cat frontend/src/api/client.js

```
cat frontend/src/pages/ClientListPage.jsx
cat frontend/src/pages/DashboardPage.jsx
cat frontend/src/pages/ClientSearchPage.jsx
cat frontend/src/pages/LoginPage.jsx
cat frontend/src/App.jsx
import React, { useState, useEffect } from 'react';
import axios from 'axios';

export default function ClientListPage() {
const [clients, setClients] = useState([]);
const [loading, setLoading] = useState(true);
const [sortBy, setSortBy] = useState('income_predicted');
const [order, setOrder] = useState('desc');

useEffect(() => {
fetchClients();
}, [sortBy, order]);

const fetchClients = async () => {
setLoading(true);
try {
const token = localStorage.getItem('token');
const response = await axios.get(
${import.meta.env.VITE_API_URL}/clients?sort=${sortBy}&order=${order}&limit=50,
{
headers: { Authorization: Bearer ${token} }
}
);
setClients(response.data.items);
} catch (error) {
alert('Ошибка загрузки клиентов');
}
setLoading(false);
};
```

```
if (loading) return <div>Загрузка...</div>;
```

```
return (
<div className="clients-list-page">
```

Список клиентов

```
  <div className="sort-controls">
    <label>
      Сортировать по:
      <select value={sortBy} onChange={(e) => setSortBy(e.target.value)}>
        <option value="income_predicted">Прогнозу дохода</option>
        <option value="income_real">Реальному доходу</option>
```

~~~
        <option value="age">Возрасту</option>
      </select>
    </label>

    <label>
      Порядок:
      <select value={order} onChange={(e) => setOrder(e.target.value)}>
        <option value="desc">↓ Убывающий</option>
        <option value="asc">↑ Возрастающий</option>
      </select>
    </label>
  </div>

  <table className="clients-table">
    <thead>
      <tr>
        <th>ID клиента</th>
        <th>Возраст</th>
        <th>Город</th>
        <th>Реальный доход</th>
        <th>Прогноз дохода</th>
        <th>Уверенность</th>
        <th>Категория</th>
      </tr>
    </thead>
    <tbody>
      {clients.map((client) => (
        <tr key={client.client_id} className="client-row">
          ```
          <td className="client-id">{client.client_id}</td>
          ```
          <td>{client.age}</td>
          <td>{client.city}</td>
          <td className="income">
            {client.income_real ? client.income_real.toLocaleString() : '-'} ₽
          </td>
          <td className="income highlight">
            {client.income_predicted ? client.income_predicted.toLocaleString() : '-'} ₽
          </td>
          <td className="confidence">
            {client.confidence ? (client.confidence * 100).toFixed(1) : '-'}%
          </td>
          <td className="category">
            <span className={`badge badge-${client.income_category?.toLowerCase()}`}>
              {client.income_category || '-'}
            </span>
          </td>
        </tr>
      ))}
    </tbody>
  </table>

  {clients.length === 0 && (
    <div className="empty-state">
      <p>Нет данных о клиентах</p>
      <p>Сначала создайте тестовые данные через POST /api/v1/clients/seed/data</p>
    </div>
  )}
</div>
~~~

);
}
import React, { useEffect, useState } from 'react';
import axios from 'axios';

```
export default function DashboardPage() {
const [dashboard, setDashboard] = useState(null);
const [loading, setLoading] = useState(true);

useEffect(() => {
fetchDashboard();
}, []);

const fetchDashboard = async () => {
try {
const token = localStorage.getItem('token');
const response = await axios.get(
${import.meta.env.VITE_API_URL}/dashboard,
{
headers: { Authorization: Bearer ${token} }
}
);
setDashboard(response.data);
} catch (error) {
alert('Ошибка загрузки dashboard');
}
setLoading(false);
};
```

```
if (loading) return <div>Загрузка...</div>;
```

```
if (!dashboard) return <div>Ошибка загрузки данных</div>;
```

```
return (
<div className="dashboard">
<h1>Система прогноза доходов Альфа-Банка</h1>
<button onClick={() => {
localStorage.removeItem('token');
window.location.href = '/';
}}>Выход</button>
```

~~~
  <section className="stats-grid">
    <div className="stat-card">
      ```
      <div className="stat-label">Всего прогнозов</div>
      ```
      ```
      <div className="stat-value">{dashboard.stats.total_predictions}</div>
      ```
    </div>

    <div className="stat-card">
      ```
      <div className="stat-label">Всего клиентов</div>
      ```
      ```
      <div className="stat-value">{dashboard.stats.total_clients}</div>
      ```
    </div>

    <div className="stat-card">
      ```
      <div className="stat-label">Средняя уверенность</div>
~~~

~~~
        ```
        ```
        <div className="stat-value">{(dashboard.stats.avg_confidence * 100).toFixed(1)}%</div>
        ```
      </div>

      <div className="stat-card">
        ```
        <div className="stat-label">Версия модели</div>
        ```
        ```
        <div className="stat-value">{dashboard.stats.model_version}</div>
        ```
      </div>
    </section>

    <section className="metrics-section">
      <h2>Метрики качества</h2>
      <table className="metrics-table">
        <thead>
          <tr>
            <th>Метрика</th>
            <th>Обучение</th>
            <th>Тест</th>
          </tr>
        </thead>
        <tbody>
          {dashboard.stats.metrics.map((m) => (
            <tr key={m.metric_name}>
              <td>{m.metric_name}</td>
              <td>{m.train_value.toFixed(4)}</td>
              <td>{m.test_value.toFixed(4)}</td>
            </tr>
          ))}
        </tbody>
      </table>
    </section>

    <section className="distribution-section">
      <h2>Распределение по категориям дохода</h2>
      <div className="distribution-grid">
        {dashboard.income_distribution.map((d) => (
          <div key={d.category} className="distribution-item">
            ```
            <div className="category-label">{d.category}</div>
            ```
            ```
            <div className="category-count">{d.count}</div>
            ```
            ```
            <div className="category-percent">{d.percentage.toFixed(1)}%</div>
            ```
          </div>
        ))}
      </div>
    </section>
  </div>
~~~

);
}
import React, { useState, useEffect } from 'react';
import axios from 'axios';

export default function ClientSearchPage() {
const [clientId, setClientId] = useState('');

const [client, setClient] = useState(null);
const [loading, setLoading] = useState(false);
const [error, setError] = useState('');

const handleSearch = async (e) => {
e.preventDefault();
if (!clientId.trim()) return;

```
setLoading(true);
setError('');

try {
  const token = localStorage.getItem('token');
  const response = await axios.get(
    `${import.meta.env.VITE_API_URL}/clients/${clientId}`,
    {
      headers: { Authorization: `Bearer ${token}` }
    }
  );
  setClient(response.data);
} catch (err) {
  setError(err.response?.data?.detail || 'Клиент не найден');
  setClient(null);
}

setLoading(false);
```

};

return (
<div className="search-page">

Поиск клиента по ID

~~~
  <form onSubmit={handleSearch} className="search-form">
    <input
      type="text"
      placeholder="Введите ID клиента (например: cli_test_001)"
      value={clientId}
      onChange={(e) => setClientId(e.target.value)}
      required
    />
    <button type="submit" disabled={loading}>
      {loading ? 'Поиск...' : 'Поиск'}
    </button>
  </form>

  ```
  {error && <div className="error-message">{error}</div>}
  ```

  {client && (
    <div className="client-card-detailed">
      <h3>Информация о клиенте</h3>

      <div className="client-details">
        <div className="detail-row">
          ```
          <span className="detail-label">ID:</span>
          ```
          ```
          <span className="detail-value">{client.client_id}</span>
          ```
~~~

~~~
          </div>

          <div className="detail-row">
            ```
            <span className="detail-label">Возраст:</span>
            ```
            ```
            <span className="detail-value">{client.age} лет</span>
            ```
          </div>

          <div className="detail-row">
            ```
            <span className="detail-label">Пол:</span>
            ```
            ```
            <span className="detail-value">{client.gender === 'M' ? 'Мужской' : 'Женский'}</span>
            ```
          </div>

          <div className="detail-row">
            ```
            <span className="detail-label">Город:</span>
            ```
            ```
            <span className="detail-value">{client.city}</span>
            ```
          </div>

          <div className="detail-row">
            ```
            <span className="detail-label">Регион:</span>
            ```
            ```
            <span className="detail-value">{client.region}</span>
            ```
          </div>

          {client.income_real && (
            <div className="detail-row highlight">
              ```
              <span className="detail-label">Реальный доход:</span>
              ```
              ```
              <span className="detail-value">{client.income_real.toLocaleString()} ₽</span>
              ```
            </div>
          )}

          {client.income_predicted && (
            <div className="detail-row highlight">
              ```
              <span className="detail-label">Прогноз дохода:</span>
              ```
              ```
              <span className="detail-value">{client.income_predicted.toLocaleString()} ₽</span>
              ```
            </div>
          )}

          {client.confidence && (
            <div className="detail-row">
              ```
              <span className="detail-label">Уверенность:</span>
              ```
              ```
              <span className="detail-value">{(client.confidence * 100).toFixed(1)}%</span>
~~~

~~~
      ```
     </div>
    )}

    {client.income_category && (
     <div className="detail-row">
      ```
      <span className="detail-label">Категория:</span>
      ```
      ```
      <span className="detail-value">{client.income_category}</span>
      ```
     </div>
    )}
   </div>
  </div>
 )}
</div>
~~~

);
}
import React, { useState } from 'react';
import axios from 'axios';

export default function LoginPage() {
const [email, setEmail] = useState('test@alfabank.ru');
const [password, setPassword] = useState('test123');
const [fullName, setFullName] = useState('Test User');
const [isRegister, setIsRegister] = useState(false);
const [loading, setLoading] = useState(false);
const [error, setError] = useState('');

const handleAuth = async (e) ⇒ {
e.preventDefault();
setLoading(true);
setError('');

```
try {
  const endpoint = isRegister ? '/auth/register' : '/auth/login';
  const data = isRegister
    ? { email, password, full_name: fullName }
    : { email, password };

  const response = await axios.post(
    `${import.meta.env.VITE_API_URL}${endpoint}`,
    data
  );

  const token = response.data.access_token;
  localStorage.setItem('token', token);

  // Перезагружаем страницу чтобы приложение это заметило
  window.location.href = '/';
} catch (err) {
  setError(err.response?.data?.detail || 'Ошибка при входе');
}

setLoading(false);
```

};

return (
<div className="login-page">
<div className="login-card">
`<h1>Альфа-Банк</h1>`
Система прогноза доходов

~~~
    <form onSubmit={handleAuth} className="login-form">
      <input
        type="email"
        placeholder="Email"
        value={email}
        onChange={(e) => setEmail(e.target.value)}
        required
      />

      {isRegister && (
        <input
          type="text"
          placeholder="ФИО"
          value={fullName}
          onChange={(e) => setFullName(e.target.value)}
          required
        />
      )}

      <input
        type="password"
        placeholder="Пароль"
        value={password}
        onChange={(e) => setPassword(e.target.value)}
        required
      />

      <button type="submit" disabled={loading}>
        {loading ? 'Загрузка...' : isRegister ? 'Регистрация' : 'Вход'}
      </button>
    </form>

    ```
    {error && <div className="error-message">{error}</div>}
    ```

    <div className="auth-toggle">
      {isRegister ? (
        <>
          Уже есть аккаунт?{' '}
          <button onClick={() => setIsRegister(false)} className="link-button">
            Войти
          </button>
        </>
      ) : (
        <>
          Нет аккаунта?{' '}
          <button onClick={() => setIsRegister(true)} className="link-button">
            Регистрация
          </button>
        </>
      )}
    </div>

    <div className="demo-credentials">
      Demo: [test@alfabank.ru](mailto:test@alfabank.ru) / test123
    </div>
  </div>
</div>
~~~

```
);
}
import React, { useEffect, useState } from 'react';
import LoginPage from './pages/LoginPage';
import DashboardPage from './pages/DashboardPage';
import ClientSearchPage from './pages/ClientSearchPage';
import ClientListPage from './pages/ClientListPage';
import './styles/index.css';

export default function App() {
const [isLoggedIn, setIsLoggedIn] = useState(false);
const [loading, setLoading] = useState(true);
const [currentPage, setCurrentPage] = useState('dashboard');

useEffect(() => {
const token = localStorage.getItem('token');
setIsLoggedIn(!!token);
setLoading(false);
}, []);
```

```
if (loading) return <div className="loading">Загрузка...</div>;
```

```
if (!isLoggedIn) return ;

return (
<div className="app">
<div className="nav-brand">Альфа-Банк</div>

<button
className={nav-link ${currentPage === 'dashboard' ? 'active' : ''}}
onClick={() => setCurrentPage('dashboard')}
>
□ Dashboard
</button>
<button
className={nav-link ${currentPage === 'search' ? 'active' : ''}}
onClick={() => setCurrentPage('search')}
>
□ Поиск клиента
</button>
<button
className={nav-link ${currentPage === 'list' ? 'active' : ''}}
onClick={() => setCurrentPage('list')}
>
□ Список клиентов
</button>
<button
className="nav-link logout"
onClick={() => {
localStorage.removeItem('token');
setIsLoggedIn(false);
}}
>
□ Выход
</button>
```

```
  <main className="main-content">
    {currentPage === 'dashboard' && <DashboardPage />}
    {currentPage === 'search' && <ClientSearchPage />}
    {currentPage === 'list' && <ClientListPage />}
  </main>
</div>
```

);
}

Идеально! Теперь у меня есть всё. Создам полную архитектурную таблицу для каждого endpoint.

Полная архитектура: БД → Backend → Frontend

1️⃣ ENDPOINT: `GET /api/v1/clients` - Список клиентов

Слой	Файл	Код	Описание
1. БД инициализация	`backend/scripts/init.sql`	`sql CREATE TABLE clients ( client_id VARCHAR(50) PRIMARY KEY, age INTEGER, gender VARCHAR(20), city VARCHAR(255), region VARCHAR(255), income_real FLOAT, income_predicted FLOAT, confidence FLOAT, income_category VARCHAR(20), created_at TIMESTAMP );`	
2. ORM Модель	`backend/app/data/models.py`	`python class Client(Base): __tablename__ = "clients" client_id = Column(String(50), primary_key=True) age = Column(Integer) gender = Column(String(20)) city = Column(String(255)) region = Column(String(255)) income_real = Column(Float) income_predicted = Column(Float) confidence = Column(Float) income_category = Column(String(20))`	
3. Backend Endpoint	`backend/app/api/v1/endpoints/clients.py`	`python @router.get("/clients") async def list_clients(sort: str = "income_real", order: str = "desc", limit: int = 50, offset: int = 0): db = SessionLocal() query = db.query(Client) if order == "desc": query = query.order_by(desc(Client.income_real)) clients = query.offset(offset).limit(limit).all() items = [{"client_id": c.client_id, "age": c.age, "income_real": c.income_real, ...}] return {"total": total, "items": items}`	
4. API Клиент (Axios)	`frontend/src/api/client.js`	`` javascript // Примерный код fetchClients: async () => { const response = await axios.get(`${VITE_API_URL}/clients?sort=income_predicted&order=desc&limit=50`, {headers: {Authorization: `Bearer ${token}`}}); return response.data.items; } ``	
5. Frontend Component	`frontend/src/pages/ClientListPage.jsx`	`jsx const [clients, setClients] = useState([]); useEffect(() => { fetchClients(); }, [sortBy, order]); return ( <table> <thead><tr><th>ID</th><th>Age</th><th>Доход</th></tr></thead> <tbody>{clients.map(c => (<tr key={c.client_id}><td>{c.client_id}</td>...)}</tbody> </table> );`	

2⃣ ENDPOINT: `GET /api/v1/clients/{client_id}` - Один клиент

Слой	Код	Описание
БД	`SELECT * FROM clients WHERE client_id = 'cli_test_001'`	Выбирает одного клиента по ID
Backend	`python @router.get("/clients/{client_id}") async def get_client(client_id: str): client = db.query(Client).filter_by(client_id=client_id).first() return {client_id, age, city, income_real, ...}`	
Frontend	``jsx const response = await axios.get(`/api/v1/clients/${clientId}`); setClient(response.data); return (<div>{client.age}, {client.city}, {client.income_real}</div>);``	

3⃣ ENDPOINT: `POST /api/v1/clients/seed/data` - Создать тестовых клиентов

Слой	Код	Описание
БД	`sql INSERT INTO clients VALUES ('cli_test_001', 35, 'M', 'Moscow', ..., 150000); INSERT INTO clients VALUES ('cli_test_002', 28, 'F', 'SPB', ..., 85000);`	
Backend	`python @router.post("/clients/seed/data") async def seed_test_data(): if db.query(Client).count() > 0: return {"message": "Data already exists"} clients = [Client(...), Client(...), ...] db.add_all(clients) db.commit()`	
Frontend	`jsx // В зависимости от использования (обычно через curl или swagger)`	

4⃣ ENDPOINT: `GET /api/v1/dashboard` - Общая статистика

Слой	Код
БД	`sql SELECT COUNT(*) FROM predictions; SELECT COUNT(*) FROM clients; SELECT AVG(confidence) FROM predictions;`
Backend	`python @router.get("/dashboard") async def get_dashboard(current_user = Depends(get_current_user)): return { "stats": {"total_predictions": 1250, "total_clients": 856, "avg_confidence": 0.87}, "income_distribution": [...] }`
Frontend	`jsx const dashboard = await axios.get('/api/v1/dashboard'); return (<div>{dashboard.stats.total_predictions} прогнозов</div>);`

5⃣ ENDPOINT: `POST /api/v1/auth/register` - Регистрация

Слой	Файл	Код
БД	`init.sql`	`sql CREATE TABLE users (user_id SERIAL, email VARCHAR(255) UNIQUE, password_hash VARCHAR(500), ...);`
ORM	`models.py`	`python class User(Base): __tablename__ = "users" email = Column(String(255), unique=True) password_hash = Column(String(500))`
Backend	`endpoints/auth.py`	`python @router.post("/auth/register") async def register(user_data: UserRegister): hashed_pw = hash_password(user_data.password) user = User(email=user_data.email, password_hash=hashed_pw) db.add(user); db.commit() token = create_access_token({"sub": user_data.email}) return {"access_token": token}`
Frontend	`LoginPage.jsx`	`jsx const response = await axios.post('/api/v1/auth/register', {email, password, full_name}); const token = response.data.access_token; localStorage.setItem('token', token);`

6️⃣ ENDPOINT: `POST /api/v1/auth/login` - Логин

Слой	Код
БД	`SELECT * FROM users WHERE email = 'test@example.com'`
Backend	`python @router.post("/auth/login") async def login(credentials: UserLogin): user = db.query(User).filter_by(email=credentials.email).first() if not verify_password(credentials.password, user.password_hash): raise HTTPException(401, "Invalid") token = create_access_token({"sub": credentials.email}) return {"access_token": token}`
Frontend	`jsx const {access_token} = await axios.post('/api/v1/auth/login', {email, password}); localStorage.setItem('token', access_token); // Используется в Authorization header`

7️⃣ ENDPOINT: `POST /api/v1/predictions/predict` - Прогноз дохода

Слой	Код
БД (write)	`sql INSERT INTO predictions (prediction_id, client_id, predicted_income, confidence) VALUES ('uuid', 'cli_test_001', 142500, 0.82);`
Backend	`python @router.post("/predictions/predict") async def predict_income(request: PredictionRequest, current_user = Depends(get_current_user)): client = db.query(Client).filter_by(client_id=request.client_id).first() predicted = client.income_real * 0.95 prediction = Prediction(client_id=client_id, predicted_income=predicted, confidence=0.82) db.add(prediction); db.commit() return PredictionResponse(...)`
Frontend	Не используется на текущий момент (можно добавить отдельную страницу для прогнозов)

Поток данных: пример для ClientListPage

```
1. Пользователь кликает на вкладку "Список клиентов"
                ↓
2. App.jsx переводит на страницу list
                ↓
3. ClientListPage.jsx загружается и выполняет useEffect
                ↓
4. fetchClients() вызывает axios.get('/api/v1/clients')
                ↓
5. Запрос идёт на бекенд (nginx → backend:8000)
                ↓
6. endpoints/clients.py @router.get("/clients") обрабатывает
                ↓
7. Выполняется: db.query(Client).order_by(...).limit(50).all()
                ↓
8. SQLAlchemy преобразует в SQL:
   SELECT * FROM clients ORDER BY income_real DESC LIMIT 50
                ↓
9. PostgreSQL возвращает 50 строк из таблицы
                ↓
10. Backend преобразует в JSON: {total: 5, items: [...]}
                ↓
11. Frontend получает response.data.items
                ↓
12. setClients(response.data.items) обновляет state
                ↓
13. React перерисовывает таблицу с данными
```

Где менять каждый компонент

Если хочешь добавить новое поле в таблицу clients:

Шаг 1: БД (init.sql)

```
ALTER TABLE clients ADD COLUMN phone VARCHAR(20);
-- или при создании:
CREATE TABLE clients (..., phone VARCHAR(20), ...);
```

Шаг 2: ORM Model (models.py)

```
class Client(Base):
    __tablename__ = "clients"
    phone = Column(String(20))  # ← добавить
```

Шаг 3: Backend API (endpoints/clients.py)

```
@router.get("/clients")
async def list_clients(...):
    items = [{
        "client_id": c.client_id,
        "phone": c.phone,  # ← добавить в возвращаемый JSON
        ...
    }]
```

Шаг 4: Frontend Table (ClientListPage.jsx)

```
<table>
  <thead>
    <tr>
      <th>Телефон</th>  {/* ← добавить заголовок */}
    </tr>
  </thead>
  <tbody>
    {clients.map(c => (
      <tr>
        <td>{c.phone}</td>  {/* ← добавить ячейку */}
      </tr>
    ))}
  </tbody>
</table>
```

**Нужно ли мне создать конкретный пример добавления нового поля или endpoint? **