Лабораторная работа №3

По дисциплине: *«Инструментальные средства разработки ПО»*

Выполнил: студент группы M3100

Коткин Михаил Яковлевич

Номер ИСУ: 501585

Санкт-Петербург

2025

Введение

Данная лабораторная работа посвящена разработке плагина для интегрированной среды разработки Visual Studio Code. Целью работы является изучение архитектуры IDE и механизмов расширения её функциональности через создание собственного плагина.

Техническое задание

Цель проекта

Разработка плагина Text Converter для быстрого преобразования текста между различными стилями написания в среде Visual Studio Code.

Функциональные требования

- Преобразование текста в UPPERCASE
- Преобразование текста в lowercase
- Преобразование текста в camelCase
- Преобразование текста в snake_case
- Преобразование текста в kebab-case
- Преобразование текста в Title Case
- Интеграция с Command Palette VS Code
- Контекстное меню для быстрого доступа

Нефункциональные требования

- Простота использования
- Быстродействие преобразований
- Совместимость с VS Code 1.90.0+
- Минимальное потребление ресурсов

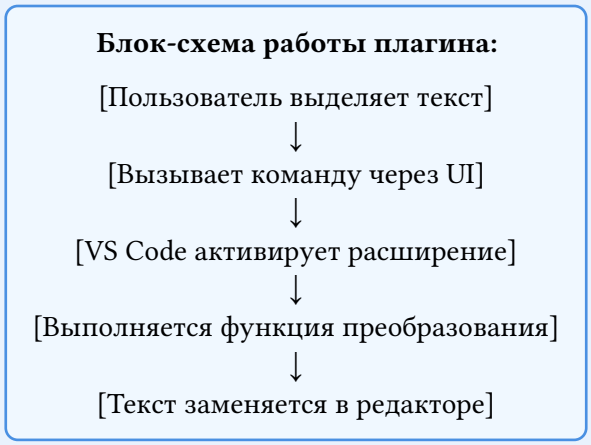
Архитектура плагина

Общая архитектура IDE VS Code

Visual Studio Code построена на архитектуре, состоящей из:

- **Основного процесса** - отвечает за UI и управление окнами
- **Процесса расширений** - изолированная среда для выполнения плагинов
- **API для расширений** - набор интерфейсов для взаимодействия с IDE

Архитектура плагина Text Converter



Компонентная структура

Основные компоненты:

- **Extension Activation** - инициализация плагина
- **Command Registry** - регистрация команд в VS Code
- **Text Processing Engine** - движок преобразования текста
- **VS Code API Integration** - интеграция с API среды

Реализация

Технологический стек

- Язык программирования: TypeScript 5.3.0
- Платформа: Node.js (через VS Code runtime)
- API: VS Code Extension API 1.90.0
- Система сборки: npm + TypeScript Compiler

Ключевые модули

extension.ts - основной модуль плагина:

```
// Активация расширения
export function activate(context: vscode.ExtensionContext) {
  // Регистрация команд преобразования
  const converters = { toUpperCase, toLowerCase, ... };

  // Динамическая регистрация команд
  for (const [cmd, fn] of Object.entries(converters)) {
    const disposable = vscode.commands.registerCommand(
      `text-converter.${cmd}`,
      () => handleConversion(fn)
    );
    context.subscriptions.push(disposable);
  }
}
```

Алгоритмы преобразования текста

camelCase преобразование:

```
text.toLowerCase().replace(/[-_\s]+(.)?/g, (_, c) =>
  c ? c.toUpperCase() : ""
)
```

snake_case преобразование:

```
text.replace(/([a-z])([A-Z])/g, "$1_$2")
    .replace(/\s+/g, "_")
    .toLowerCase()
```

Документирование

Используемые инструменты

TypeDoc - генератор документации из комментариев

Markdown - для пользовательской документации

Структура документации

docs/
├── index.html # Главная страница
├── modules.html # Модули
└── globals.html # Глобальные функции

Пример документирования кода

typescript function toCamelCase(text: string): string { }

Тестирование и отладка

Методы тестирования

Функциональное тестирование команд

Тестирование алгоритмов преобразования

Интеграционное тестирование с VS Code

Пользовательское тестирование

Инструменты отладки

Встроенный отладчик VS Code

Extension Development Host

Console logging для диагностики