
# Подсистема CRC Python IDE для анализа и ассистирования модификаций кода

*Кузнецов Илья Александрович*

Отчет по учебной практике
в форме «Производственное задание»

Научный руководитель:
профессор кафедры системного программирования, д.т.н. Д. В. Кознов
Консультант:
ведущий инженер ключевых проектов ООО «Техкомпания Хуавэй» Н. В. Тропин

Санкт-Петербургский государственный университет
Кафедра системного программирования
Группа 24.М71-ММ

# Введение

- Средства разработки напрямую влияют на качество и стоимость сопровождения ПО, а также формируют процессы разработки, поэтому рассматриваются технологическими компаниями как стратегический актив
  - В Chebyshev Research Center (CRC) была создана собственная среда разработки — CRC IDE
- Python остается одним из самых популярных языков программирования, однако его динамическая семантика существенно усложняет анализ кода и сопровождение крупных проектов
- Современные среды разработки (IDE) и ИИ-помощники (ADE) активно развивают статический анализ кода, исправления и рефакторинги, а также LLM-ассистентов
- Несмотря на зрелость существующих решений, сохраняется потребность в развитии собственных конкурентоспособных средств — таких как CRC Python IDE

# Цель и задачи

## Цель

Проектирование и реализация подсистемы в существующей среде разработки для языка Python (CRC Python IDE), предоставляющей инструменты (Features) для анализа и ассистирования модификаций кода

## Задачи

- ❏ Провести обзор инструментов для анализа и ассистирования модификаций кода в существующих средствах разработки для языка Python
- ❏ Выполнить проектирование подсистемы, определив группы инструментов (Features), предоставляемые пользователям, а также изменения существующей модели кода CRC Python IDE, необходимые для реализации этих инструментов
- ❏ Выполнить реализацию спроектированной подсистемы и интегрировать ее в CRC Python IDE
- ❏ Провести разностороннее тестирование подсистемы, модернизировав существующую тестовую инфраструктуру в CRC Python IDE
- ❏ Провести апробацию и доработку подсистемы на основе обратной связи от пользователей CRC Python IDE

# Обзор: статические анализаторы, среды разработки (IDE) и ИИ-помощники (ADE)

- Статические анализаторы — обеспечивают наиболее точный и формально обоснованный анализ кода на Python, выявляя нетривиальные ошибки и оценивая его качество и корректность; могут быть интегрированы в другие инструменты разработки
  - Mypy, Pyright
- Среды разработки (IDE) — предоставляют быстрый и точный статический анализ, формальные исправления и рефакторинги, а также ассистируют при написании кода на Python; в них часто интегрируются другие инструменты разработки
  - PyCharm, VSCode, CRC Python IDE
- ИИ-помощники (ADE) — используют LLM-ассистентов при написании кода на Python; их эффективность существенно повышается при глубокой интеграции в другие инструменты разработки через технику для получения релевантного контекста Retrieval Augmented Generation (RAG)
  - Cursor, Copilot

# Обзор: сравнение инструментов

- PyCharm, VSCode и CRC Python IDE используют собственные модели кода, качество которых можно оценить через количество разрешенных квалифицированных ссылок (QR), отражающее глубину понимания проекта и работу ключевых компонентов — PSI, Resolve и Type Inference

- На начало 2025 года CRC Python IDE демонстрирует более высокую точность при сопоставимой с PyCharm стабильности, что является базовым конкурентным преимуществом для проектируемой подсистемы

| | Pylance (PyRight) | PyCharm | SRC IDE |
|---|---|---|---|
| Flask (5072 QR) | 32.96 % | 29.20 % | 39.33 % |
| Jedi (10884 QR) | 32.77 % | 45.61 % | 60.63 % |
| PyTorch (599914 QR) | | 54.29 % | 61.36 % |
| Tensorflow (491977 QR) | | 65.96 % | 84.09 % |
| Django (202578 QR) | 28.75 % | 59.16 % | 76.08 % |
| Numpy (85004 QR) | | 61.51 % | 87.64 % |
| ToolBench (2913 QR) | 57.56 % | 62.92 % | 79.71 % |
| SciPy (123591 QR) | | 32.37 % | 85.63 % |
| Scikit-learn (90175 QR) | | 33.78 % | 69.52 % |

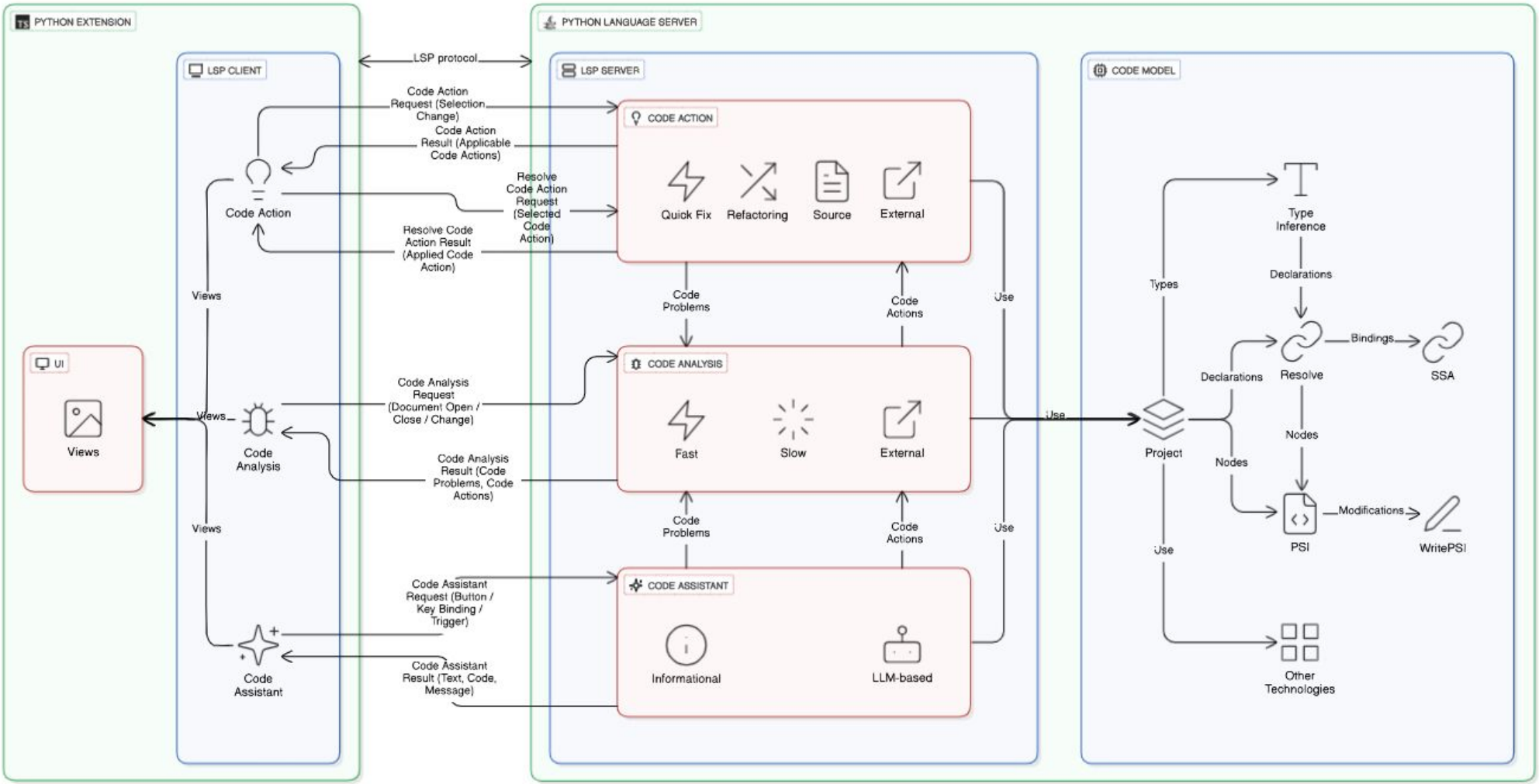
# Архитектура: инструменты подсистемы

- Инструменты для статического анализа кода
- Инструменты для исправления кода
- Инструменты для рефакторинга кода
- Информационные ассистенты
- LLM-ассистенты

# Архитектура: модель кода

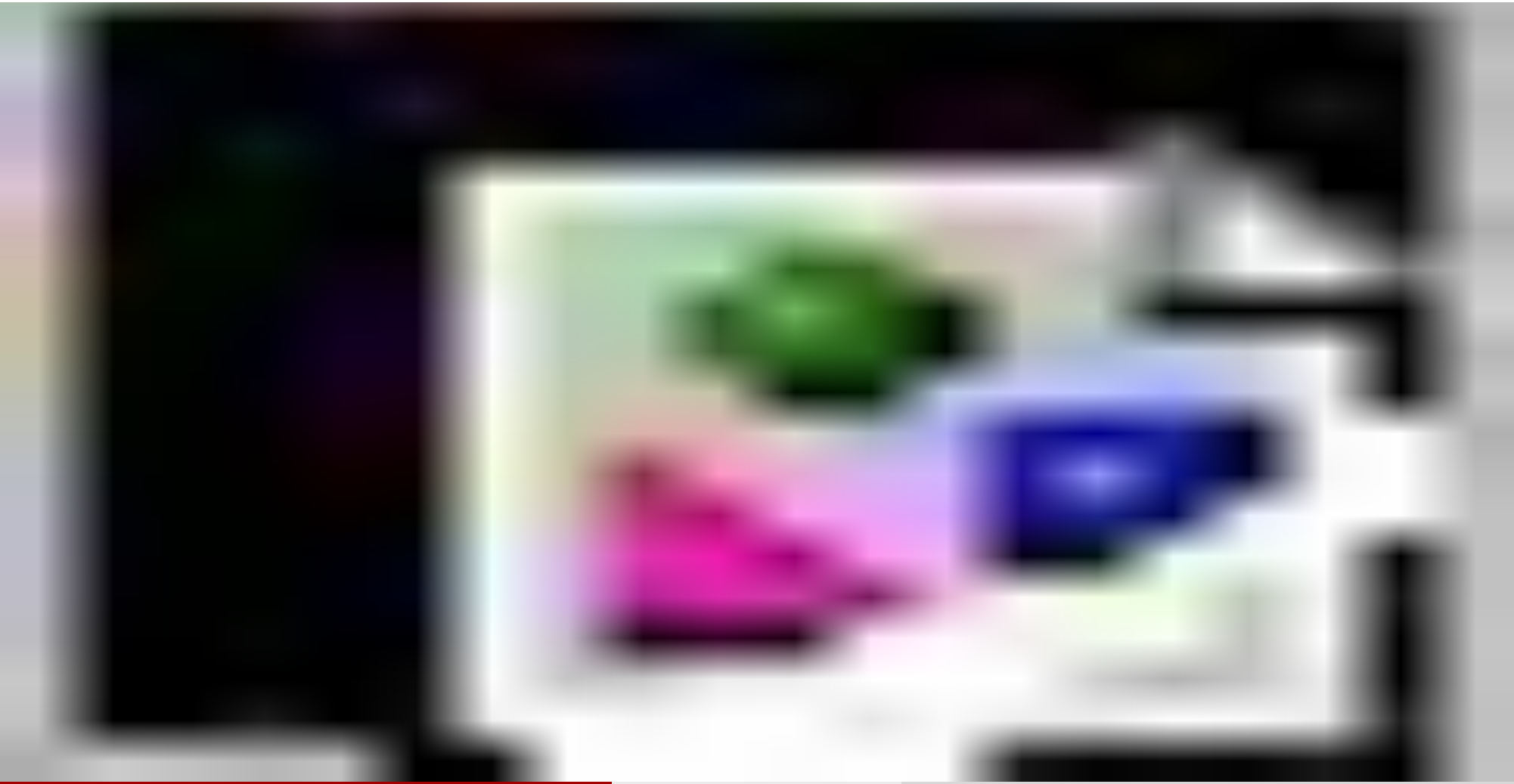
- Синтаксическое и семантическое представление программы (PSI)
- Система разрешения ссылок (Resolve)
- Система вывода типов (Type Inference)

# Архитектура: структура подсистемы



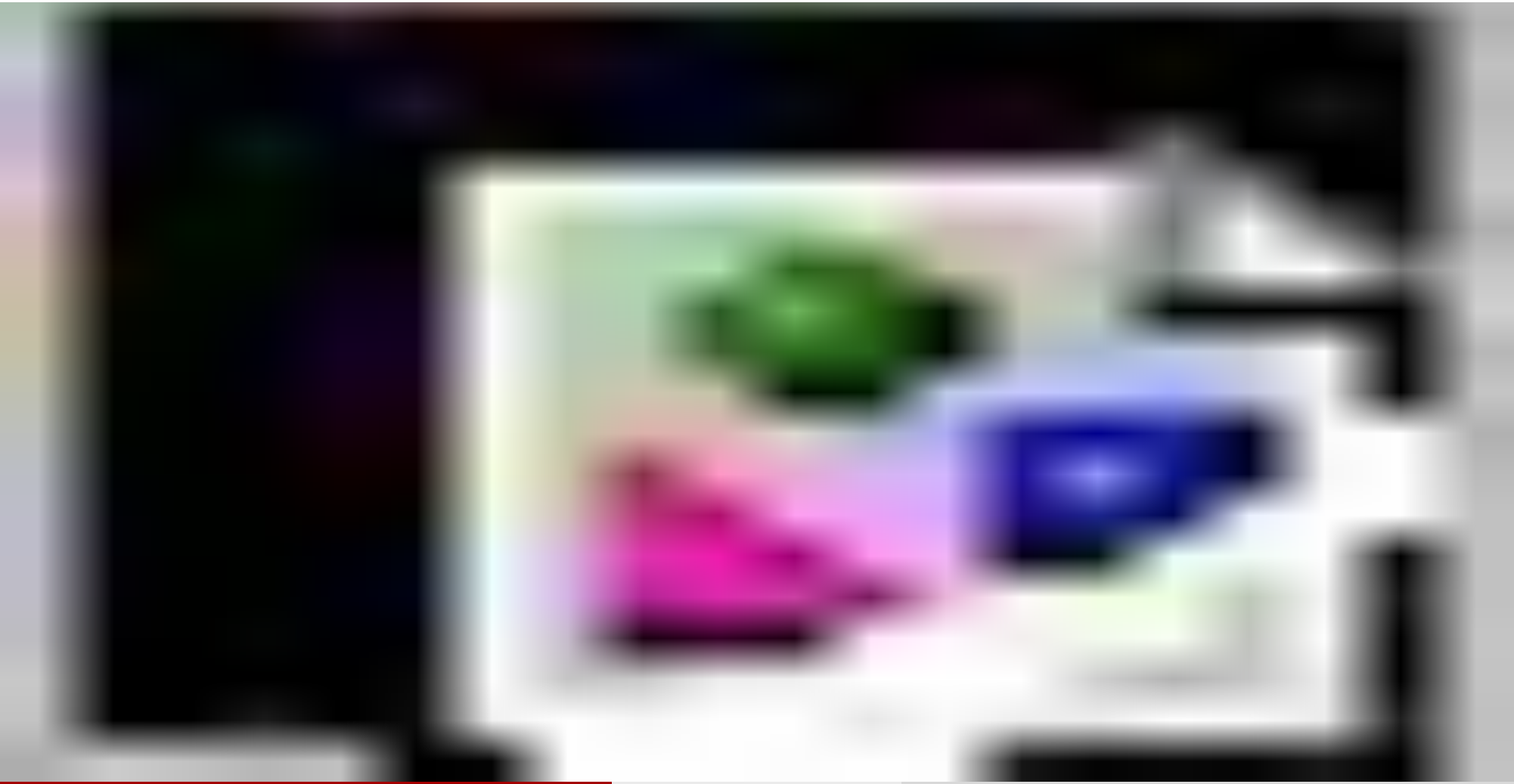
# Реализация: инструменты для статического анализа кода

- Для статического анализа кода было реализовано 40 инструментов
  - Syntax Correctness, Reference Resolve, Expected Type и др.



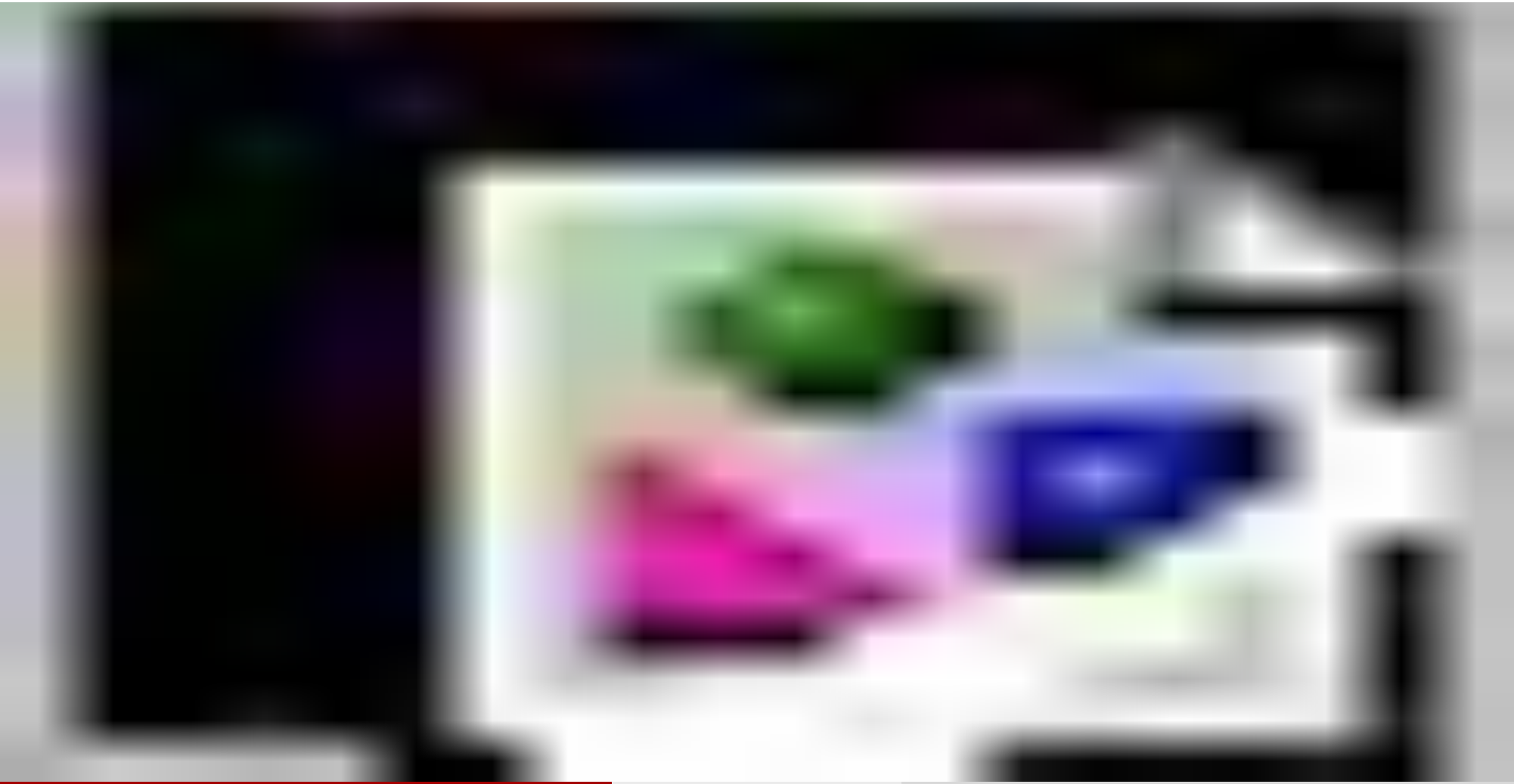
# Реализация: инструменты для исправления кода

- Для исправления кода было реализовано 18 инструментов
  - Install Package, Import Reference, Create Reference и др.



# Реализация: инструменты для рефакторинга кода

- Для рефакторинга кода было реализовано 8 инструментов
  - Extract, Inline, Rename и др.

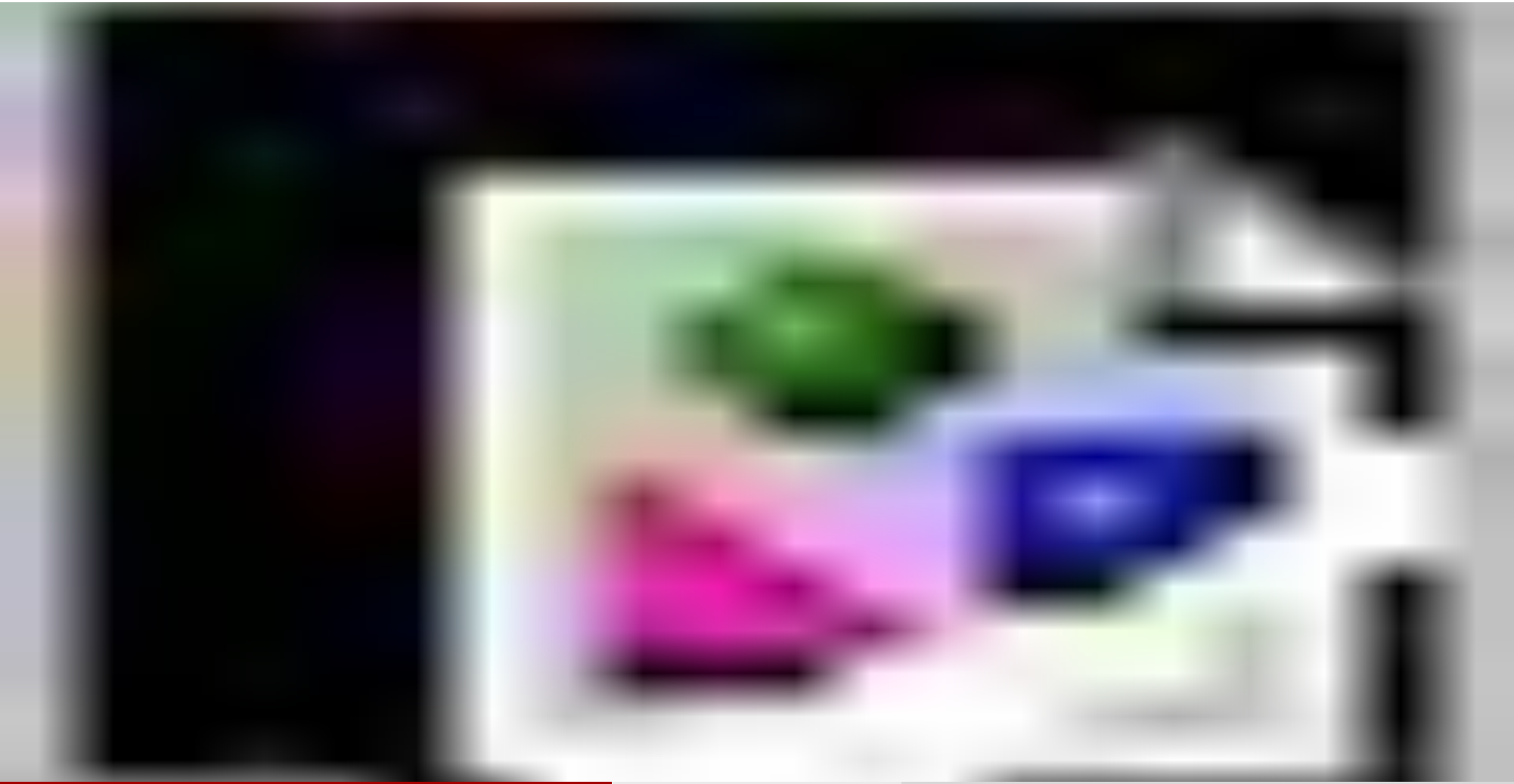


# Реализация: информационные ассистенты

- Для информационного ассистирования было реализовано 3 инструмента
  - Hover, Signature Help и Inlay Hints

# Реализация: LLM-ассистенты

- Для LLM-ассистирования было реализовано 3 инструмента
  - AI Explain Code, AI Improve Code и AI Fix Code



# Тестирование

- Инфраструктура — поддержаны конфигурируемые тестовые проекты с фиксацией ожидаемых результатов для каждой группы инструментов
- Модульное — всего было создано 3000 модульных тестов для всех групп инструментов
- Регрессионное — всего было создано 5 бенчмарков, по одному для каждой группы инструментов, и поддержан их мониторинг
- Комплексное — на уровне клиента были настроены автотесты с помощью фреймворка Selenium и применялся догфудинг (Dogfooding)

# Бенчмаркинг и мониторинг

- Для подсистемы регулярно запускаются бенчмарки, собирающие метрики, которые публикуются в VictoriaMetrics и отображаются в системе мониторинга CRC Python IDE на базе Grafana, что позволяет отслеживать производительность, улучшения и регрессии

# Апробация

- Апробация подсистемы была проведена на основе отзывов пользователей CRC Python IDE, по их запросу было реализовано 3 новых инструмента и исправлено 10 проблем
  - Version Compatibility, Incorrect Formatting и Reformat
  - Ложно-положительные/отрицательные срабатывания в анализах кода

# Заключение

Для достижения цели выпускной квалификационной работы были получены следующие результаты

- Рассмотрены соответствующие инструменты (Features) в статических анализаторах Mypy и Pyright, средах разработки PyCharm, VSCode и CRC Python IDE, а также в ИИ-помощниках Cursor и Copilot
- Спроектирована подсистема среды разработки, для реализации выбрано 5 групп инструментов (Features): для статического анализа, исправления и рефакторинга кода, а также информационные и LLM-ассистенты
- Выполнена реализация спроектированной подсистемы: 40 инструментов для статического анализа, 18 инструментов для исправления кода и 8 инструментов для рефакторинга кода, а также 3 информационных ассистента и 3 LLM-ассистента
- Выполнена модернизация тестовой инфраструктуры, создано 3000 модульных тестов и 5 бенчмарков для регрессионного тестирования подсистемы, в качестве комплексного тестирования были настроены автотесты с помощью фреймворка Selenium и применялся догфудинг (Dogfooding)
- Проведена апробация подсистемы на основе отзывов пользователей CRC Python IDE, что способствовало созданию 3 новых инструментов и исправлению 10 проблем