнего фарма
    user_data["last_farm_time"] = datetime.now().isoformat()
    update_player_data(user_id, user_data)

# Callback обработка
async def handle_callback(update: Update, context:
ContextTypes.DEFAULT_TYPE):
    """Обработка кнопок."""
    query = update.callback_query
    user_id = query.from_user.id
    user_data = get_player_data(user_id, update.effective_chat.id)

    if query.data == "my_cards":
        # Раздел выбора редкости
        keyboard = []
        for rarity in RARITY_ORDER:
            # Проверка, есть ли у пользователя хотя бы одна карта с
этой редкостью
            cards_of_rarity = [card for card in user_data["cards"]
if card["rarity"] == rarity]
            if cards_of_rarity:  # Если есть карты данной редкости
                keyboard.append([InlineKeyboardButton(rarity,
callback_data=f"rarity_{rarity}")])

        # Если карты есть
        if keyboard:
            reply_markup = InlineKeyboardMarkup(keyboard)
            try:
                # Попробуем редактировать сообщение, если оно еще
существует
                await query.edit_message_text("Выберите редкость:",
reply_markup=reply_markup)
            except Exception as e:
                # Логируем ошибку, если редактирование не удалось
                print(f"Ошибка редактирования сообщения: {e}")
                # Если редактирование не удалось, отправляем новое
сообщение
                await query.message.reply_text("Выберите
редкость:", reply_markup=reply_markup)
        else:
            # Если у пользователя нет карт вообще
```

```
            await query.edit_message_text("У вас нет карт.
Пожалуйста, заработайте карты с помощью команды /banana.")
    elif query.data.startswith("rarity_"):
        # Отображение карт выбранной редкости
        rarity = query.data.split("_")[1]
        cards = [card for card in user_data["cards"] if
card["rarity"] == rarity]

        # Определяем клавиатуру до использования
        keyboard = []
        if not cards:
            await query.edit_message_text(f"У вас нет карт редкости
«{rarity}».")
        else:
            keyboard = [[InlineKeyboardButton(card["name"],
callback_data=f"card_{card['name']}")] for card in cards]
            reply_markup = InlineKeyboardMarkup(keyboard)
            try:
                await query.edit_message_text(f"Вот ваши карты
редкости «{rarity}»:", reply_markup=reply_markup)
            except Exception as e:
                print(f"Ошибка редактирования сообщения: {e}")
                await query.answer("Произошла ошибка при попытке
отобразить карты.")
    elif query.data.startswith("card_"):
        # Подробности по карте
        card_name = query.data.split("_")[1]
        card = next((card for card in user_data["cards"] if
card["name"] == card_name), None)
        if card:
            # Теперь проверяем, если картинка есть в данных
            if "image" in card and card["image"]:  # Проверяем,
есть ли ключ и непустое значение
                print(f"Пытаемся открыть файл: {card['image']}")
                try:
                    with open(card["image"], "rb") as img:
                        await query.message.reply_photo(
                            photo=img,
                            caption=(
                                f"🃏Название • {card['name']}\n\n"
                                f"💠Редкость •
{card['rarity']}\n\n"
                                f"♦Рубины • {card['rubies']}\n\n"

f"[{user_id}](tg://user?id={user_id}\n\n)"
                            ),
                            parse_mode="Markdown"
                        )
```

```
                except FileNotFoundError:
                    await query.message.reply_text(f"Изображение
карты {card['name']} не найдено.")
            else:
                # Если изображения нет, отправляем только текстовую
информацию
                await query.message.reply_text(
                    text=(
                        f"🃏Название: {card['name']}\n\n"
                        f"💠Редкость: {card['rarity']}\n\n"
                        f"♦Рубины: {card['rubies']}\n\n"
                        f"[{user_id}](tg://user?id={user_id}\n\n)"
                        "Изображение отсутствует"
                    ),
                    parse_mode="Markdown"
                )


def main():
    """Запуск бота."""
    application = ApplicationBuilder().token("token").build()

    application.add_handler(CommandHandler("start", start))
    application.add_handler(CommandHandler("profile", profile))
    application.add_handler(CommandHandler("banana", banana))
    application.add_handler(CallbackQueryHandler(handle_callback))
    application.add_handler(CommandHandler("leaderboard",
leaderboard))
    application.add_handler(CommandHandler("give_card", give_card))

    application.run_polling()


if __name__ == "__main__":
    main()
```