Полное описание каждого Endpoint

AUTH.PY - **Аутентификация и авторизация**

1. `POST /api/v1/auth/register` - Регистрация нового пользователя

За что отвечает:

- Создаёт нового пользователя в системе
- Хеширует пароль для безопасности
- Возвращает JWT токен для последующих запросов

Параметры:

```
{
  "email": "user@example.com",      // email пользователя (уникальный)
  "password": "password123",         // пароль (будет захеширован)
  "full_name": "Ivan Petrov"         // полное имя пользователя
}
```

Пример запроса:

```
curl -X POST "http://178.72.152.189/api/v1/auth/register" \
  -H "Content-Type: application/json" \
  -d '{
    "email": "ivan@alfabank.com",
    "password": "SecurePass123!",
    "full_name": "Ivan Petrov"
  }'
```

Ответ (успех):

```
{
  "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJpdmFuQGFsZmFiYW5rLmNvb
}
```

Возможные ошибки:

- 400: `"Email already registered"` - этот email уже зарегистрирован
- 500: `"Internal server error"` - ошибка на сервере

2. `POST /api/v1/auth/login` - Логин пользователя

За что отвечает:

- Проверяет корректность email и пароля
- Если верные - возвращает JWT токен
- Токен используется для доступа к защищённым endpoints

Параметры:

```
{
  "email": "user@example.com",      // email зарегистрированного пользователя
  "password": "password123"          // пароль
}
```

Пример запроса:

```
curl -X POST "http://178.72.152.189/api/v1/auth/login" \
  -H "Content-Type: application/json" \
  -d '{
    "email": "ivan@alfabank.com",
    "password": "SecurePass123!"
  }'
```

Ответ (успех):

```
{
  "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
}
```

Возможные ошибки:

- 401: `"Invalid credentials"` - неверный email или пароль
- 401: `"Invalid credentials"` - пользователь не найден

DASHBOARD.PY - Общая статистика системы

3. `GET /api/v1/dashboard` - Получить общий dashboard

За что отвечает:

- Показывает общую статистику системы
- Количество прогнозов и клиентов
- Качество модели (метрики WMAE, MAE)
- Распределение доходов по категориям
- Требует аутентификации (нужен JWT токен)

Параметры: Нет

Пример запроса:

```
TOKEN="ваш_jwt_токен_здесь"

curl -X GET "http://178.72.152.189/api/v1/dashboard" \
  -H "Authorization: Bearer $TOKEN"
```

Ответ:

```
{
  "stats": {
    "total_predictions": 1250,           // всего прогнозов в системе
    "total_clients": 856,                // всего клиентов
    "avg_confidence": 0.87,              // средняя уверенность модели (87%)
    "model_version": "1.0.0",            // версия модели
    "last_update": "2025-11-28T22:00:00Z", // когда обновлена модель
    "metrics": [
      {
        "metric_name": "WMAE",           // Weighted Mean Absolute Error
        "train_value": 0.1234,           // значение на тренировочных данных
        "test_value": 0.1456,            // значение на тестовых данных
        "unit": ""
      },
      {
        "metric_name": "MAE",            // Mean Absolute Error
        "train_value": 5234.50,          // средняя ошибка (₽)
        "test_value": 6123.75,
        "unit": "₽"
      }
    ]
  },
  "income_distribution": [
    {
      "category": "LOW",                 // низкие доходы
      "count": 320,                      // количество клиентов в этой категории
      "percentage": 37.4                 // процент от всех
    },
    {
      "category": "MIDDLE",              // средние доходы
      "count": 380,
      "percentage": 44.4
    },
    {
      "category": "HIGH",                // высокие доходы
      "count": 156,
      "percentage": 18.2
    }
  ],
  "recent_predictions": [],              // последние прогнозы (сейчас пусто)
  "top_features": []                     // важные признаки (сейчас пусто)
}
```

Что показывает:

- Сколько клиентов в системе и их распределение по доходам
- Насколько хорошо работает модель машинного обучения
- Метрики качества прогнозирования

✓ HEALTH.PY - Проверка здоровья системы

4. `GET /api/v1/health` - Базовая проверка здоровья

За что отвечает:

- Проверяет, работает ли API
- Не требует аутентификации
- Используется для мониторинга (например, балансировщиками нагрузки)

Параметры: Нет

Пример запроса:

```
curl -X GET "http://178.72.152.189/api/v1/health"
```

Ответ:

```
{
  "status": "healthy",                          // статус системы
  "timestamp": "2025-11-29T12:38:00.123456",  // время ответа
  "service": "alfa-bank-income-prediction"    // название сервиса
}
```

Зачем нужен:

- Проверить, запущен ли бекенд
- Убедиться, что API доступен
- Используется Docker для health checks

5. `GET /api/v1/health/detailed` - Подробная проверка здоровья

За что отвечает:

- Проверяет здоровье всех компонентов системы
- Проверяет связь с БД (PostgreSQL)
- Проверяет связь с кэшем (Redis)
- Не требует аутентификации

Параметры: Нет

Пример запроса:

```
curl -X GET "http://178.72.152.189/api/v1/health/detailed"
```

Ответ:

```
{
  "status": "ok",                              // общий статус
  "timestamp": "2025-11-29T12:38:00.123456",  // время проверки
  "database": "ok",                            // БД работает
  "redis": "ok"                                // кэш работает
}
```

Что показывает:

- ✅ БД доступна и работает
- ✅ Redis (кэш) работает
- ✅ Все сервисы в порядке

CLIENTS.PY - Управление клиентами

6. `POST /api/v1/clients/seed/data` - Создать тестовых клиентов

За что отвечает:

- Заполняет БД 5 тестовыми клиентами
- Создаёт для каждого клиента прогноз дохода
- Используется для тестирования и демонстрации
- Не требует аутентификации

Параметры: Нет

Пример запроса:

```
curl -X POST "http://178.72.152.189/api/v1/clients/seed/data"
```

Ответ (при первом запуске):

```
{
  "message": "✓ Test data created",
  "created_clients": 5,
  "clients": [
    "cli_test_001",  // Иван, 35 лет, Москва, доход 150k
    "cli_test_002",  // Мария, 28 лет, СПб, доход 85k
```

```
    "cli_test_003",  // Петр, 42 года, Москва, доход 250k
    "cli_test_004",  // Анна, 31 год, Екатеринбург, доход 95k
    "cli_test_005"   // Сергей, 55 лет, Москва, доход 350k
  ]
}
```

Ответ (если данные уже существуют):

```
{
  "message": "Data already exists",
  "count": 5  // количество уже созданных клиентов
}
```

Важно:

- Запустить один раз для заполнения БД тестовыми данными
- При повторном запуске ничего не создаст

7. `GET /api/v1/clients` - Получить список всех клиентов

За что отвечает:

- Выводит список всех клиентов из БД
- Показывает их данные и последний прогноз дохода
- Поддерживает сортировку и пагинацию
- Не требует аутентификации

Параметры (query parameters):

```
sort=income_real    // сортировать по (income_real или age)
order=desc          // порядок (desc или asc)
limit=50            // сколько записей выводить
offset=0            // с какой позиции начинать
```

Примеры запросов:

Базовый запрос - 50 клиентов, отсортированы по доходу (от большего):

```
curl -X GET "http://178.72.152.189/api/v1/clients"
```

Сортировка по возрасту (от меньшего):

```
curl -X GET "http://178.72.152.189/api/v1/clients?sort=age&order=asc"
```

Первые 10 клиентов:

```
curl -X GET "http://178.72.152.189/api/v1/clients?limit=10&offset=0"
```

Клиенты 20-30:

```
curl -X GET "http://178.72.152.189/api/v1/clients?limit=10&offset=20"
```

Ответ:

```
{
  "total": 5,  // всего клиентов в системе
  "items": [
    {
      "client_id": "cli_test_005",           // уникальный ID
      "age": 55,                             // возраст
      "gender": "M",                         // пол (M/F)
      "city": "Moscow",                      // город
      "region": "Moscow",                    // регион
      "income_real": 350000,                 // реальный доход (₽)
      "income_predicted": 332500,            // прогноз модели (₽)
      "confidence": 0.82,                    // уверенность (82%)
      "income_category": "MIDDLE"            // категория (LOW/MIDDLE/HIGH)
    },
    {
      "client_id": "cli_test_003",
      "age": 42,
      "gender": "M",
      "city": "Moscow",
      "region": "Moscow",
      "income_real": 250000,
      "income_predicted": 237500,
      "confidence": 0.82,
      "income_category": "MIDDLE"
    }
    // ... остальные клиенты
  ]
}
```

Что показывает:

- Полный список всех клиентов
- Для каждого - его данные и прогноз дохода
- Можно фильтровать, сортировать, страничить

8. `GET /api/v1/clients/{client_id}` - Получить одного клиента

За что отвечает:

- Выводит информацию об одном конкретном клиенте
- Показывает его реальный доход и прогноз
- Не требует аутентификации

Параметры:

```
client_id - ID клиента (например, cli_test_001)
```

Пример запроса:

```
curl -X GET "http://178.72.152.189/api/v1/clients/cli_test_001"
```

Ответ:

```
{
  "client_id": "cli_test_001",          // ID клиента
  "age": 35,                            // возраст
  "gender": "M",                        // пол
  "city": "Moscow",                     // город проживания
  "region": "Moscow",                   // регион
  "income_real": 150000,                // реальный доход
  "income_predicted": 142500,           // прогноз модели
  "confidence": 0.82,                   // уверенность прогноза
  "income_category": "MIDDLE"           // категория дохода
}
```

Возможные ошибки:

- 404: `"Not found"` - клиента с таким ID нет в БД

PREDICTIONS.PY - Прогнозирование доходов

9. `POST /api/v1/predictions/predict` - Получить прогноз дохода

За что отвечает:

- Делает прогноз дохода для конкретного клиента
- Использует модель машинного обучения
- Возвращает предсказанный доход, ошибку и уверенность
- Требует аутентификации (JWT токен)

Параметры:

```
{
  "client_id": "cli_test_001"  // ID клиента для прогноза
}
```

Пример запроса:

```
# Сначала получить токен
TOKEN=$(curl -s -X POST "http://178.72.152.189/api/v1/auth/login" \
  -H "Content-Type: application/json" \
  -d '{"email":"ivan@alfabank.com","password":"SecurePass123!"}' | jq -r '.access_token')

# Затем запросить прогноз
curl -X POST "http://178.72.152.189/api/v1/predictions/predict" \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer $TOKEN" \
  -d '{"client_id": "cli_test_001"}'
```

Ответ:

```
{
  "prediction_id": "a1b2c3d4-e5f6-7890-abcd-ef1234567890",  // уникальный ID прогноза
  "client_id": "cli_test_001",                              // для какого клиента
  "predicted_income": 142500,                               // прогноз дохода (₽)
  "actual_income": 150000,                                  // реальный доход (₽)
  "confidence": 0.82,                                       // уверенность (82%)
  "income_category": "MIDDLE",                              // категория (LOW/MIDDLE/HIGH
  "error": 7500,                                            // ошибка в ₽ (|142500-150000
  "error_percent": 5.0,                                     // ошибка в % (7500/150000*10
  "recommendations": [],                                    // рекомендации (сейчас пусто
  "explanation": [],                                        // объяснение (сейчас пусто)
  "timestamp": "2025-11-28T22:00:00Z",                      // когда был сделан прогноз
  "model_version": "1.0.0"                                  // версия модели
}
```

Что показывает:

- **predicted_income** vs **actual_income** - как модель ошиблась
- **confidence** - насколько модель уверена в своём прогнозе
- **error** и **error_percent** - абсолютная и относительная ошибка
- **income_category** - категория дохода (для какого сегмента клиентов)

Возможные ошибки:

- 401: `"Not authenticated"` - не передан токен
- 404: `"Client not found"` - клиента с таким ID нет

ПУСТЫЕ ENDPOINTS (не реализованы)

✖ MONITORING.PY

- Файл пуст, нет endpoints
- Можно добавить мониторинг производительности

✖ RECOMMENDATIONS.PY

- Файл пуст, нет endpoints
- Можно добавить рекомендации финансовых продуктов

✖ EXPLANATIONS.PY

- Файл пуст, нет endpoints
- Можно добавить объяснения прогнозов (SHAP, LIME)

Таблица всех endpoints

Метод	Path	Аутентификация	Описание
POST	`/api/v1/auth/register`	✖	Регистрация пользователя
POST	`/api/v1/auth/login`	✖	Логин пользователя
GET	`/api/v1/health`	✖	Базовая проверка здоровья
GET	`/api/v1/health/detailed`	✖	Подробная проверка здоровья
POST	`/api/v1/clients/seed/data`	✖	Создать 5 тестовых клиентов
GET	`/api/v1/clients`	✖	Список всех клиентов
GET	`/api/v1/clients/{id}`	✖	Один клиент по ID
POST	`/api/v1/predictions/predict`	✔	Прогноз дохода клиента
GET	`/api/v1/dashboard`	✔	Общая статистика

Типичный сценарий использования

```
# 1. Проверить здоровье API
curl -X GET "http://178.72.152.189/api/v1/health"

# 2. Заполнить БД тестовыми клиентами
curl -X POST "http://178.72.152.189/api/v1/clients/seed/data"

# 3. Посмотреть всех клиентов
curl -X GET "http://178.72.152.189/api/v1/clients"

# 4. Зарегистрироваться
```

```
TOKEN=$(curl -s -X POST "http://178.72.152.189/api/v1/auth/register" \
  -H "Content-Type: application/json" \
  -d '{"email":"test@example.com","password":"123","full_name":"Test"}' | jq -r '.access_

# 5. Получить прогноз для клиента
curl -X POST "http://178.72.152.189/api/v1/predictions/predict" \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer $TOKEN" \
  -d '{"client_id":"cli_test_001"}'

# 6. Посмотреть общий dashboard
curl -X GET "http://178.72.152.189/api/v1/dashboard" \
  -H "Authorization: Bearer $TOKEN"
```