



Искусственный интеллект в разработке ПО

Егор Баранов, Solution Architect



Введение

Данный доклад является обзорным, рассматривая современные возможности искусственного интеллекта в разработке программного обеспечения, при этом со значительным углублением в технические аспекты, архитектуру и устройство подобных систем.

Ниже представлены основные составляющие доклада:

Введение

Данный доклад является обзорным, рассматривая современные возможности искусственного интеллекта в разработке программного обеспечения, при этом со значительным углублением в технические аспекты, архитектуру и устройство подобных систем.

Ниже представлены основные составляющие доклада:

Углубимся в
устройство ИИ-
систем для
разработчиков,
разберем основные
компоненты и их
составляющие

Введение

Данный доклад является обзорным, рассматривая современные возможности искусственного интеллекта в разработке программного обеспечения, при этом со значительным углублением в технические аспекты, архитектуру и устройство подобных систем.

Ниже представлены основные составляющие доклада:

Углубимся в
устройство ИИ-
систем для
разработчиков,
разберем основные
компоненты и их
составляющие

Поговорим о том,
как собирается
контекст и устроены
такие технологии,
как RAG, function
calling и model
agnostic

Введение

Данный доклад является обзорным, рассматривая современные возможности искусственного интеллекта в разработке программного обеспечения, при этом со значительным углублением в технические аспекты, архитектуру и устройство подобных систем.

Ниже представлены основные составляющие доклада:

Углубимся в устройство ИИ-систем для разработчиков, разберем основные компоненты и их составляющие

Поговорим о том, как собирается контекст и устроены такие технологии, как RAG, function calling и model agnostic

Проанализируем будущее подобных систем и разберем, чем отличаются понятия AI Assisted/Driven/Led development

Введение

Данный доклад является обзорным, рассматривая современные возможности искусственного интеллекта в разработке программного обеспечения, при этом со значительным углублением в технические аспекты, архитектуру и устройство подобных систем.

Ниже представлены основные составляющие доклада:

Углубимся в устройство ИИ-систем для разработчиков, разберем основные компоненты и их составляющие

Поговорим о том, как собирается контекст и устроены такие технологии, как RAG, function calling и model agnostic

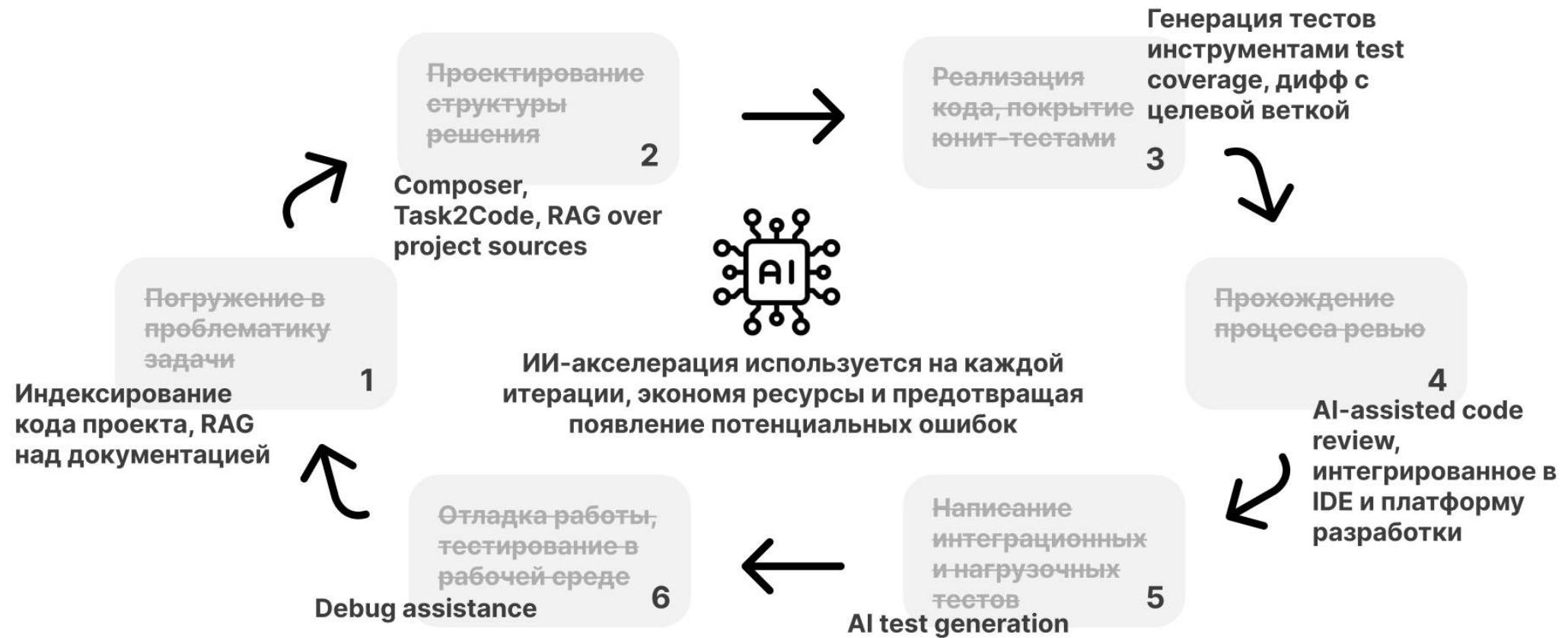
Проанализируем будущее подобных систем и разберем, чем отличаются понятия AI Assisted/Driven/Led development

Найдем реальные причины использовать ИИ в разработке, рассмотрим возможности и их принцип работы

Классический SDLC



AI-Assisted/Driven SDLC



Ориентиры в дизайне ИИ-систем

Скорость ответа, генерации, стриминг и кеширование

Ориентиры в дизайне ИИ-систем

Скорость ответа, генерации, стриминг и кеширование

Размер контекста, который система способна
обработать

Ориентиры в дизайне ИИ-систем

Скорость ответа, генерации, стриминг и кеширование

Размер контекста, который система способна
обработать

Точность ответа, минимизация количества
артефактов

Ориентиры в дизайне ИИ-систем

Скорость ответа, генерации, стриминг и кеширование

Стабильность работы системы,
минимизация ошибок и «сломанных»
генераций

Размер контекста, который система способна
обработать

Точность ответа, минимизация количества
артефактов

Ориентиры в дизайне ИИ-систем

Скорость ответа, генерации, стриминг и кеширование

Стабильность работы системы,
минимизация ошибок и «сломанных»
генераций

Размер контекста, который система способна
обработать

Масштабируемость и
взаимозаменяемость компонентов,
включая модели

Точность ответа, минимизация количества
артефактов

ИИ-ассистенты на сегодня

No Assistance

Традиционная
разработка без
использования ИИ

Стандартная
разработка в IDE,
включая
встроенное
автодополнение

ИИ-ассистенты на сегодня

No Assistance	Code Completion
Традиционная разработка без использования ИИ	Появление первых ассистентов на рынке, имеющих только возможности автокомплита
Стандартная разработка в IDE, включая встроенное автодополнение	Code completion, code chat with current file context 0.5-7B params 2-10K context

ИИ-ассистенты на сегодня

AI Assisted Development

No Assistance	Code Completion	Code Creation
Традиционная разработка без использования ИИ	Появление первых ассистентов на рынке, имеющих только возможности автокомплита	Генерация кода и тестов ассистентом с учетом контекста всего проекта
Стандартная разработка в IDE, включая встроенное автодополнение	Code completion, code chat with current file context	Code generation, testcase generation, composer, RAG over project sources

– 2024

Основные компоненты систем

Client Application



Retrieval Pipeline



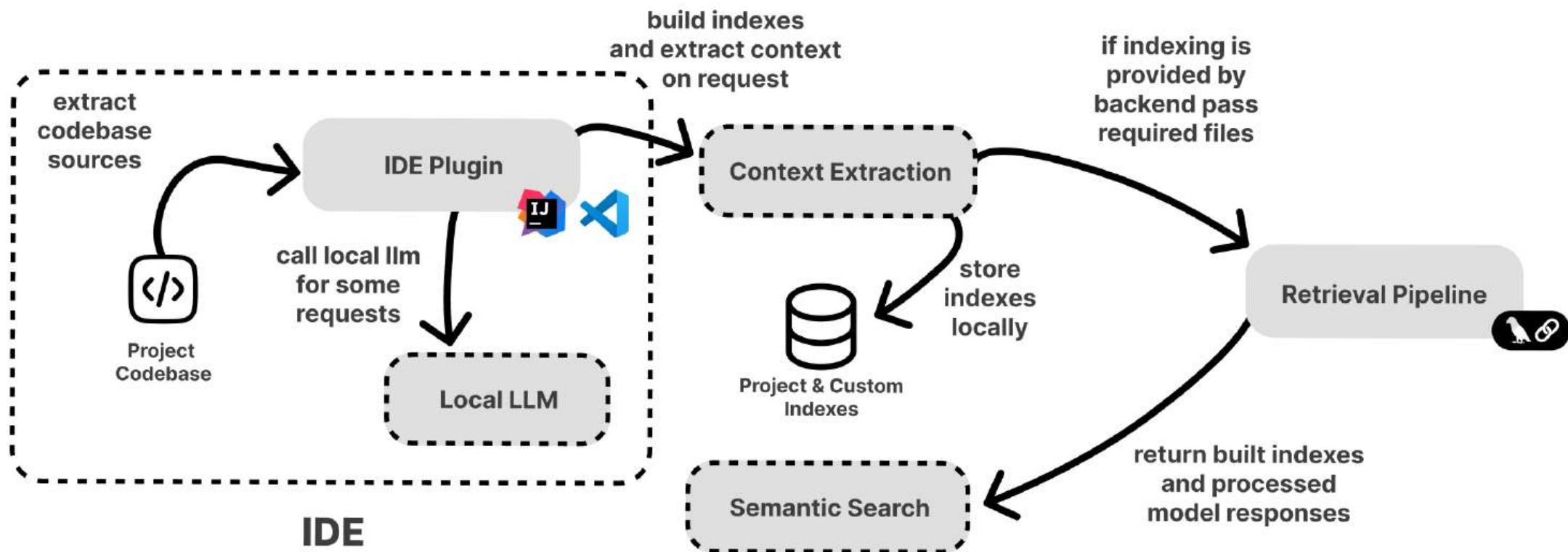
Model Runtime



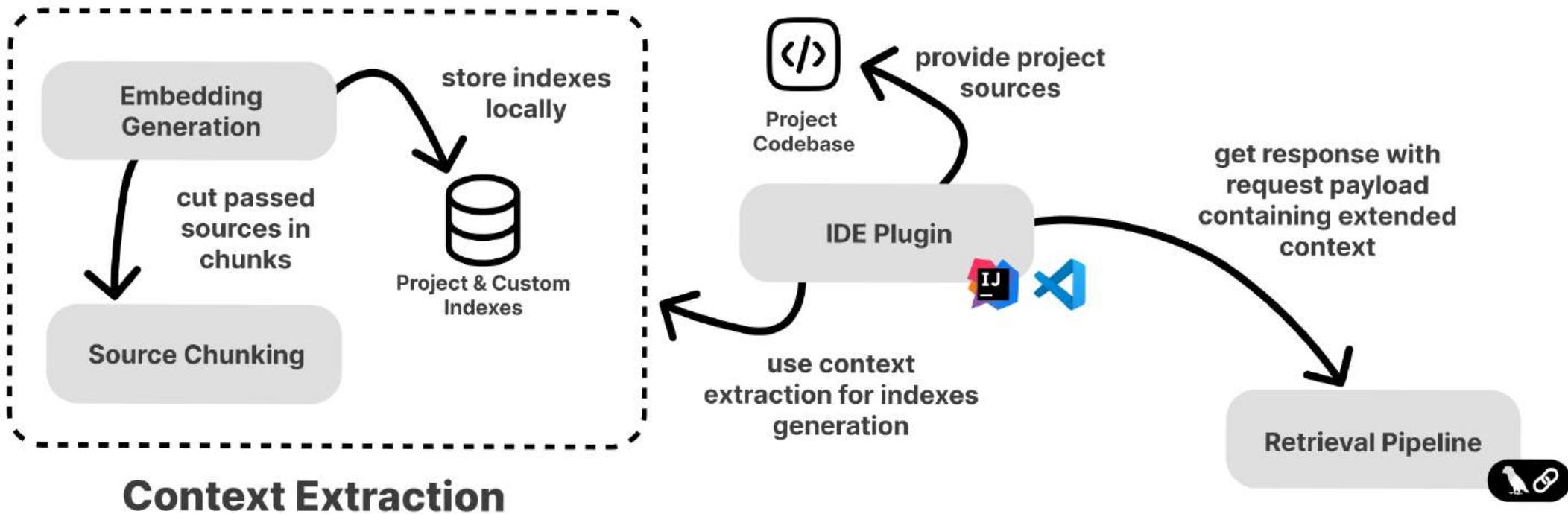
LLMs



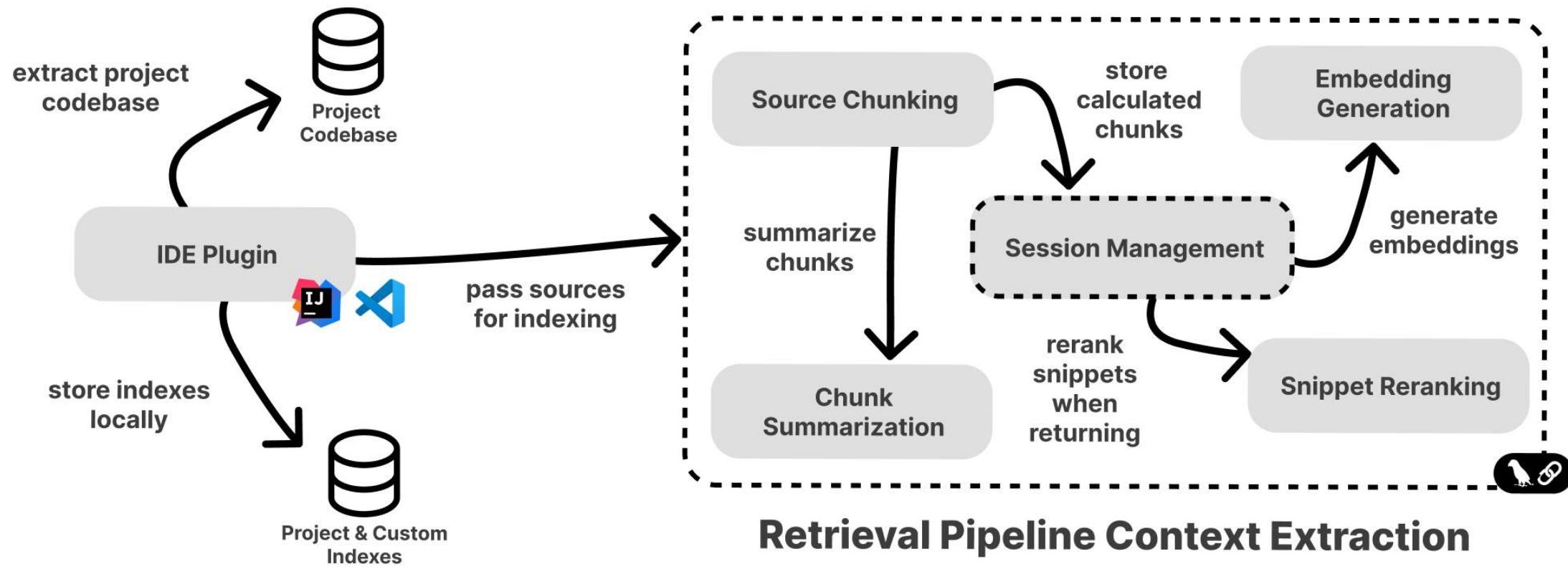
Клиентские приложения



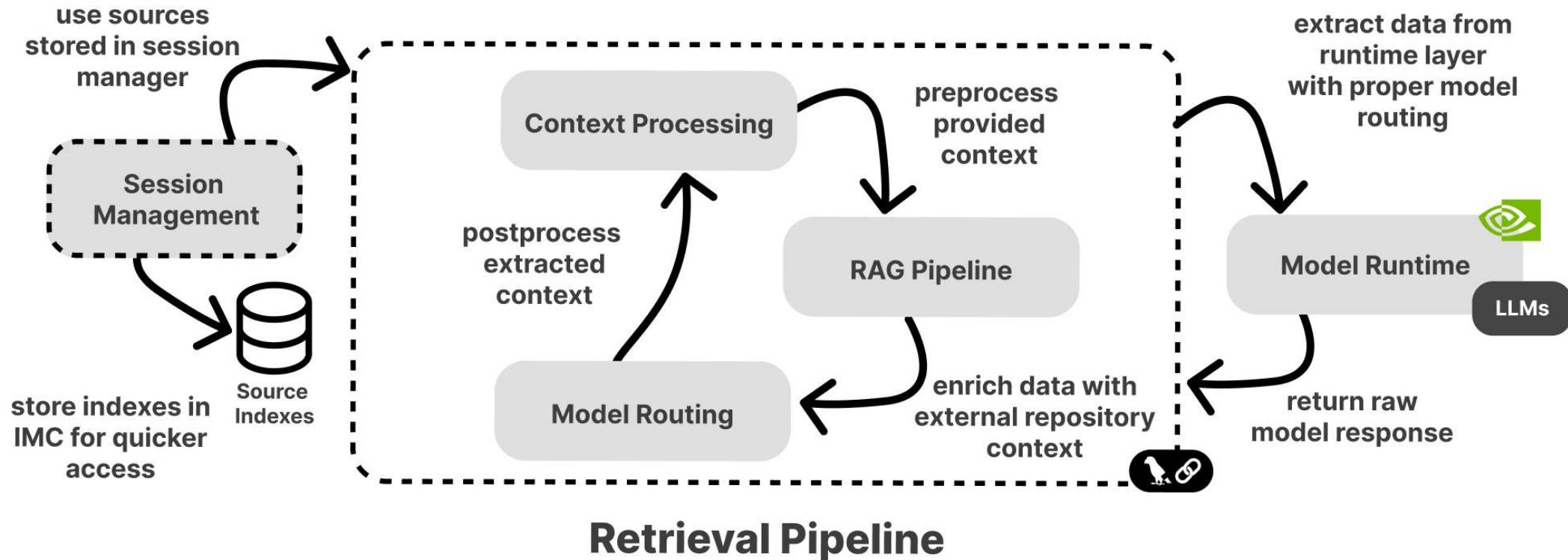
RAG над проектом (client-side)



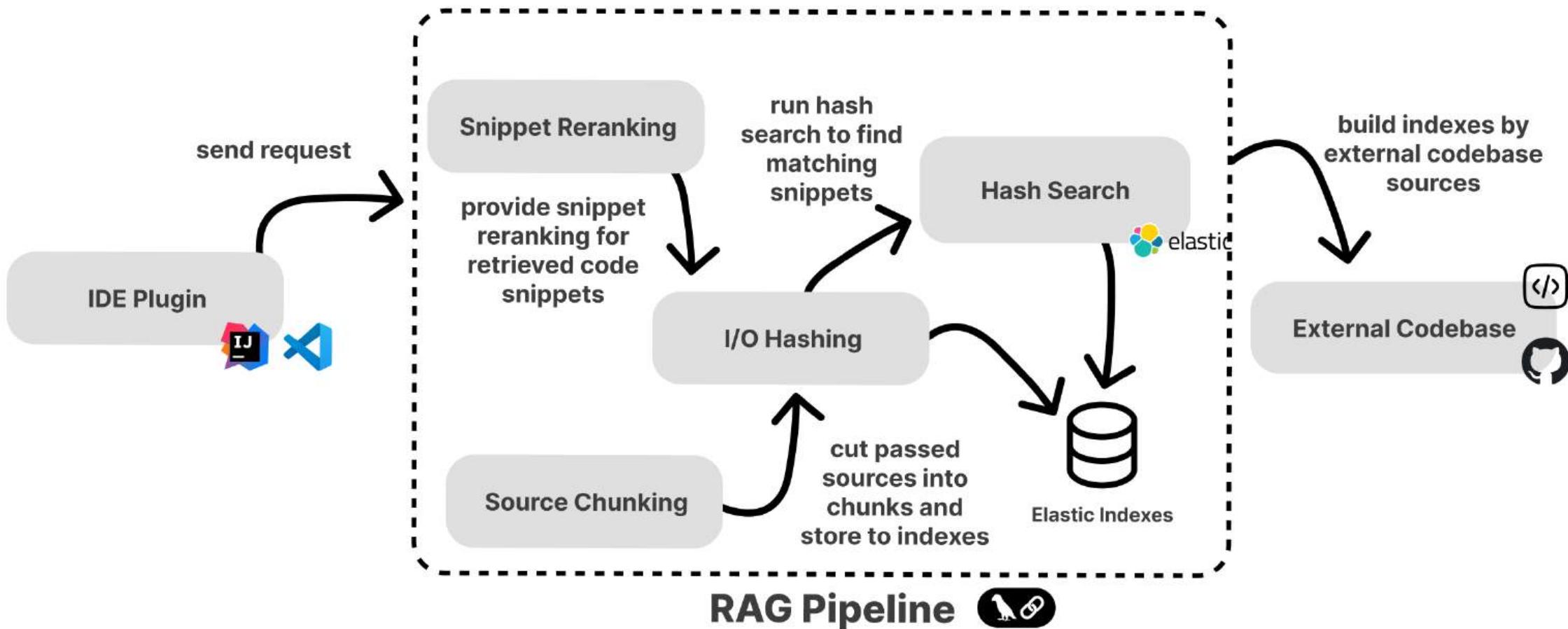
RAG над проектом (server-side)



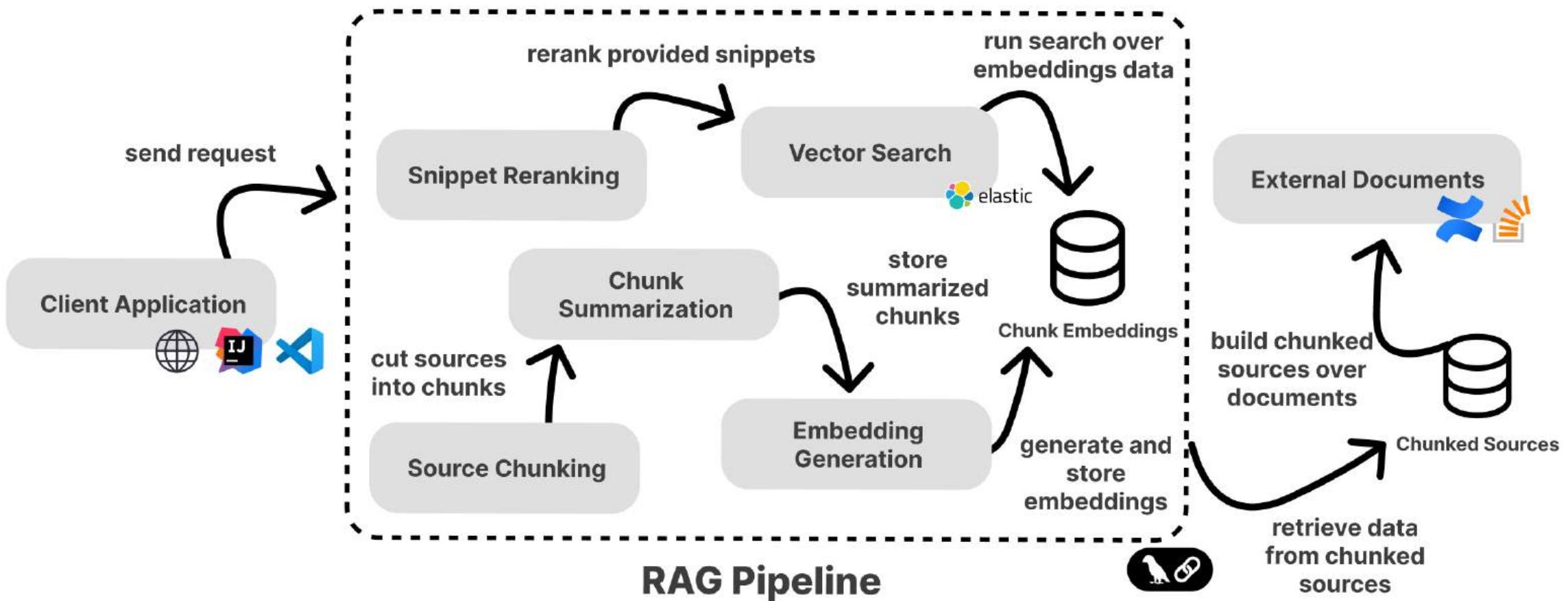
Retrieval pipeline



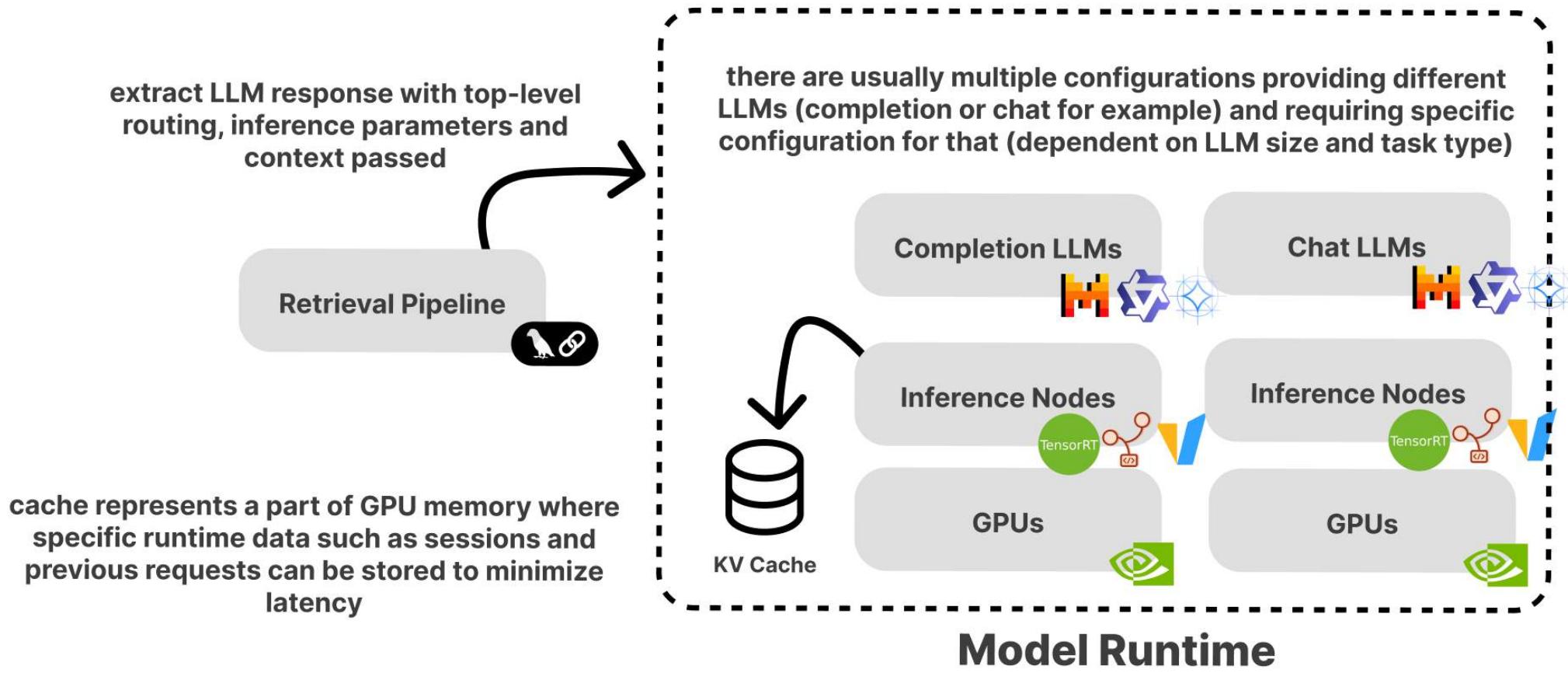
RAG над кодовой базой (GitHub)



RAG над документами



Model runtime

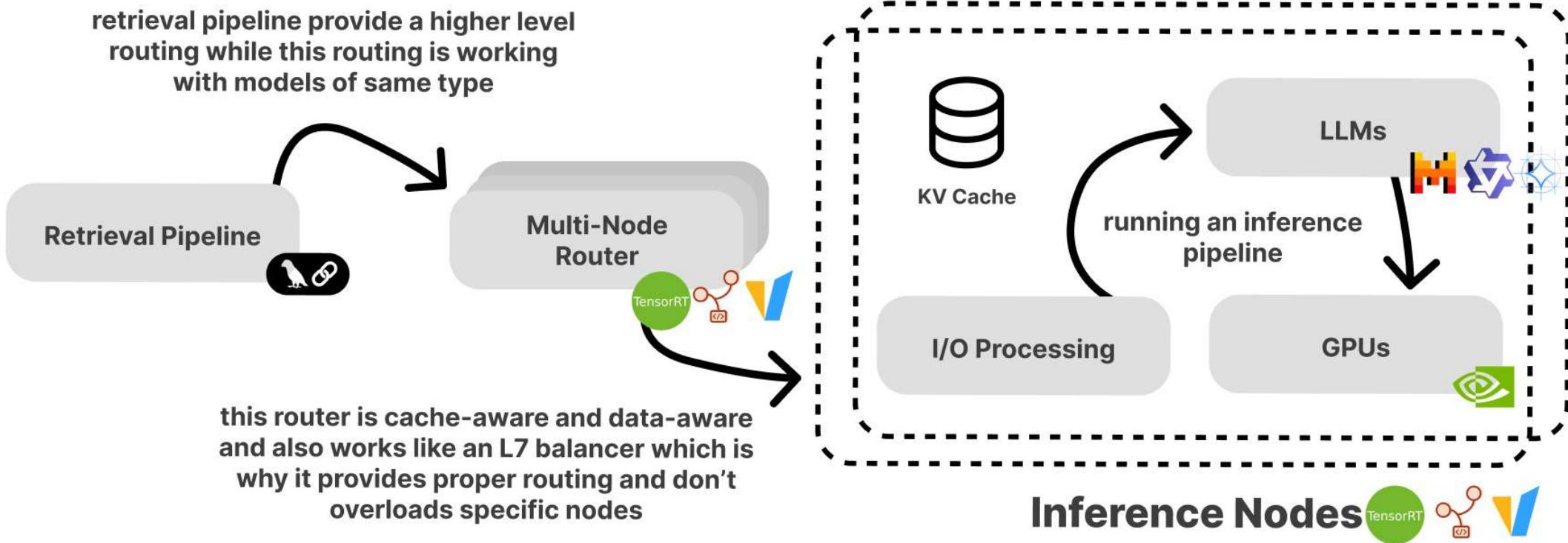


Разберем параллелизм

На уровне запросов

С использованием multi-node роутера, распределяющего нагрузку, учитывая ресурсы нод, такие как количество свободного кеша и количества обрабатываемых запросов

Parallel request processing



Разберем параллелизм

На уровне запросов

С использованием multi-node роутера, распределяющего нагрузку, учитывая ресурсы нод, такие как количество свободного кеша и количества обрабатываемых запросов

Разберем параллелизм

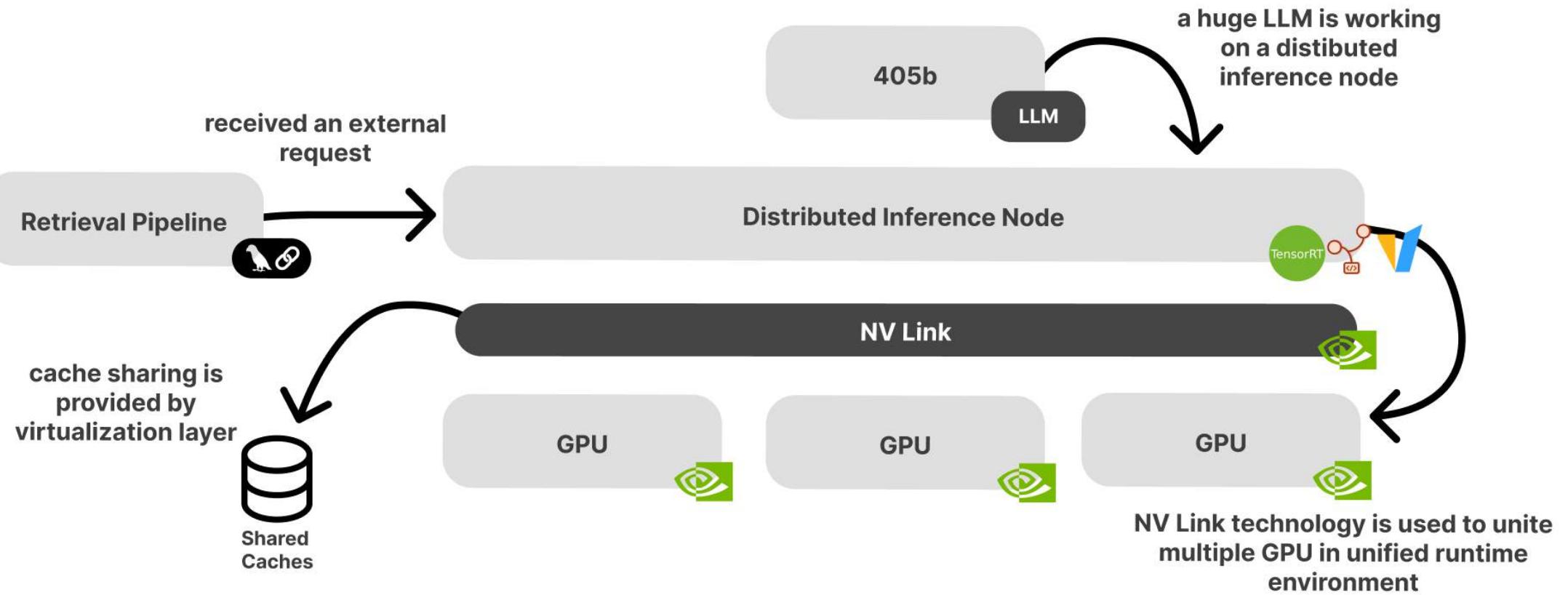
На уровне запросов

С использованием multi-node роутера, распределяющего нагрузку, учитывая ресурсы нод, такие как количество свободного кеша и количества обрабатываемых запросов

Объединение видеокарт

Большие LLM зачастую не помещаются на одну видеокарту, для чего используется технология виртуализации NV Link, позволяющая объединить несколько GPU в один рантайм

Multi-GPU inference



Разберем параллелизм

На уровне запросов

С использованием multi-node роутера, распределяющего нагрузку, учитывая ресурсы нод, такие как количество свободного кеша и количества обрабатываемых запросов

Объединение видеокарт

Большие LLM зачастую не помещаются на одну видеокарту, для чего используется технология виртуализации NV Link, позволяющая объединить несколько GPU в один рантайм

Разберем параллелизм

На уровне запросов

С использованием multi-node роутера, распределяющего нагрузку, учитывая ресурсы нод, такие как количество свободного кеша и количества обрабатываемых запросов

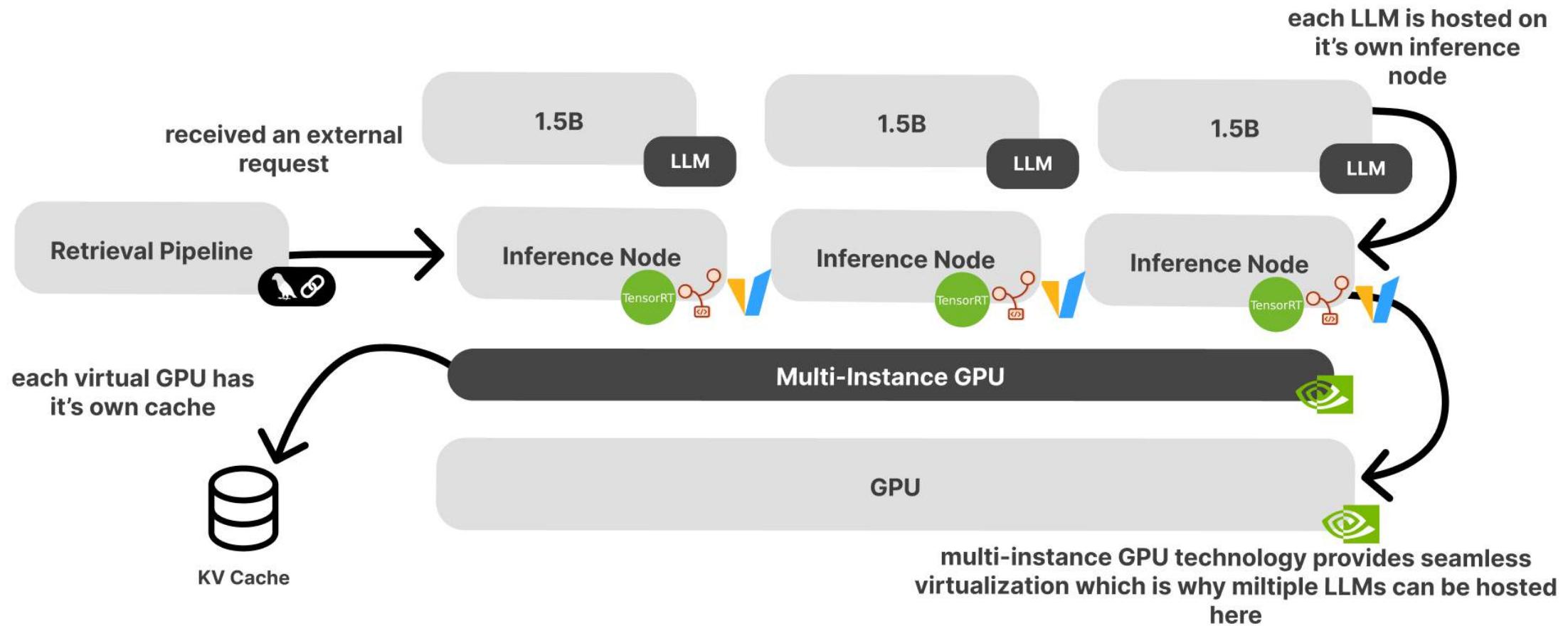
Объединение видеокарт

Большие LLM зачастую не помещаются на одну видеокарту, для чего используется технология виртуализации NV Link, позволяющая объединить несколько GPU в один рантайм

В рамках одной видеокарты

Когда стоит задача сэкономить ресурсы можно одновременно запустить несколько моделей на одной видеокарте, разделяя ее на виртуальные при помощи Multi-instance GPU

Multi-instance GPU



AI-assisted development systems

Client Application



Retrieval Pipeline



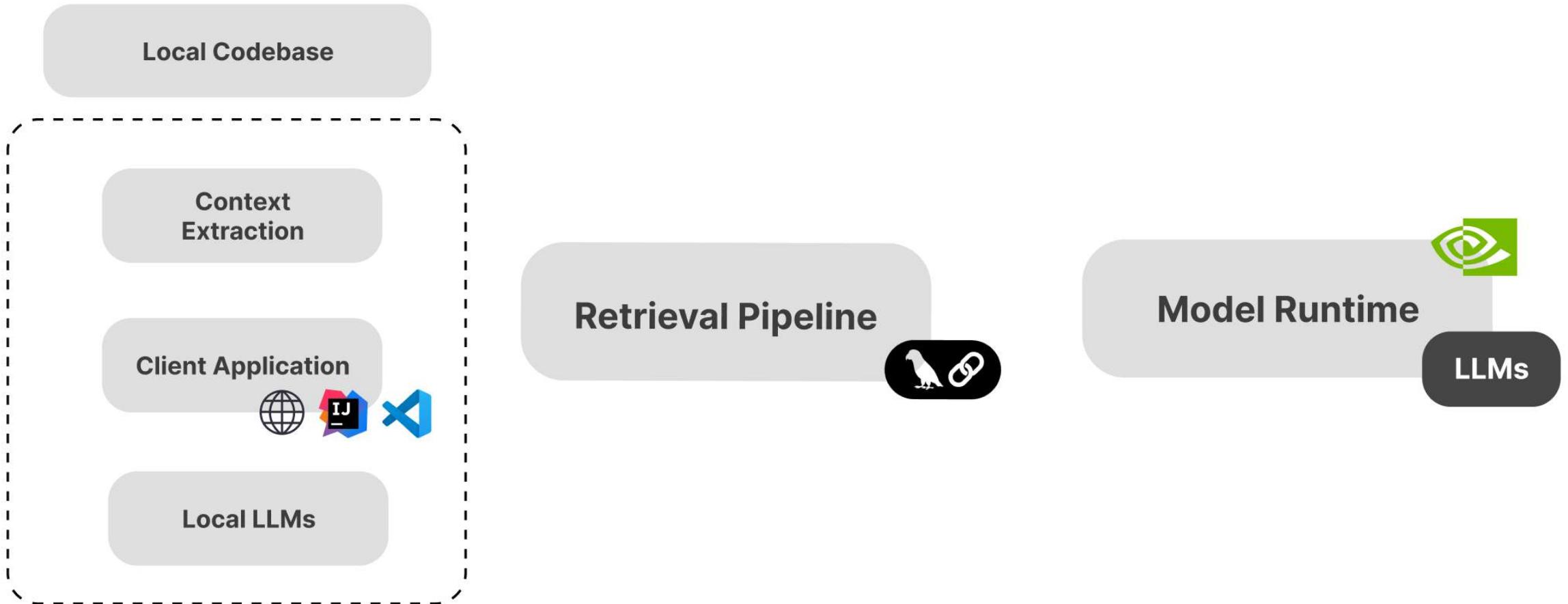
Model Runtime



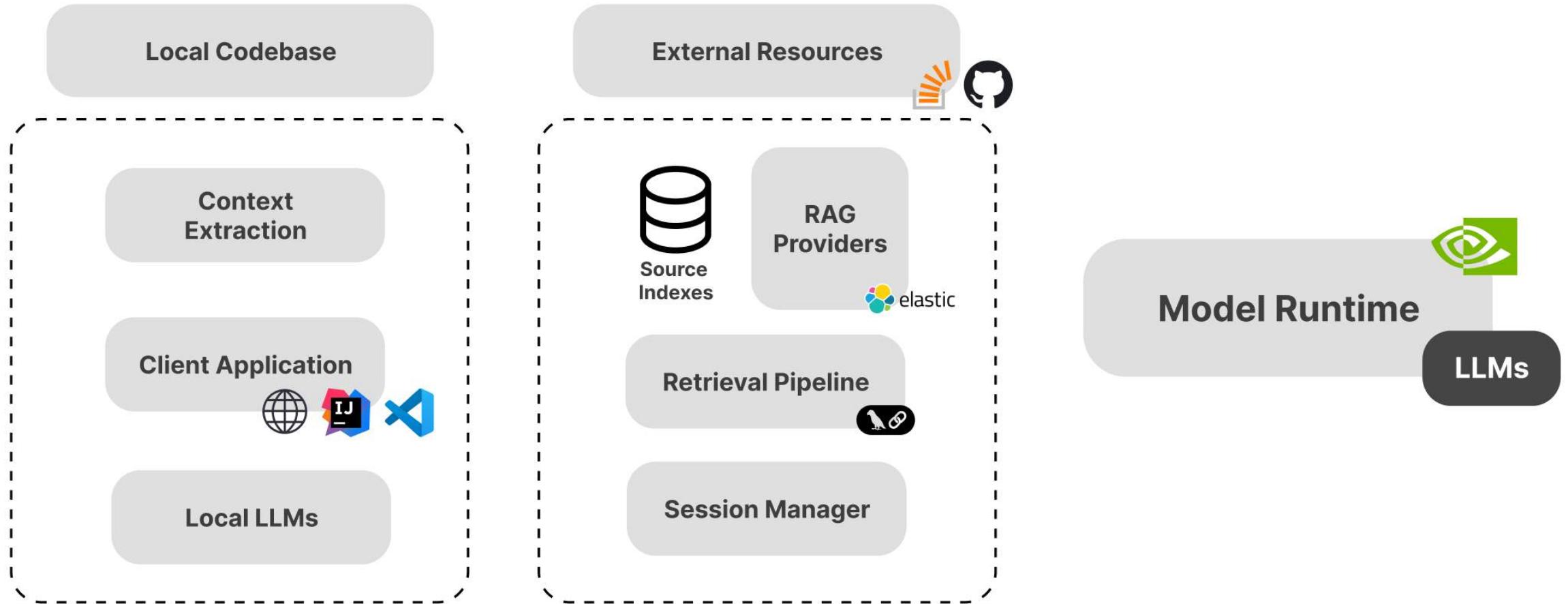
LLMs



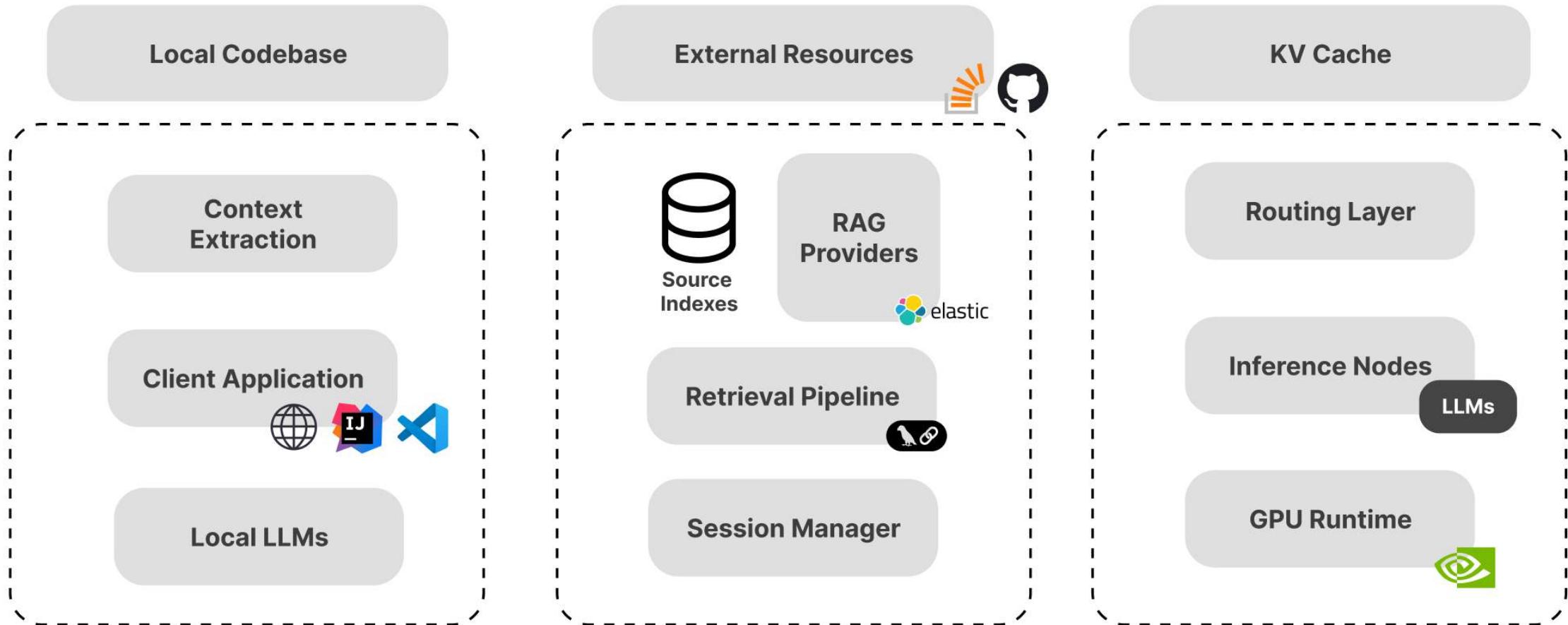
AI-assisted development systems



AI-assisted development systems



AI-assisted development systems



Ближайшее будущее ИИ-ассистентов

AI Assisted Development

No Assistance	Code Completion	Code Creation
Традиционная разработка без использования ИИ	Появление первых ассистентов на рынке, имеющих только возможности автокомплита	Генерация кода и тестов ассисстентом с учетом контекста всего проекта
Стандартная разработка в IDE, включая встроенное автодополнение	Code completion, code chat with current file context	Code generation, testcase generation, composer, RAG over project sources

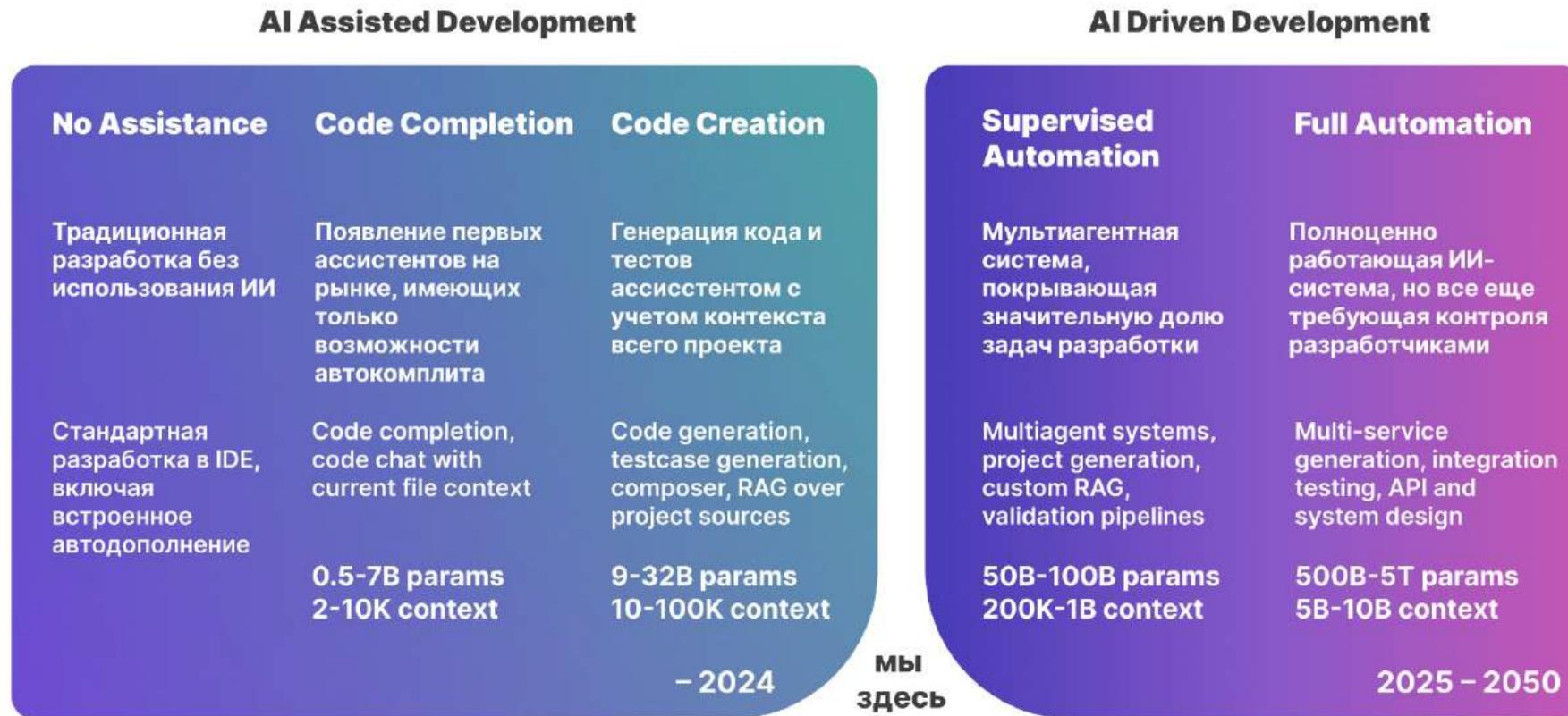
– 2024

Ближайшее будущее ИИ-ассистентов

AI Assisted Development



Ближайшее будущее ИИ-ассистентов



Возможности ИИ-ассистентов

Автодополнение кода

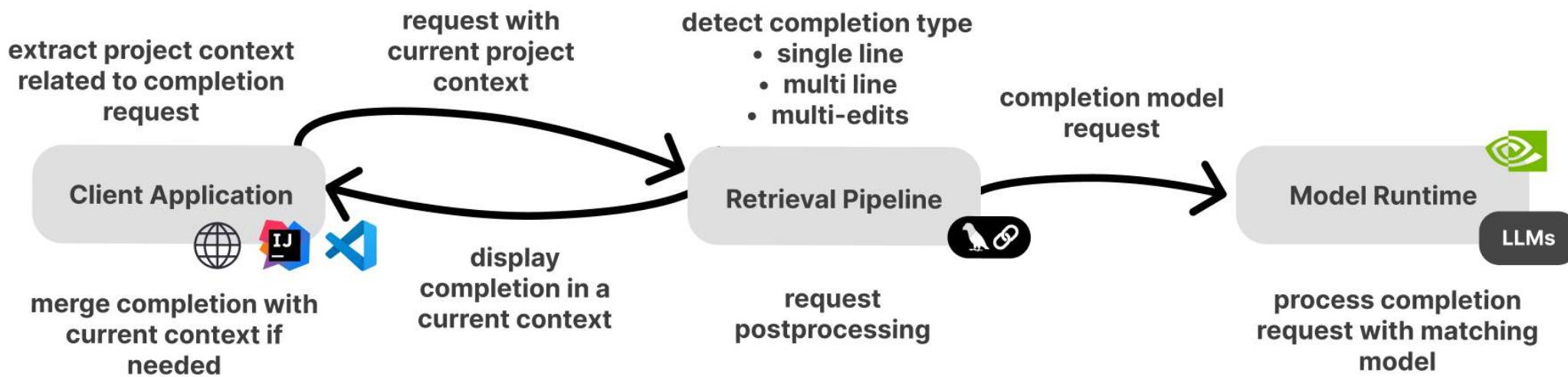
Интегрированное в редактор кода автодополнение, подсказывающее вероятные продолжения строки исходя из контекста текущего файла и проекта сразу после курсора

Автодополнение кода

```
37
38     @GetMapping("/portfolio")
39     public String portfolio( @NotNull Model model) {
40         model.addAttribute( attributeName: "title", attributeValue: "Portfolio");
41         model.addAttribute( attributeName: "description", attributeValue: "I have a few projects that I have worked on.");
42         return "portfolio";
43     }
44
45     @GetMapping("/blog") Enter to complete
46     public String blog(Model model) {
47         model.addAttribute("title", "Blog");
48         model.addAttribute("description", "I have a few blog posts that I have written.");
49         return "blog";
50     }
51 }
```



Автодополнение кода



Возможности ИИ-ассистентов

Автодополнение кода

Интегрированное в редактор кода автодополнение, подсказывающее вероятные продолжения строки исходя из контекста текущего файла и проекта сразу после курсора

Возможности ИИ-ассистентов

Автодополнение кода

Интегрированное в редактор кода автодополнение, подсказывающее вероятные продолжения строки исходя из контекста текущего файла и проекта сразу после курсора

Чат с кодовой моделью

Чат с ассистентом, работающим с кодом проекта. Может дополнительно иметь дополнительные возможности вроде коротких команд и обращений к определенным агентам

Чат с кодовой моделью

The screenshot shows the GitHub Copilot interface. On the left, there's a sidebar with icons for file navigation, search, and other tools. The main area has a header "GITHUB COPILOT: CHAT". Below it, a user named "monalisa" sends a message: "Write a unit test for this function". GitHub Copilot responds with a Python script:

```
import unittest
import datetime

class TestParseExpenses(unittest.TestCase):
    def test_parse_expenses_with_valid_input(self):
        expenses_string = '2023-01-02 -34.01 USD\n2023-01-03 2.59 DKK
        expected_output = [(datetime.datetime(2023, 1, 2),
                            float(-34.01), 'USD'),
                            (datetime.datetime(2023, 1, 3),
                            float(2.59), 'DKK')]
        self.assertEqual(parse_expenses(expenses_string),
                        expected_output)

    def test_parse_expenses_with_empty_input(self):
        expenses_string = ''
        expected_output = []
        self.assertEqual(parse_expenses(expenses_string),
                        expected_output)

if __name__ == '__main__':
    unittest.main()
```

A note below the code says: "The code assumes that the datetime module is imported. The test cases cover different scenarios such as valid input, empty input, input with comments, invalid date format, and invalid value format."

On the right, the code editor shows a file named "module.py" with the following content:

```
import datetime

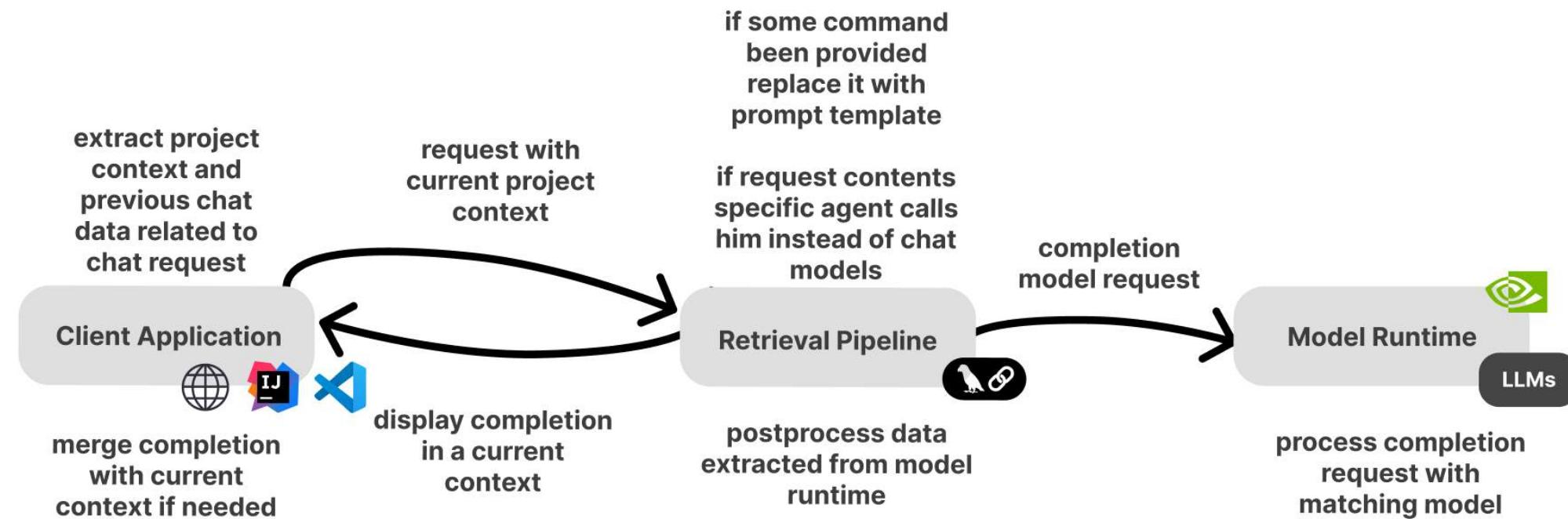
def parse_expenses(expenses_string):
    """Parse the list of expenses and return the list of triples (date, amount, currency).
    Ignore lines starting with #.
    Parse the date using datetime.
    Example expenses_string:
        2023-01-02 -34.01 USD
        2023-01-03 2.59 DKK
        2023-01-03 -2.72 EUR
    """
    expenses = []

    for line in expenses_string.splitlines():
        if line.startswith("#"):
            continue
        date, value, currency = line.split(" ")
        expenses.append(datetime.datetime.strptime(date, "%Y-%m-%d"),
                        float(value),
                        currency))

    return expenses

expenses_data = '''2023-01-02 -34.01 USD
                    2023-01-03 2.59 DKK
                    2023-01-03 -2.72 EUR'''
```

Чат с кодовой моделью



Возможности ИИ-ассистентов

Автодополнение кода

Интегрированное в редактор кода автодополнение, подсказывающее вероятные продолжения строки исходя из контекста текущего файла и проекта сразу после курсора

Чат с кодовой моделью

Чат с ассистентом, работающим с кодом проекта. Может дополнительно иметь дополнительные возможности вроде коротких команд и обращений к определенным агентам

Возможности ИИ-ассистентов

Автодополнение кода

Интегрированное в редактор кода автодополнение, подсказывающее вероятные продолжения строки исходя из контекста текущего файла и проекта сразу после курсора

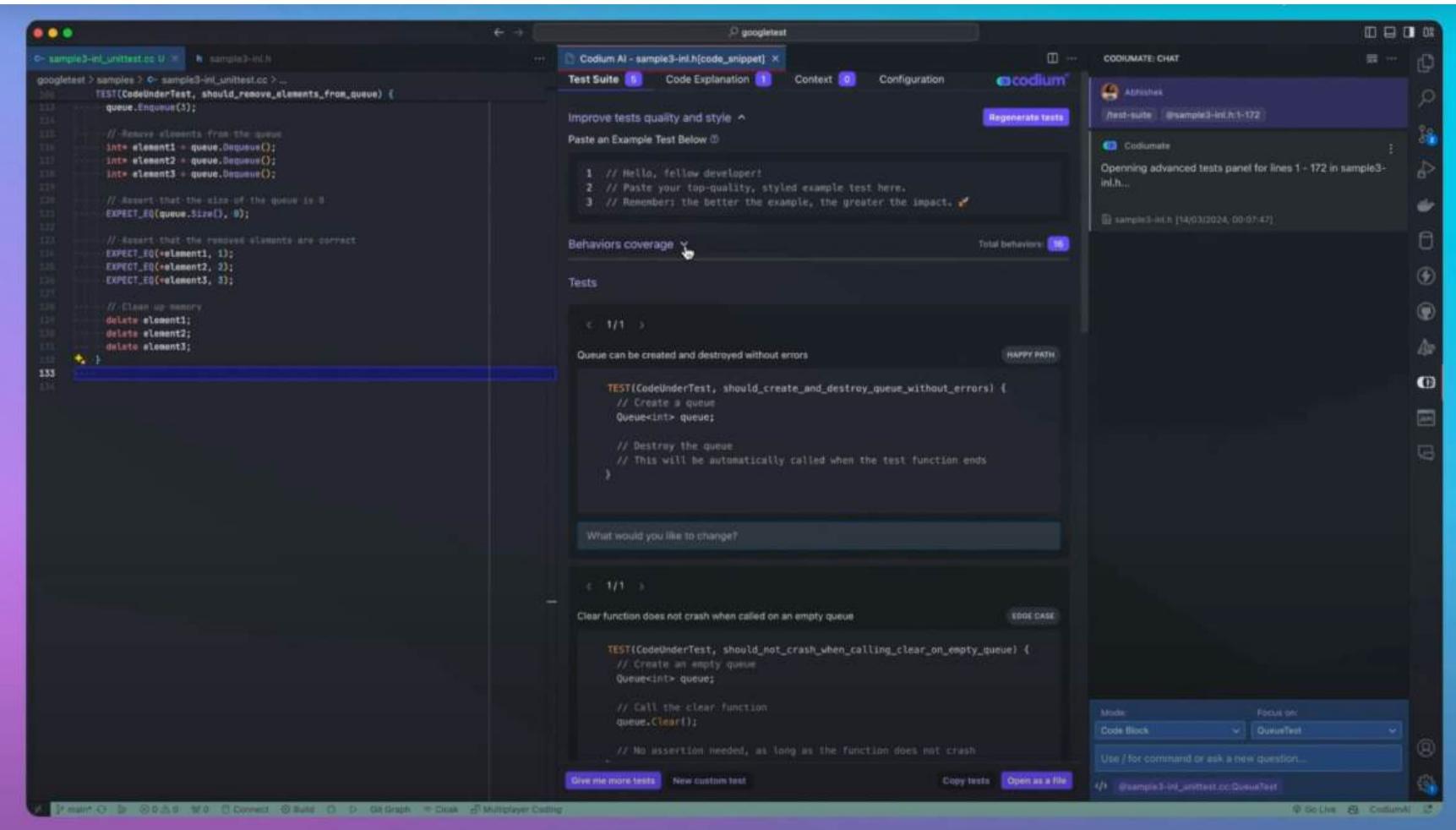
Чат с кодовой моделью

Чат с ассистентом, работающим с кодом проекта. Может дополнительно иметь дополнительные возможности вроде коротких команд и обращений к определенным агентам

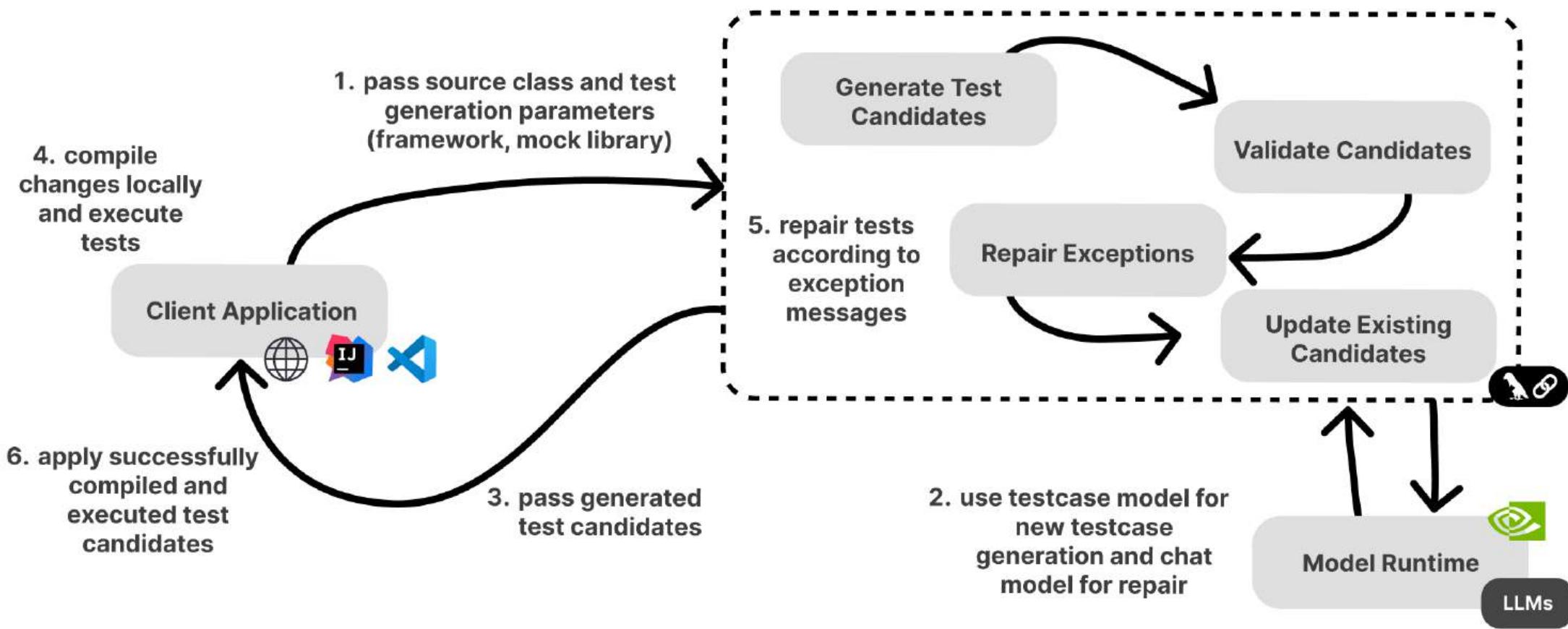
Генерация тестов

Генерация тестов, как правило выносящаяся в отдельное окно с параметрами (фреймворк, тест библиотека). Под капотом – пайплайн с компиляцией и запуском и исправлением кода

Генерация тестов



Генерация тестов



Возможности ИИ-ассистентов

Автодополнение кода

Интегрированное в редактор кода автодополнение, подсказывающее вероятные продолжения строки исходя из контекста текущего файла и проекта сразу после курсора

Чат с кодовой моделью

Чат с ассистентом, работающим с кодом проекта. Может дополнительно иметь дополнительные возможности вроде коротких команд и обращений к определенным агентам

Генерация тестов

Генерация тестов, как правило выносящаяся в отдельное окно с параметрами (фреймворк, тест библиотека). Под капотом – пайплайн с компиляцией и запуском и исправлением кода

Возможности ИИ-ассистентов

Автодополнение кода

Интегрированное в редактор кода автодополнение, подсказывающее вероятные продолжения строки исходя из контекста текущего файла и проекта сразу после курсора

Чат с кодовой моделью

Чат с ассистентом, работающий с кодом проекта. Может дополнительно иметь дополнительные возможности вроде коротких команд и обращений к определенным агентам

Генерация тестов

Генерация тестов, как правило выносящаяся в отдельное окно с параметрами (фреймворк, тест библиотека). Под капотом – пайплайн с компиляцией и запуском и исправлением кода

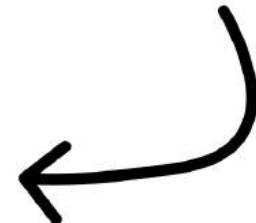
Composer (Multi-file editing)

Многофайловое редактирование в соответствии с инструкцией пользователя, реализуется через онлайн чат – тулбар, появляющийся в редакторе при выделении кода или по сочетанию клавиш

Composer (multi-file editing)

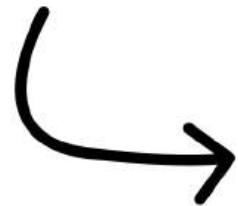
The screenshot shows the Composer interface with multiple files open: app.py, index.html, and sitemap. The sitemap file is active, displaying a snippet about rewriting UI from jQuery to Alpine.js. Below it, a snippet from index.html discusses changes to accommodate the new UI. A third snippet from app.py lists five steps for the backend update. At the bottom, there's an 'index.html' tab, an 'app.py' tab, and a command input field with a placeholder 'Type your instructions, / for commands'. A status bar at the bottom indicates a Pylance error: 'Import "supabase" could not be resolved'.

**Встроенный онлайн-чат
с ассистентом и
возможностью
написания инструкций**



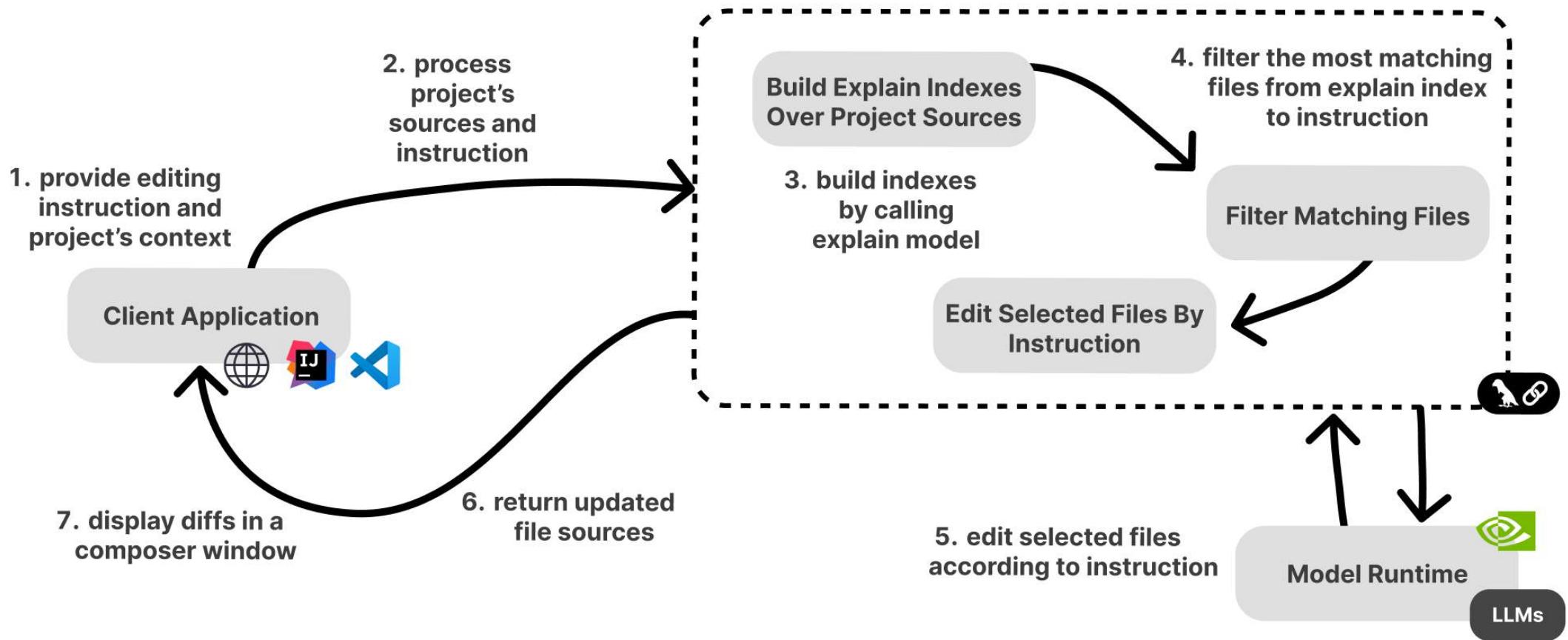
Composer (multi-file editing)

В соседних вкладках
отображаются
редактируемые файлы с
инлайн диффами с
текущим файлом

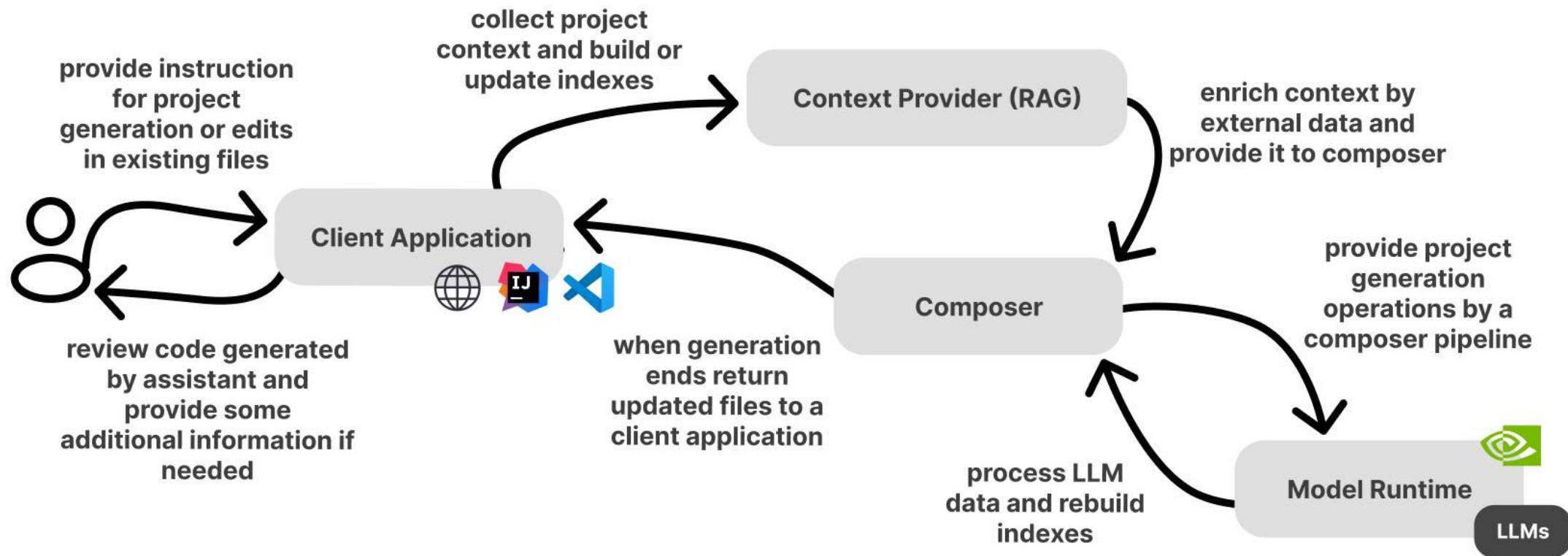


The screenshot illustrates the Composer interface for multi-file editing. It shows two tabs open: 'index.html' (active) and 'app.py'. The code editor displays the contents of 'index.html' with inline diff annotations. A search bar at the top right contains the text 'anotherword'. At the bottom, there is a command bar with buttons for 'Save', 'Apply', 'Reject', and 'Accept'. The overall layout demonstrates how multiple files can be edited simultaneously with inline diffing.

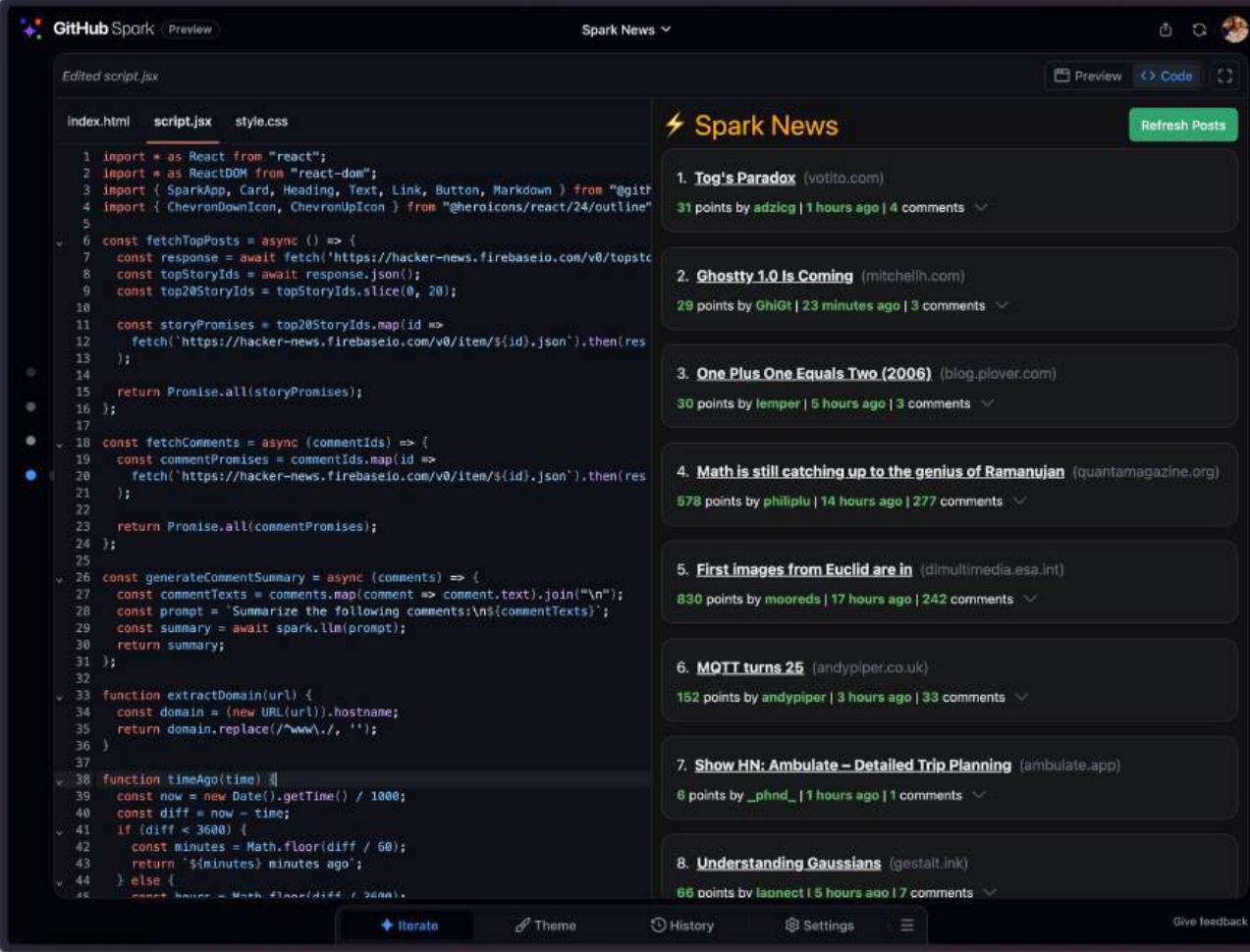
Composer (multi-file editing)



AI-driven development systems



AI-driven систем пока не так много

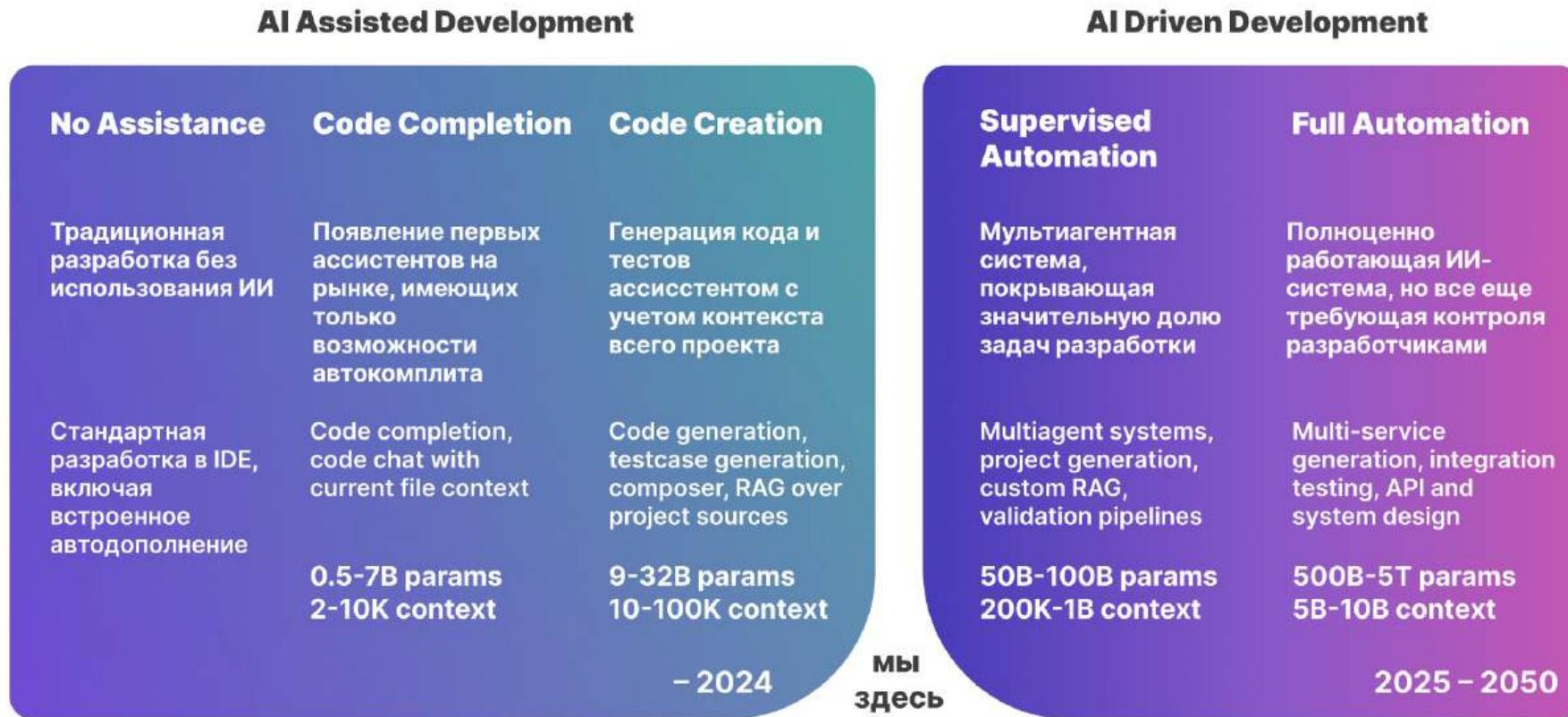


The screenshot shows the GitHub Spark interface. On the left, there's a code editor with a file named 'script.jsx' containing React code. On the right, there's a news feed titled 'Spark News' with several posts listed. A large black arrow points from the text on the right towards the GitHub Spark interface.

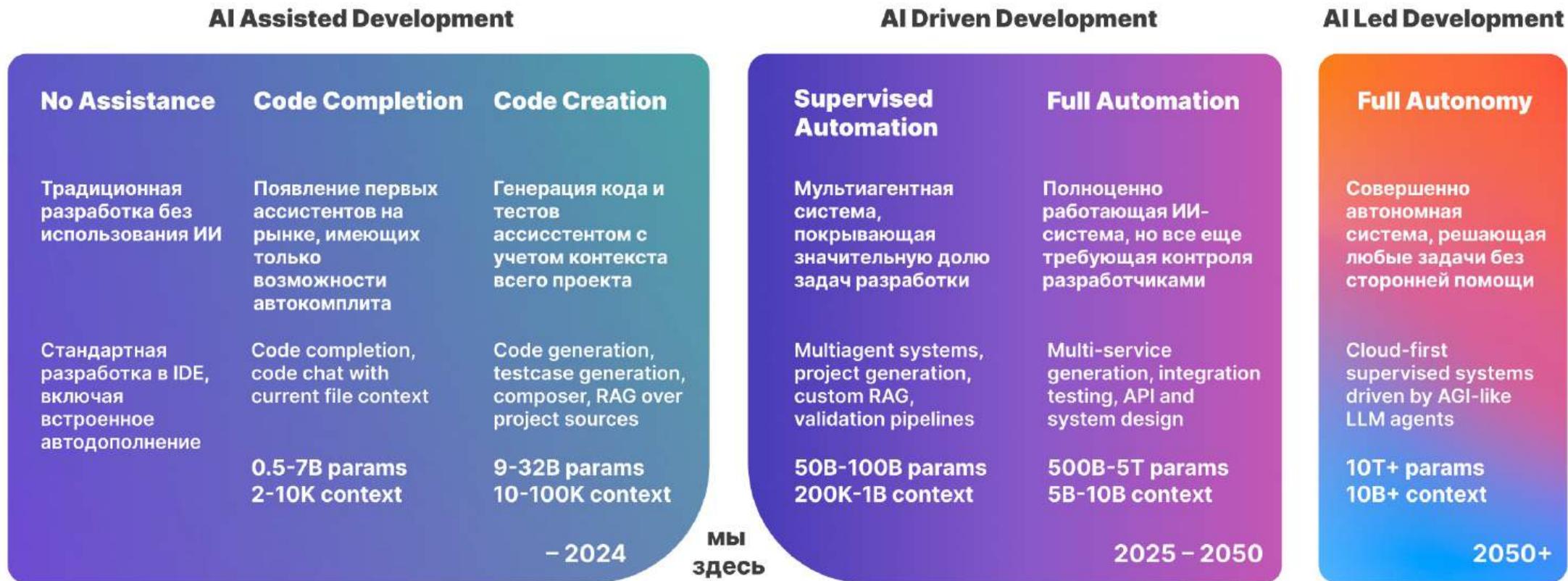
```
index.html script.jsx style.css
1 import * as React from "react";
2 import * as ReactDOM from "react-dom";
3 import { SparkApp, Card, Heading, Text, Link, Button, Markdown } from "@gitb
4 import { ChevronDownIcon, ChevronUpIcon } from "@heroicons/react/24/outline"
5
6 const fetchTopPosts = async () => {
7   const response = await fetch('https://hacker-news.firebaseio.com/v0/topst
8   const topStoryIds = await response.json();
9   const top20StoryIds = topStoryIds.slice(0, 20);
10
11   const storyPromises = top20StoryIds.map(id =>
12     fetch(`https://hacker-news.firebaseio.com/v0/item/${id}.json`).then(res
13   );
14
15   return Promise.all(storyPromises);
16 };
17
18 const fetchComments = async (commentIds) => {
19   const commentPromises = commentIds.map(id =>
20     fetch(`https://hacker-news.firebaseio.com/v0/item/${id}.json`).then(res
21   );
22
23   return Promise.all(commentPromises);
24 };
25
26 const generateCommentSummary = async (comments) => {
27   const commentTexts = comments.map(comment => comment.text).join("\n");
28   const prompt = `Summarize the following comments:\n${commentTexts}`;
29   const summary = await spark.llm(prompt);
30   return summary;
31 };
32
33 function extractDomain(url) {
34   const domain = (new URL(url)).hostname;
35   return domain.replace(/^www\./, '');
36 }
37
38 function timeAgo(time) {
39   const now = new Date().getTime() / 1000;
40   const diff = now - time;
41   if (diff < 3600) {
42     const minutes = Math.floor(diff / 60);
43     return `${minutes} minutes ago`;
44   } else {
45     const hours = Math.floor(diff / 3600);
46     const days = Math.floor(hours / 24);
47     if (days > 0) {
48       const weeks = Math.floor(days / 7);
49       if (weeks > 0) {
50         const months = Math.floor(weeks / 4);
51         if (months > 0) {
52           const years = Math.floor(months / 12);
53           if (years > 0) {
54             return `${years} years ago`;
55           } else {
56             return `${months} months ago`;
57           }
58         } else {
59           return `${weeks} weeks ago`;
60         }
61       } else {
62         return `${days} days ago`;
63       }
64     } else {
65       return `${hours} hours ago`;
66     }
67   }
68 }
```

По сути в GitHub Spark – это просто прототип с генерацией проекта и аналогом composer, добавляющий только возможность генерации проекта

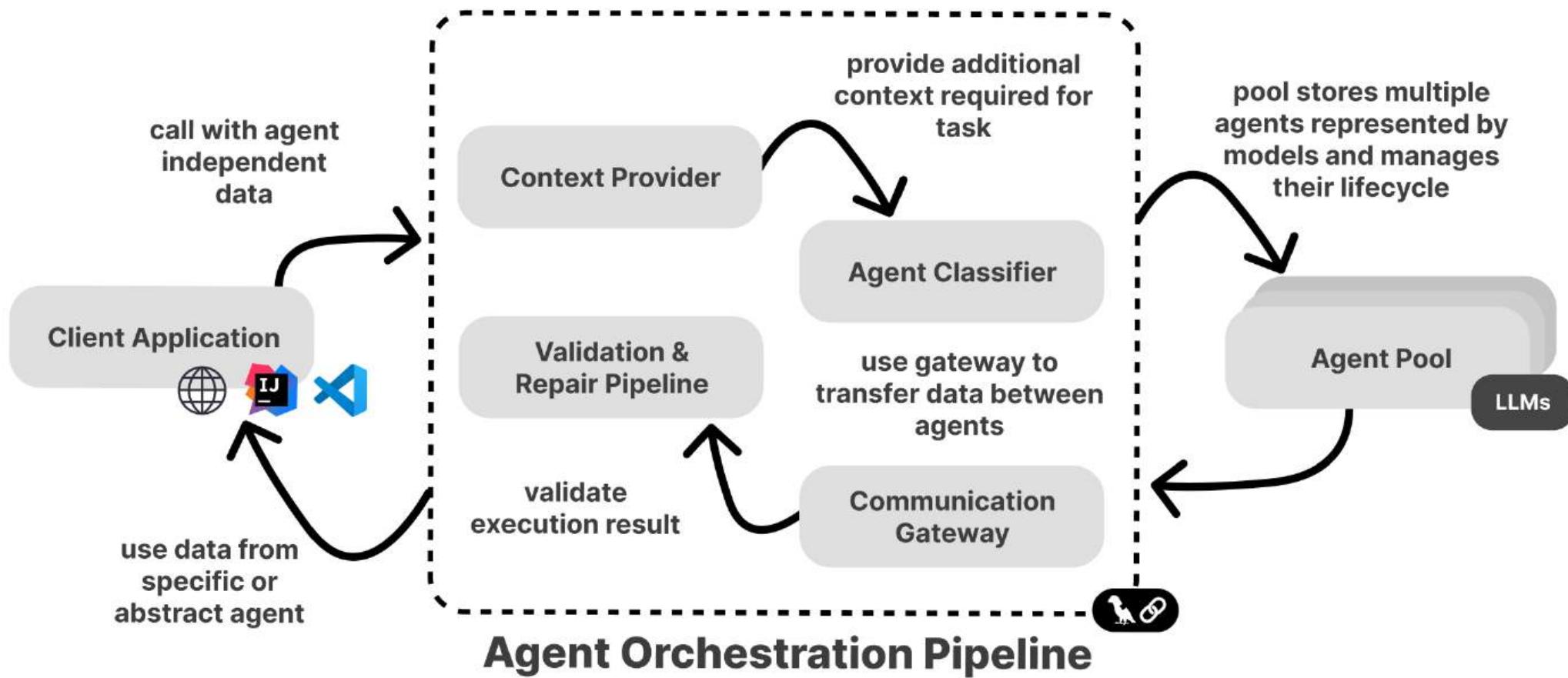
Долгосрочное развитие ИИ-ассистентов



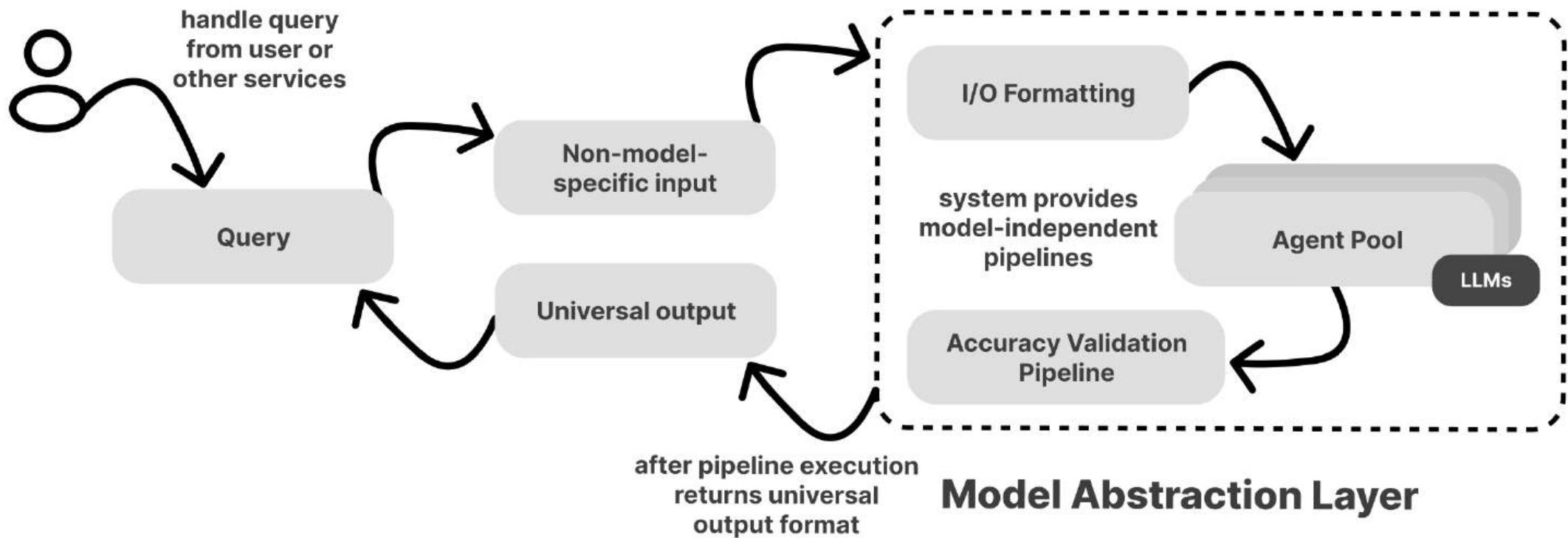
Долгосрочное развитие ИИ-ассистентов



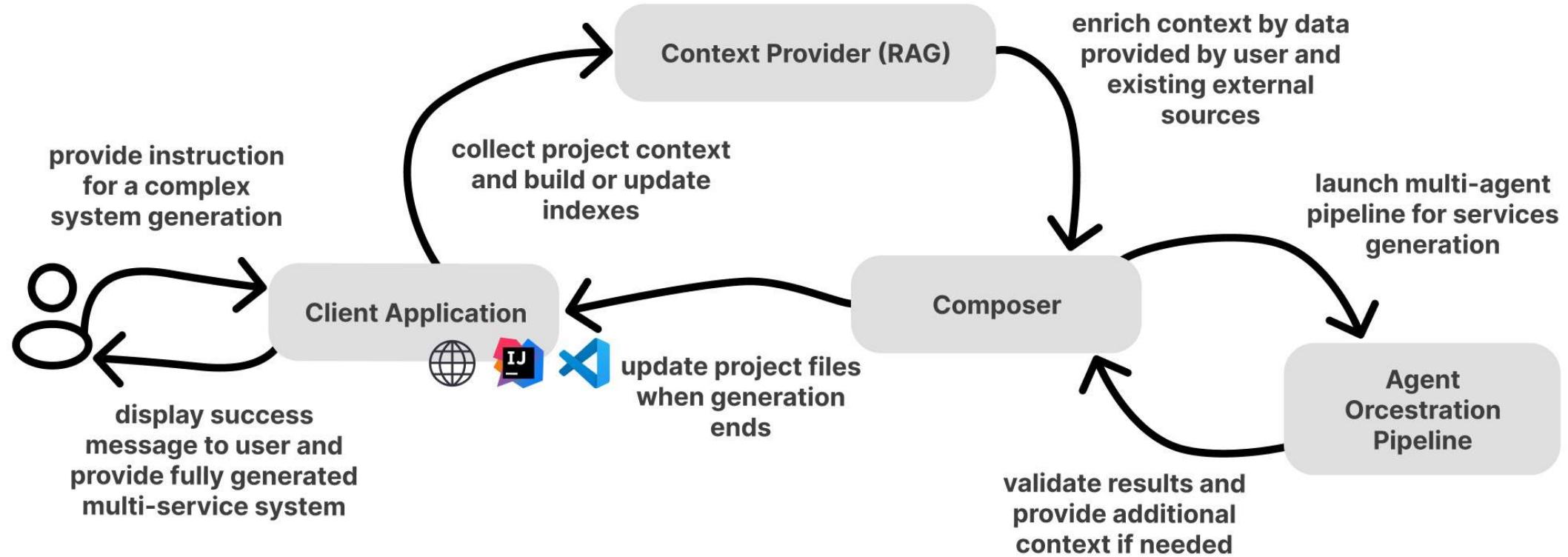
Мультиагентные системы



Model agnostic



AI-led development systems



Подведем итоги

Code Completion

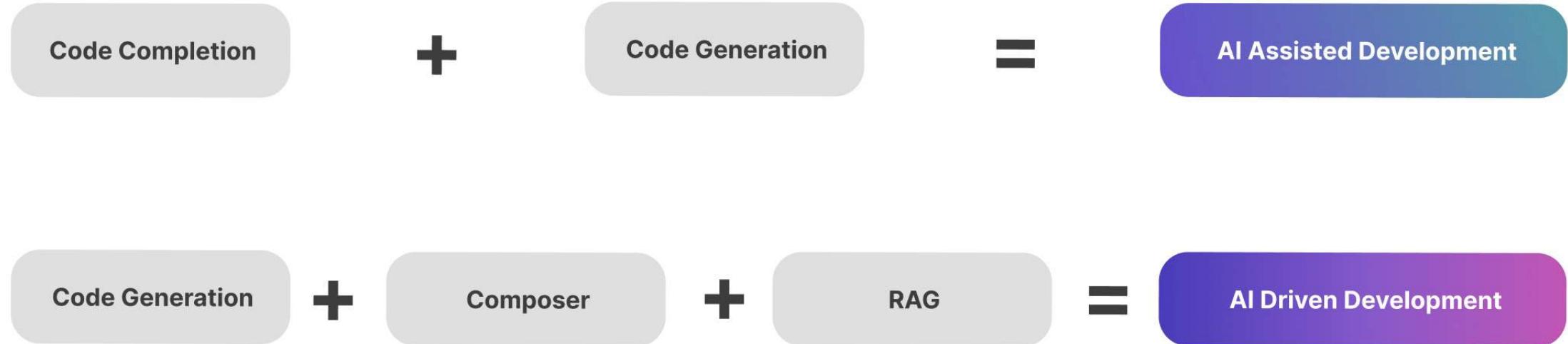


Code Generation

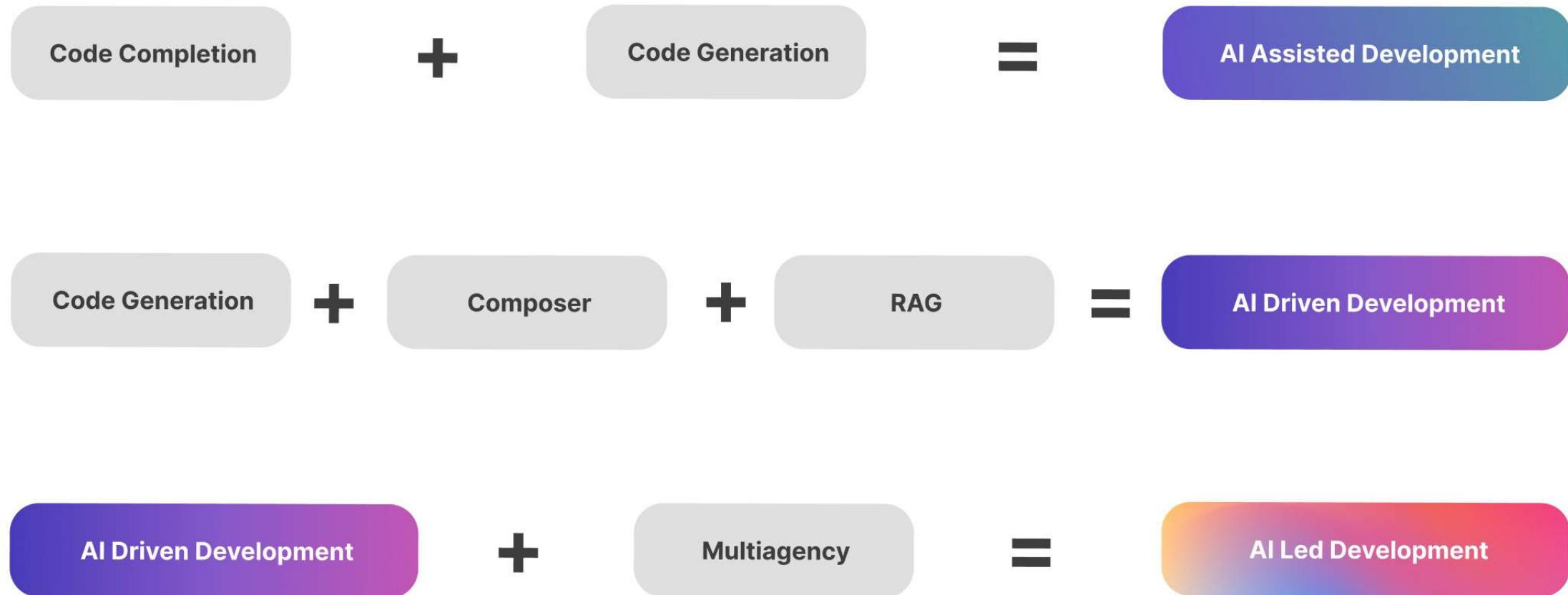


AI Assisted Development

Подведем итоги



Подведем итоги



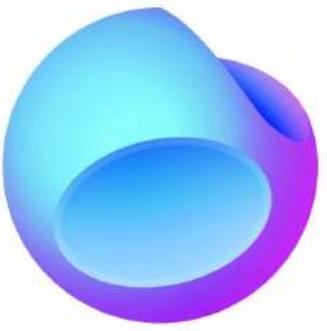
Обзор решений на рынке

	 GigaCode	 GitHub Copilot	 JetBrains AI	 Cursor
Inline Code Completion	Yes	Yes	Yes	Yes
Chat	Yes	Yes	Yes	Yes
Test Generation	Yes	Yes	No	No
Custom RAG	Beta	No	No	No
Composer	Beta	Spark (Preview)	Yes	Yes
Inline Chat	Yes	Yes	Yes	Yes

Обзор решений на рынке (часть 2)

	 Codeium	 Qodo (Codium)	 Continue	 Tabnine
Inline Code Completion	Yes	Yes	Yes	Yes
Chat	Yes	Yes	Yes	Yes
Test Generation	Yes	Yes	No	No
Custom RAG	Yes	No	Yes	Yes
Composer	No	No	No	No
Inline Chat	Yes	No	No	No

Уже сейчас есть много IDE со встроенным ИИ



The screenshot shows the GigaCode IDE interface. On the right is a code editor window titled 'JS index.js' with the following code:

```
/*
 * Добро пожаловать в GIGA IDE –
 * новый способ исследовать вселенную GitVerse!
 */

const http = require('http');
const port = process.env.PORT || 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  const msg = 'Hello GitVerse';
  res.end(msg);
});

GigaCode: explain | explain step by step | doc | test
server.listen(port, () => {
  console.log(`Server running on ${port}`);
});

GigaCode: explain | explain step by step | doc | test
server.on('error', (err) => {
  console.error(err);
  process.exit(1);
});
```

On the left is a sidebar with sections for 'CODECHAT' and 'INLINE'. The 'INLINE' section contains a message from 'GigaCode' explaining how the code works. It mentions that the code is an event handler for a Node.js request, setting a status code of 200, creating a variable 'msg' with the value 'Hello GitVerse', and sending it back to the client.

The bottom of the interface shows various status icons and a footer with file statistics: 'Ln 23, Col 1 Spaces: 2 UTF-8 LF () JavaScript'.

Почему стоит использовать ИИ в разработке?

Быстрее генерировать код



Почему стоит использовать ИИ в разработке?

Быстрее генерировать код

Снизить когнитивную нагрузку

Работать с большим количеством контекста

Reduce task switching
and maintain flow state

Improve developer
retention

Enhance developer
experience

Improve code quality
and maintainability

Additional value AI code
assistants can provide

Почему стоит использовать ИИ в разработке?

Быстрее генерировать код

Снизить когнитивную нагрузку

Работать с большим количеством контекста

Избежать появления потенциальных ошибок



Additional value AI code assistants can provide

Почему стоит использовать ИИ в разработке?

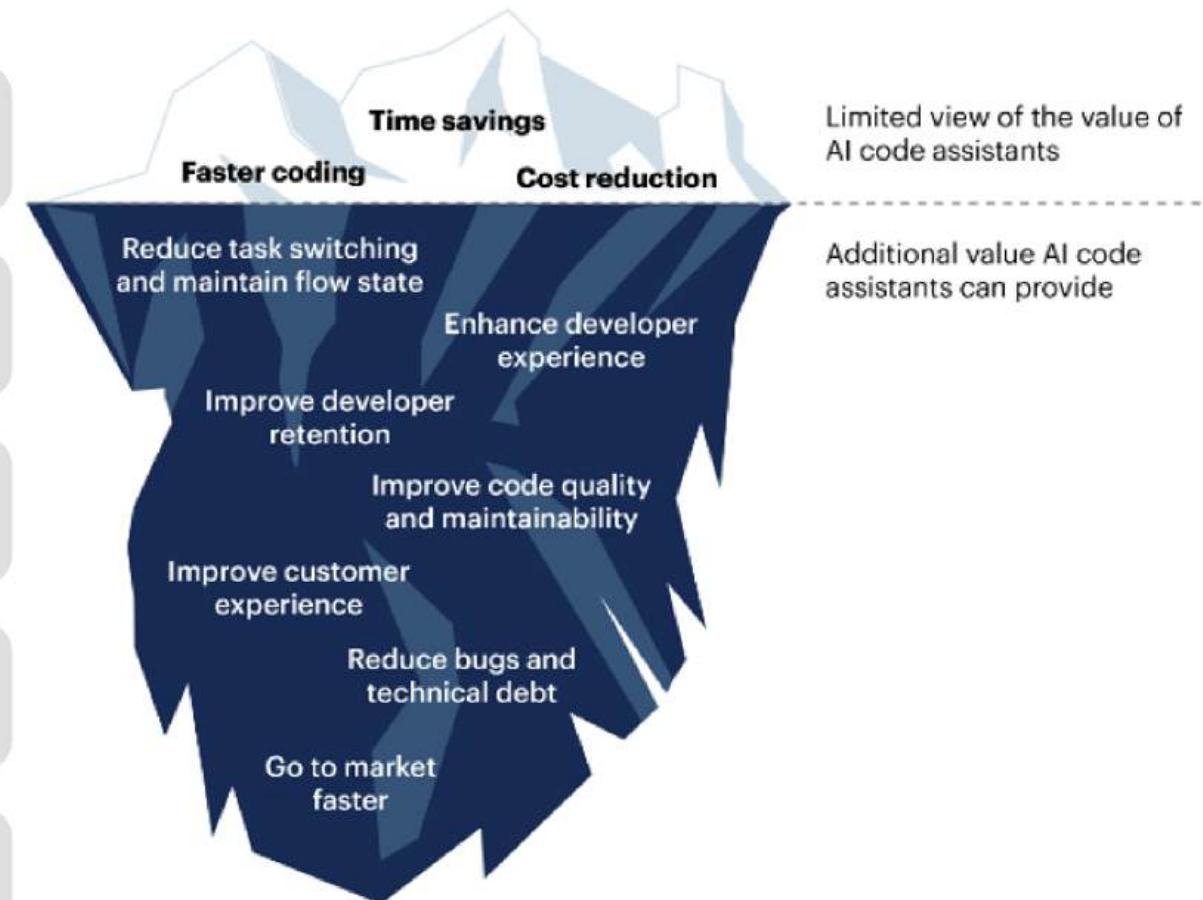
Быстрее генерировать код

Снизить когнитивную нагрузку

Работать с большим количеством контекста

Избежать появления потенциальных ошибок

Быстрее и проще изучать новые области



Презентация и полезные материалы



<https://github.com/egor-baranov/highload-2024>



Искусственный интеллект в разработке ПО

Проголосовать за мой доклад



Егор Барапнов

Solution Architect  

