Отчет
По ДЗ
Дисциплина “Парадигмы и конструкции”

Студент: Прищенко А.А
Группа: ИБМ3-34Б

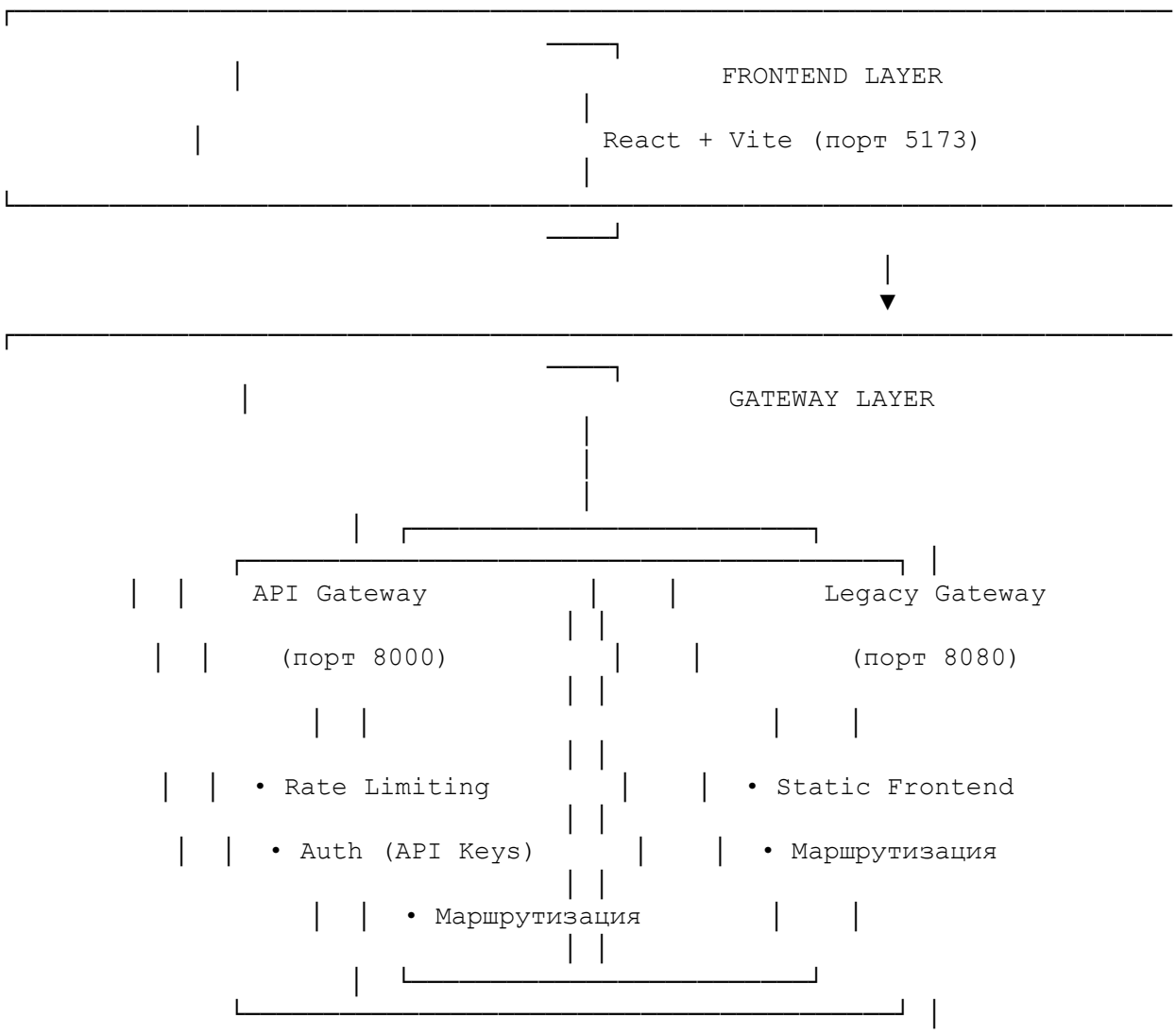
Архитектура системы Interview Prep Platform

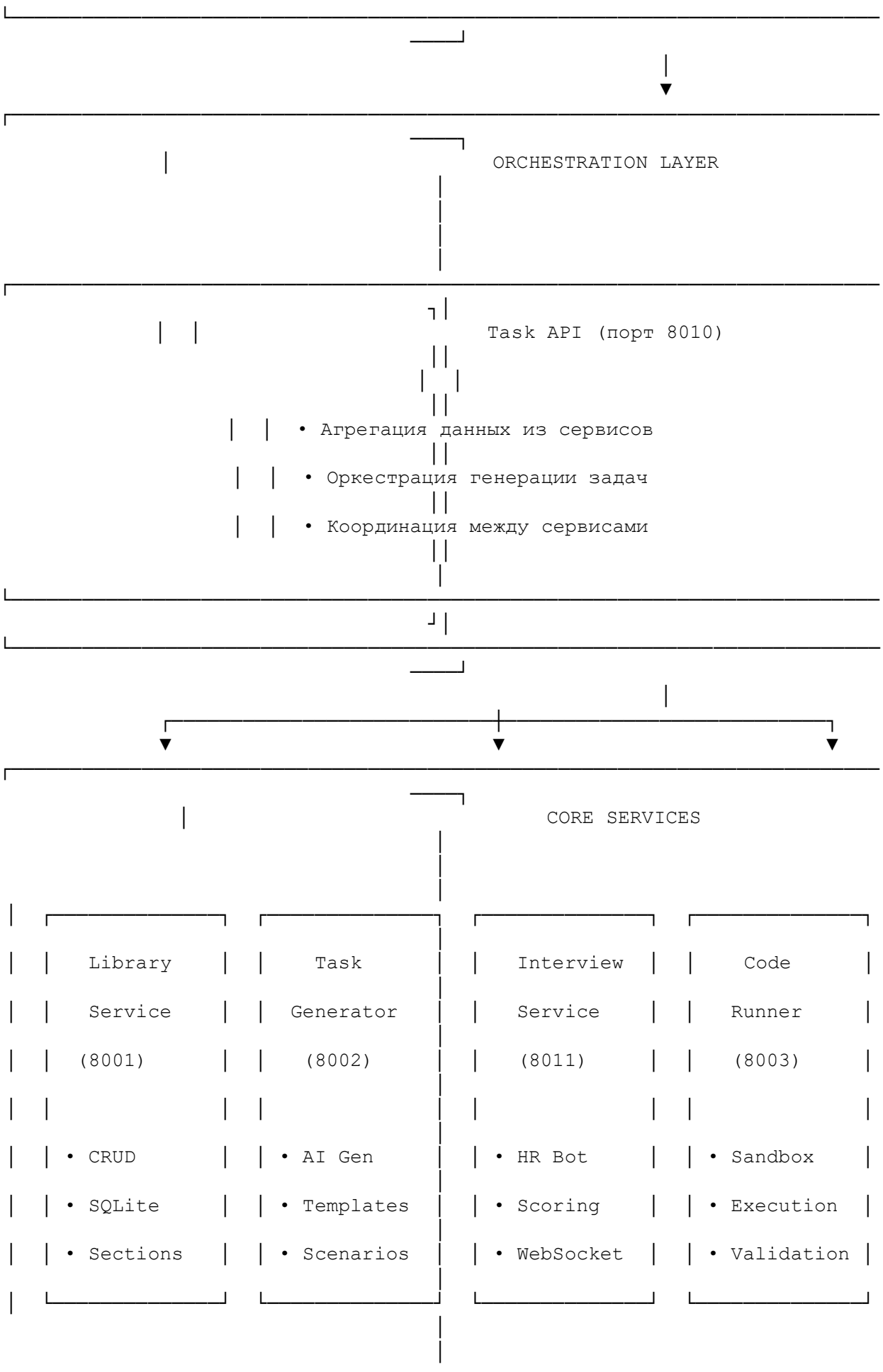
📋 Оглавление

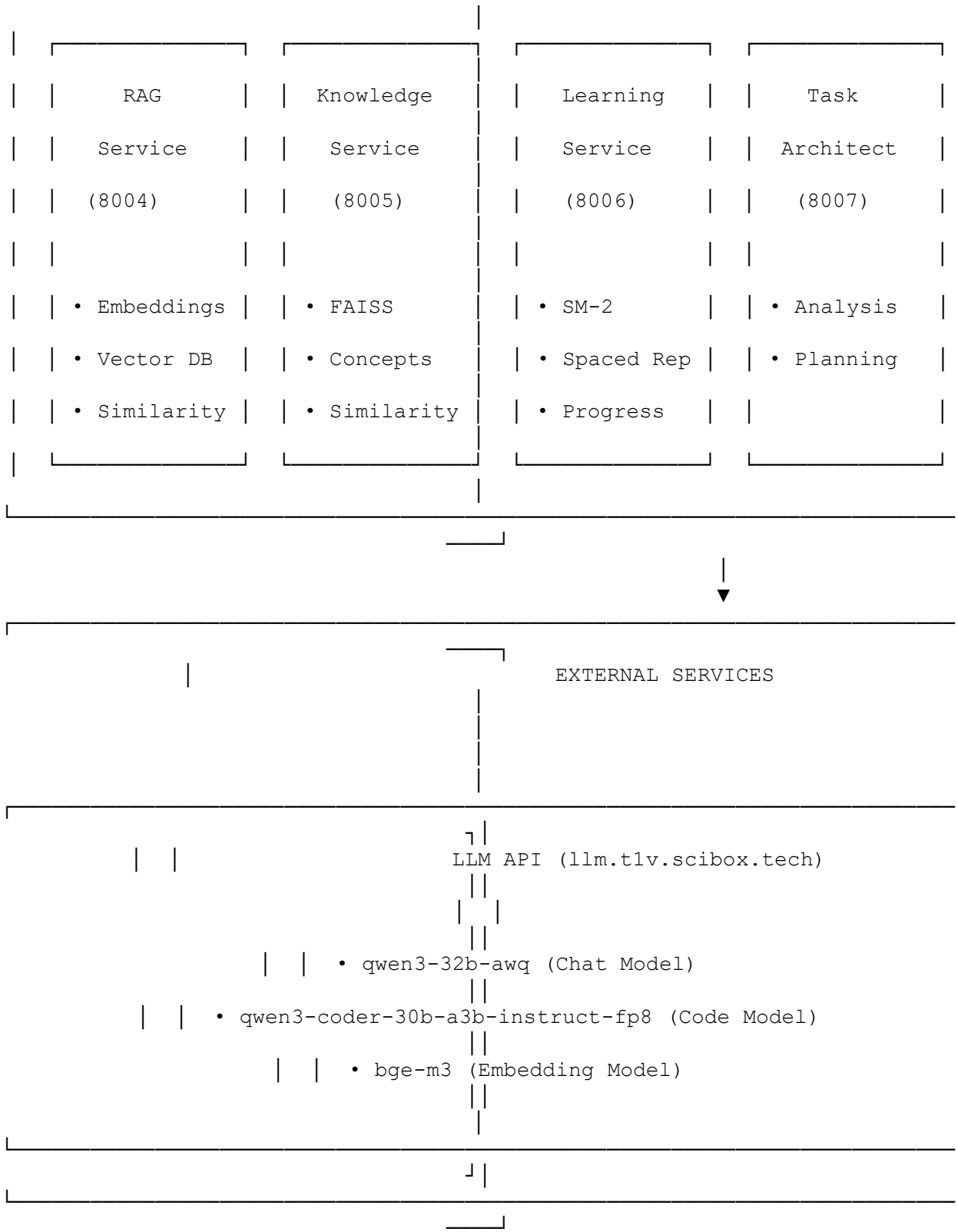
Обзор архитектуры

Система построена на **микросервисной архитектуре** с использованием:
- **FastAPI** для REST API сервисов
- **Docker Compose** для оркестрации контейнеров
- **LLM API** (Qwen3) для генерации контента и оценки
- **WebSocket** для real-time коммуникации

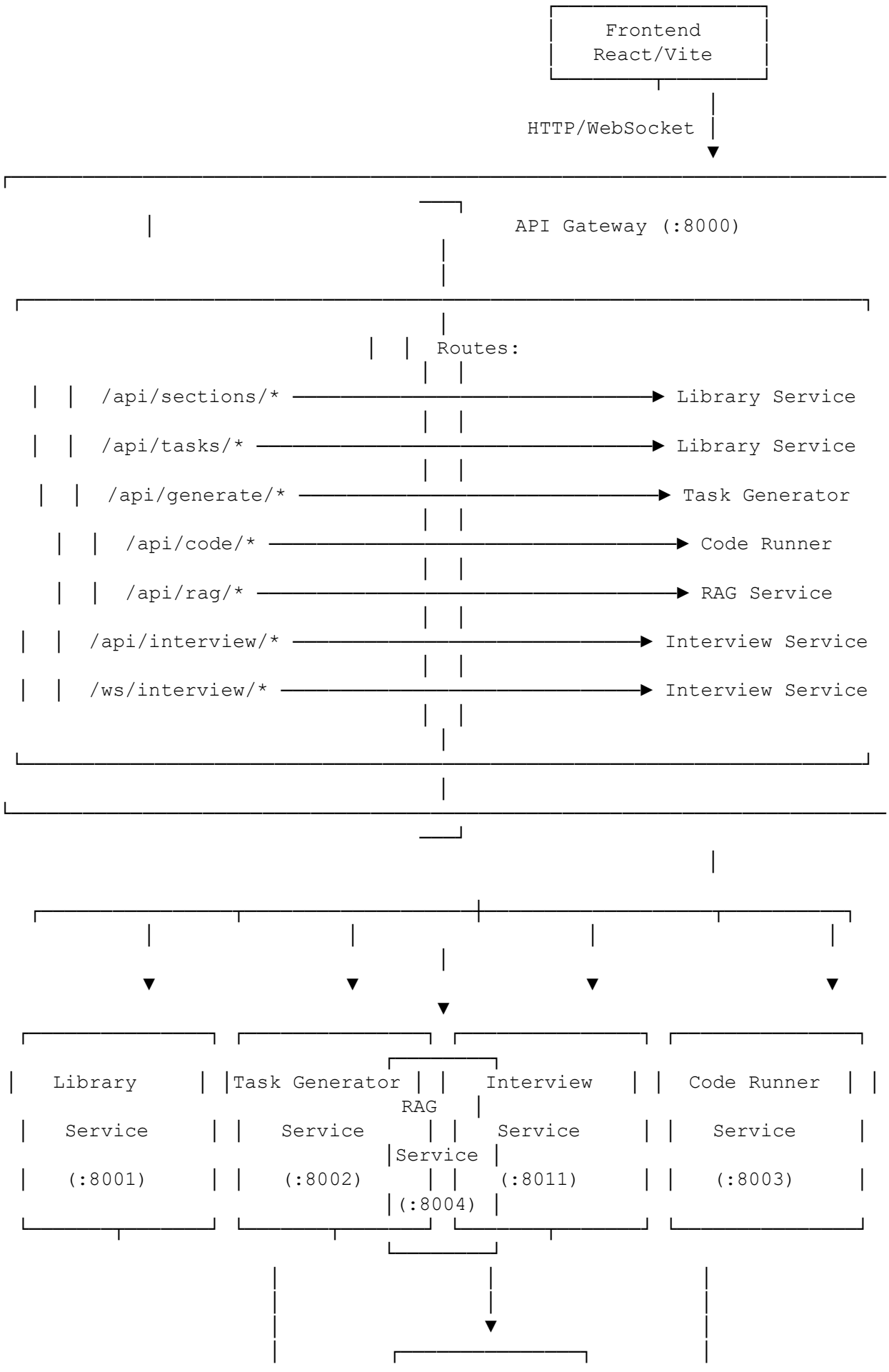
Основные слои:

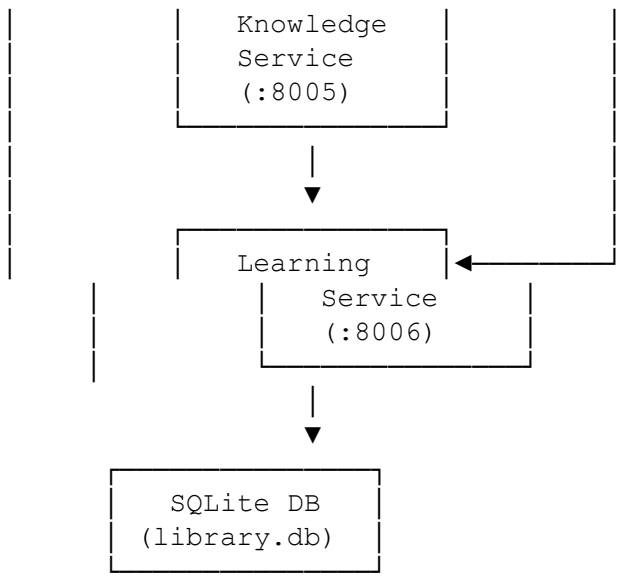



ORCHESTRATION LAYER
Task API (порт 8010)
• Агрегация данных из сервисов
• Оркестрация генерации задач
• Координация между сервисами
CORE SERVICES
Library Service (8001)
• CRUD
• SQLite
• Sections
Task Generator (8002)
• AI Gen
• Templates
• Scenarios
Interview Service (8011)
• HR Bot
• Scoring
• WebSocket
Code Runner (8003)
• Sandbox
• Execution
• Validation




```

---

## Диаграмма компонентов

### Детальная схема взаимодействия сервисов:

```


Frontend
React/Vite
HTTP/WebSocket
API Gateway (:8000)
Routes:
/api/sections/*
Library Service
/api/tasks/*
Library Service
/api/generate/*
Task Generator
/api/code/*
Code Runner
/api/rag/*
RAG Service
/api/interview/*
Interview Service
/ws/interview/*
Interview Service
Library Service (:8001)
Task Generator Service (:8002)
RAG Service (:8004)
Interview Service (:8011)
Code Runner Service (:8003)




```

---

## Описание сервисов

### 1. Interview Service (порт 8011)
**Главный сервис для проведения интервью**

| Компонент | Описание |
|---|---|
| `InterviewEngine` | Ядро логики интервью: генерация вопросов, обработка ответов, управление потоком |
| `AIJudge` | Оценка ответов кандидата с помощью LLM |
| `SchemaParser` | Парсинг схемы интервью (nodes/edges) в последовательность шагов |
| `InterviewSession` | Хранение состояния сессии: история, баллы, текущий шаг |

**Ключевые endpoints:**
- `POST /interview/start` - Начало интервью
- `POST /interview/message` - Отправка сообщения
- `POST /interview/code` - Отправка кода (Live Coding)
- `WS /ws/interview/{session_id}` - Real-time чат

### 2. Task Generator Service (порт 8002)
**Генерация задач с помощью AI**

| Компонент | Описание |
|---|---|
| `FlexibleGenerator` | Гибкая генерация без жёстких категорий |
| `HashtagGenerator` | Генерация на основе хэштегов и примеров |
| `ScenarioEngine` | Динамические сценарии (fix_code, complete, debug) |
| `Templates` | Шаблоны для разных типов задач |

**Типы сценариев:**
- `fix_code` - Исправление багов
- `complete` - Дополнение кода (TODO)
```

- `debug_output` - Отладка по выводу
- `refactor` - Рефакторинг
- `implement` - Реализация с нуля

3. Code Runner Service (порт 8003)
Безопасное выполнение кода

- Docker sandbox для изоляции
- Лимиты: 10 сек, 256 MB RAM
- Валидация против тест-кейсов
- Поддержка Python

4. Knowledge Service (порт 8005)
Управление знаниями и концепциями

- FAISS для векторного поиска
- Автоматическое обнаружение концепций
- Similarity threshold: 0.65
- BGE-M3 embeddings

5. Learning Service (порт 8006)
Адаптивное обучение

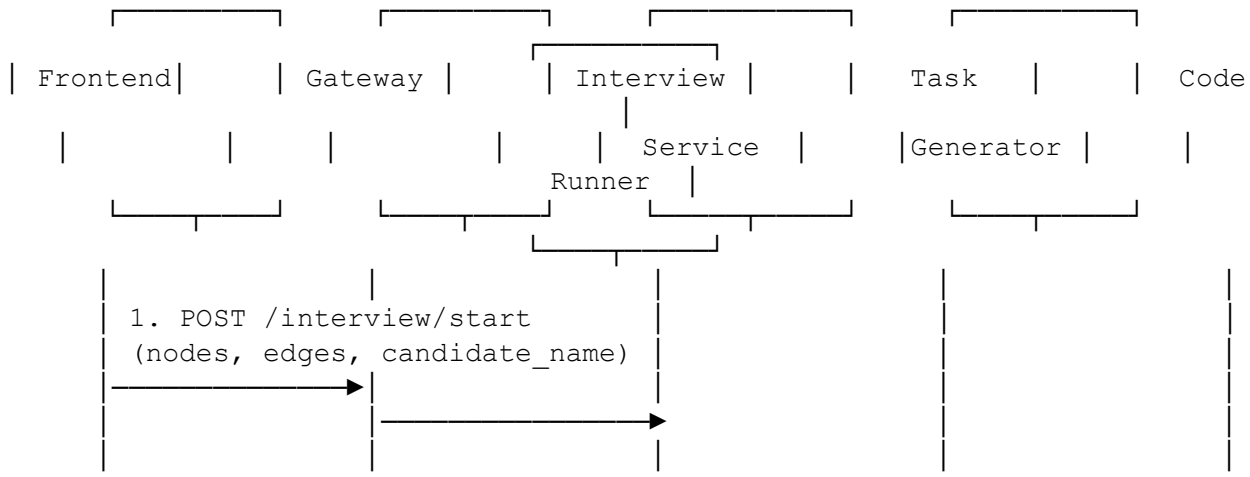
- SM-2 алгоритм интервального повторения
- Отслеживание прогресса пользователя
- Адаптивная сложность задач

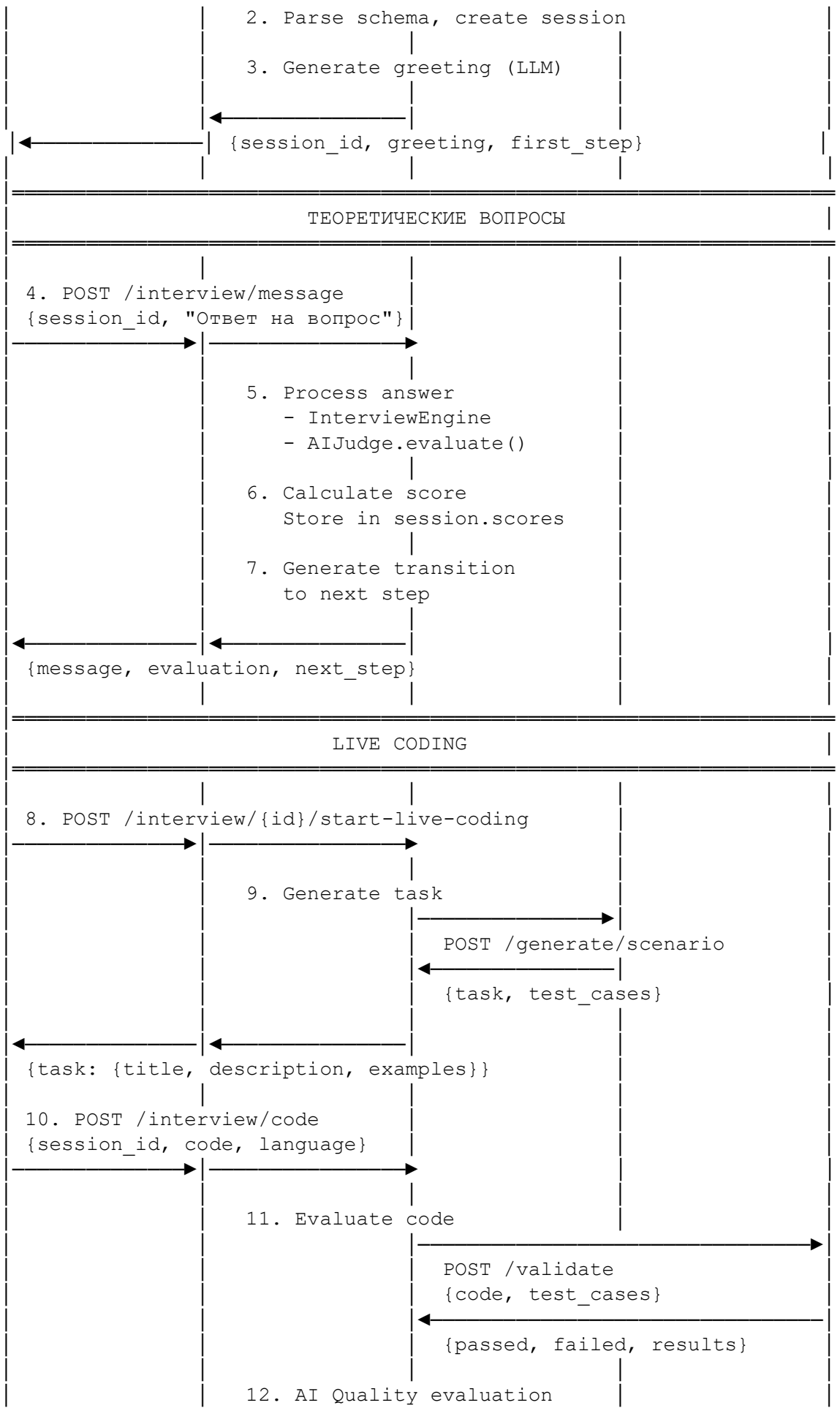
6. Library Service (порт 8001)
CRUD для контента

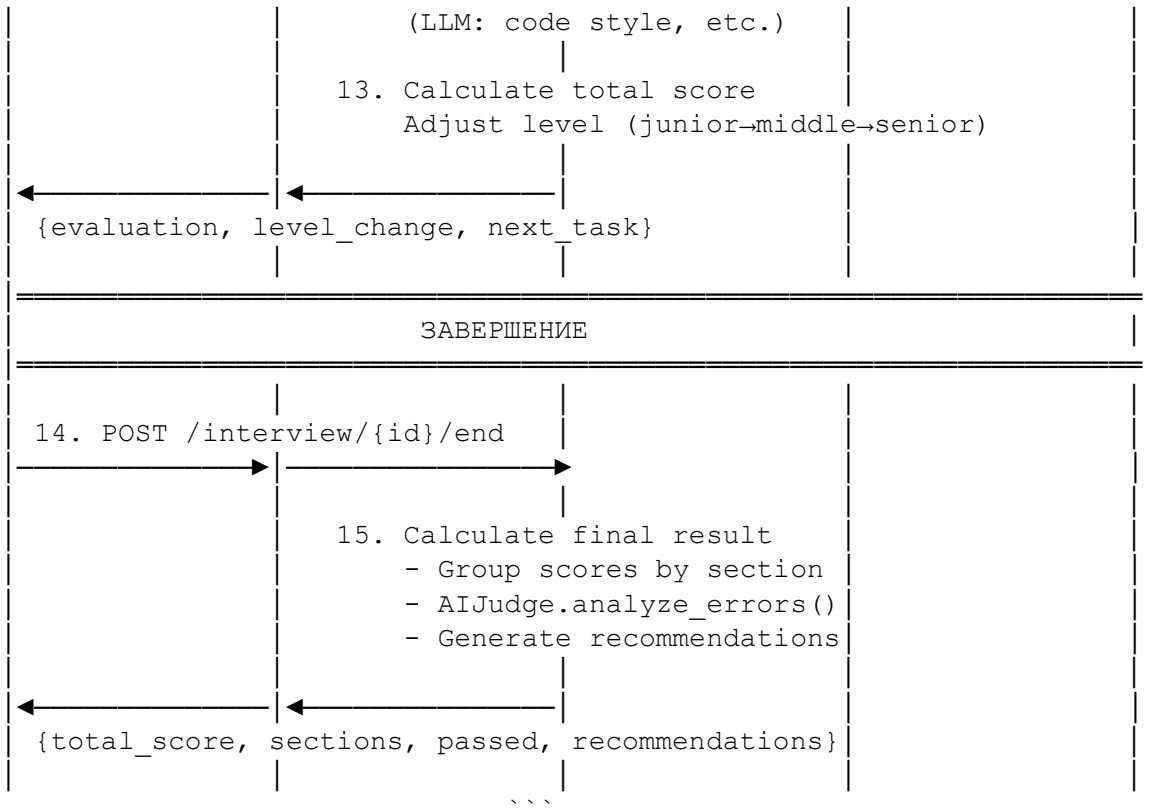
- Секции, папки, задачи
- SQLite база данных
- Иерархическая структура

Процесс выполнения интервью

Sequence Diagram: Типичное интервью

```


```


2. Parse schema, create session
3. Generate greeting (LLM)
{session_id, greeting, first_step}
ТЕОРЕТИЧЕСКИЕ ВОПРОСЫ
4. POST /interview/message
{session_id, "Ответ на вопрос"}
5. Process answer
- InterviewEngine
- AIJudge.evaluate()
6. Calculate score
Store in session.scores
7. Generate transition
to next step
{message, evaluation, next_step}
LIVE CODING
8. POST /interview/{id}/start-live-coding
9. Generate task
POST /generate/scenario
{task, test_cases}
{task: {title, description, examples}}
10. POST /interview/code
{session_id, code, language}
11. Evaluate code
POST /validate
{code, test_cases}
{passed, failed, results}
12. AI Quality evaluation



Детальный процесс интервью

Фаза 1: Инициализация

```python
# 1. Frontend отправляет схему интервью
POST /interview/start
{
  "nodes": [
    {"id": "start", "type": "start", "data": {"label": "Начало"}},
    {"id": "python-basics", "type": "skill-group", "data": {"label": "Python Basics"}},
    {"id": "q1", "type": "skill-check", "data": {"label": "Типы данных", "points": 10}},
    {"id": "live-coding", "type": "skill-group", "data": {"label": "Live Coding"}},
    {"id": "task1", "type": "skill-check", "data": {"label": "Алгоритм", "points": 20}},
    {"id": "end", "type": "end", "data": {"label": "Завершение"}}
  ],
  "edges": [
    {"source": "start", "target": "python-basics"},
    {"source": "python-basics", "target": "q1"},
    {"source": "q1", "target": "live-coding"},
    {"source": "live-coding", "target": "task1"},
    {"source": "task1", "target": "end"}
  ],
  "candidate_name": "Иван Петров"
}
```

```
# 2. Interview Service создаёт сессию
InterviewSession:
  session_id: "uuid-xxx"
  candidate_name: "Иван Петров"
  steps: [parsed steps from schema]
  current_step_index: 0
  scores: {}
  history: []
  current_level: "junior"
```

Фаза 2: Теоретические вопросы

```python
# 3. HR Bot генерирует вопрос
InterviewEngine.generate_question(session, step):
  prompt = f"""
  Текущий этап: {step.label}
  Важность: {step.importance}
  Задай вопрос кандидату...
  """
→ LLM → "Расскажите о различиях между list и tuple в Python"

# 4. Кандидат отвечает
POST /interview/message
{
  "session_id": "uuid-xxx",
  "message": "List изменяемый, tuple неизменяемый..."
}

# 5. AIJudge оценивает ответ
AIJudge.evaluate_answer():
  → LLM анализирует ответ
→ Возвращает: {points: 8, max: 10, feedback: "Хороший ответ..."}
```

Фаза 3: Live Coding

```python
# 6. Переход к Live Coding
POST /interview/{session_id}/start-live-coding

# 7. Task Generator создаёт задачу
InterviewEngine.generate_live_coding_task(session, step, level="junior"):
  → POST task-generator:8002/generate/scenario
  → Возвращает задачу с тестами

# 8. Кандидат пишет код
POST /interview/code
{
  "session_id": "uuid-xxx",
  "code": "def solution(arr): return max(arr)",
  "language": "python"
}
```

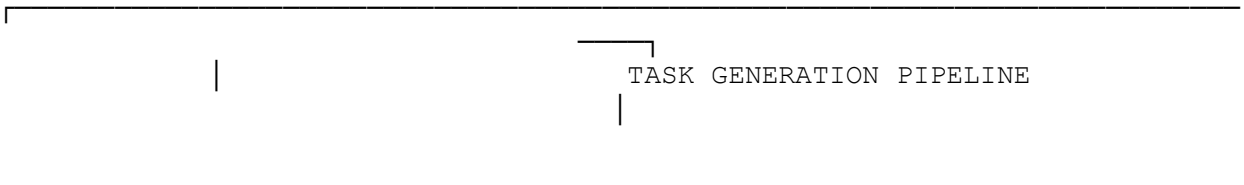
```
# 9. Оценка кода
InterviewEngine.evaluate_code():
# a) Запуск тестов
→ POST code-runner:8003/validate
→ {passed: 4, failed: 1, tests: [...]}

# b) Оценка качества (LLM)
→ Анализ читаемости, эффективности, стиля
→ {score: 0.8, feedback: "Хорошо, но можно оптимизировать"}

# c) Итоговый балл
test_score = (4/5) * 20 * 0.7 = 11.2  # 70% за корректность
quality_score = 0.8 * 20 * 0.3 = 4.8   # 30% за качество
total = 16 баллов из 20

# 10. Адаптация уровня
if all_passed:
session.current_level = "middle"  # Повышение
else:
session.current_level = "junior"  # Остаётся
```

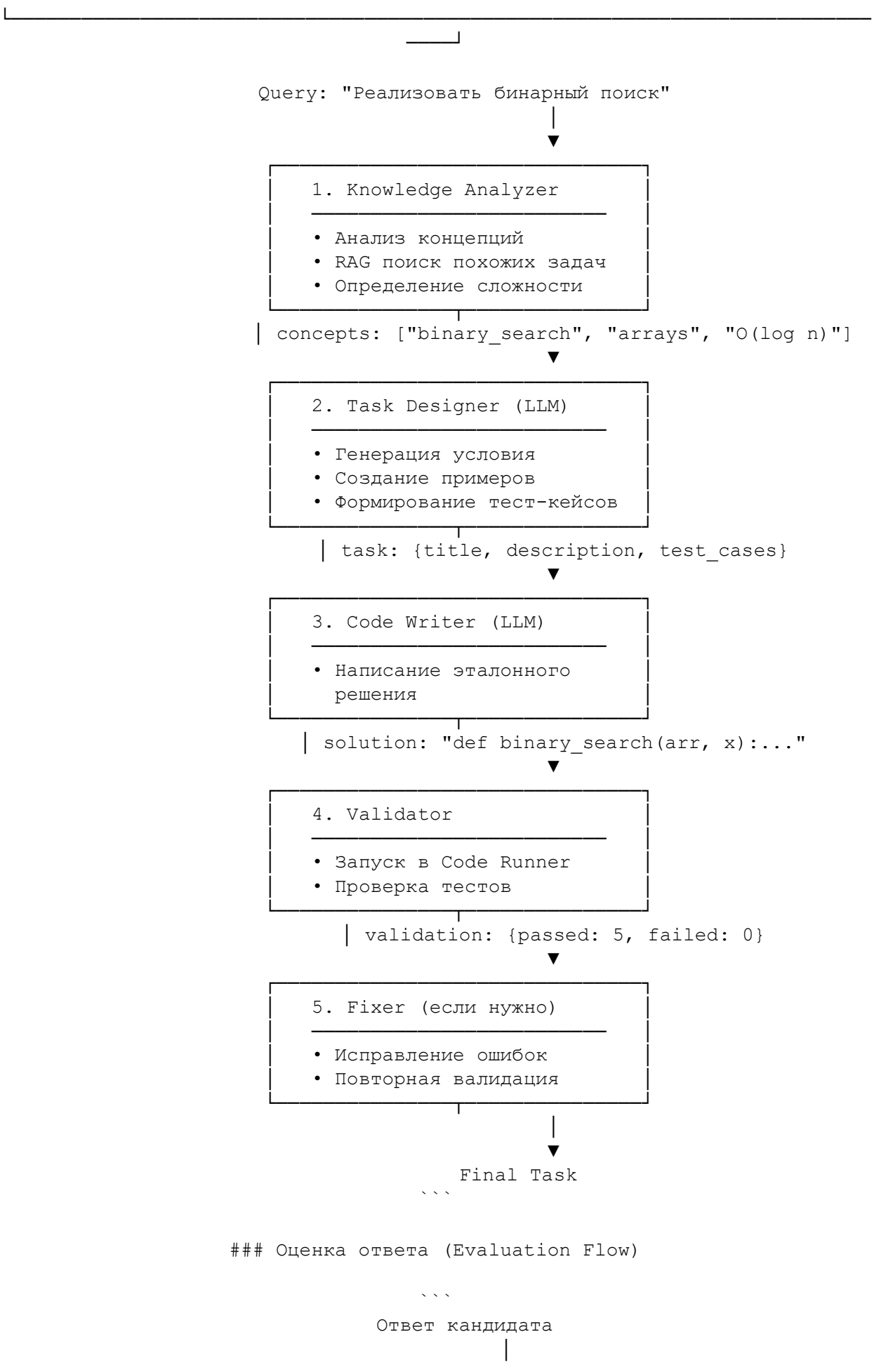
Фаза 4: Завершение

```python
# 11. Подсчёт итогов
_calculate_final_result(session):
sections = {
"Python Basics": {earned: 8, max: 10},
"Live Coding": {earned: 16, max: 20}
}
total = 24 / 30 = 80%
passed = True  # >= 60%

# AI анализ ошибок
main_errors = AIJudge.analyze_main_errors()
→ ["Недостаточно глубокое понимание GIL"]

# Рекомендации
recommendations = AIJudge.generate_recommendations()
→ ["Изучить многопоточность", "Практиковать алгоритмы"]
```

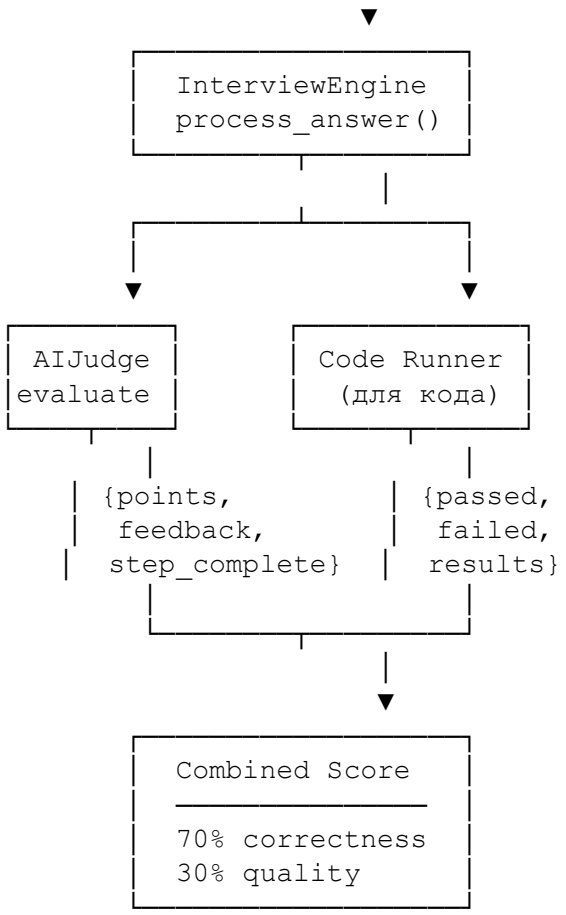
Потоки данных

Генерация задачи (Task Generator Pipeline)

```


```



Оценка ответа (Evaluation Flow)




```

---

## API взаимодействия

### Interview Service API

| Endpoint | Method | Описание |
|---|---|---|
| `/interview/start` | POST | Начать интервью |
| `/interview/start/stream` | POST | Начать с SSE стримингом |
| `/interview/message` | POST | Отправить сообщение |
| `/interview/message/stream` | POST | Сообщение с SSE |
| `/interview/code` | POST | Отправить код |
| `/interview/{id}/start-live-coding` | POST | Начать Live Coding |
| `/interview/{id}/status` | GET | Статус интервью |
| `/interview/{id}/history` | GET | История чата |
| `/interview/{id}/skip` | POST | Пропустить шаг |
| `/interview/{id}/end` | POST | Завершить интервью |
| `/ws/interview/{id}` | WS | WebSocket чат |

### Task Generator API

| Endpoint | Method | Описание |
|---|---|---|
| `/generate` | POST | Гибкая генерация |
| `/generate/hashtag` | POST | По хэштегам |
| `/generate/multifile` | POST | Многофайловая задача |
```

`/generate/scenario`	POST	Динамический сценарий
`/scenario-types`	GET	Список типов сценариев
`/templates`	GET	Список шаблонов

Code Runner API

Endpoint	Method	Описание
`/run`	POST	Выполнить код
`/validate`	POST	Валидация с тестами

Конфигурация Docker

Порты сервисов

Сервис	Внутренний порт	Внешний порт
API Gateway	8000	8000
Library Service	8001	8001
Task Generator	8002	8002
Code Runner	8003	8003
RAG Service	8004	8004
Knowledge Service	8005	8005
Learning Service	8006	8006
Task Architect	8007	8007
Task API	8010	8010
Interview Service	8011	8011
Legacy Gateway	8000	8080

Volumes

- `library_data` - SQLite база Library Service
- `rag_data` - Векторные индексы RAG
- `knowledge_data` - FAISS индексы Knowledge Service
- `learning_data` - Данные прогресса Learning Service

Заключение

Система представляет собой модульную микросервисную архитектуру, где:

1. **Interview Service** - центральный компонент для проведения интервью
2. **Task Generator** - гибкая генерация задач с помощью AI
3. **Code Runner** - безопасное выполнение кода в sandbox
4. **Knowledge/Learning Services** - адаптивное обучение

Все сервисы общаются через REST API, а для real-time взаимодействия используется WebSocket.