Vim w OCamlu

Modalny edytor tekstowy w funkcyjnym wydaniu.

Filip Figzał

2026-01-15

Outline

Założenia projektu

Cel

Stworzenie ~~uproszczonego~~ **niezawodnego** edytora modalnego, który prezentuje siłę programowania funkcyjnego.

Dlaczego?

Cel

Stworzenie ~~uproszczonego~~ **niezawodnego** edytora modalnego, który prezentuje siłę programowania funkcyjnego.

Dlaczego?

- Większość edytorów to “spaghetti” mutowalnego stanu.

Cel

Stworzenie ~~uproszczonego~~ **niezawodnego** edytora modalnego, który prezentuje siłę programowania funkcyjnego.

Dlaczego?

- Większość edytorów to “spaghetti” mutowalnego stanu.
- Ciekawe wyzwanie.

Podstawowe funkcje

- Wyświetlanie pełnego stanu w terminalu.

Podstawowe funkcje

- Wyświetlanie pełnego stanu w terminalu.
- Tryby edytora (normalny, insert, command) modelowane jako **ADT**.

Podstawowe funkcje

- Wyświetlanie pełnego stanu w terminalu.
- Tryby edytora (normalny, insert, command) modelowane jako **ADT**.
- Drzewiasta historia zmian.

Podstawowe funkcje

- Wyświetlanie pełnego stanu w terminalu.
- Tryby edytora (normalny, insert, command) modelowane jako **ADT**.
- Drzewiasta historia zmian.
- Operacje I/O na plikach (otwieranie/zapisywanie).

Podstawowe funkcje

- Wyświetlanie pełnego stanu w terminalu.
- Tryby edytora (normalny, insert, command) modelowane jako **ADT**.
- Drzewiasta historia zmian.
- Operacje I/O na plikach (otwieranie/zapisywanie).
- Nawigacja po tekście:
  - h/j/k/l (podstawowe poruszanie się);
  - w/b/e (nawigacja po słowach).

Architektura

Model Jedno źródło prawdy.

Zarys

Model Jedno źródło prawdy.

View Deklaratywne renderowanie stanu do bufora terminala.

Zarys

Model Jedno źródło prawdy.

View Deklaratywne renderowanie stanu do bufora terminala.

Update Czysta funkcja `State -> Event -> State`. Łatwa do testowania.

Struktury Danych

Zarządzanie stanem

- Algebraiczny Typ Danych dla trybów.
- Zapobiega niepoprawnym stanom.

Struktury Danych

Zarządzanie stanem

- Algebraiczny Typ Danych dla trybów.
- Zapobiega niepoprawnym stanom.

Reprezentacja tekstu

- **Zipper** zamiast zwykłej tablicy.
- **Zipper of Zippers**: optymalizacja dle wielu linii.

Technologie

- Biblioteka `Notty` → deklaratywne renderowanie w terminalu.

Technologie

- Biblioteka `Notty` ⟶ deklaratywne renderowanie w terminalu.
- Moduł `Unix`:
  - I/O na plikach
  - obsługa naciśnięć w terminalu

Technologie

- Biblioteka `Notty` ⟶ deklaratywne renderowanie w terminalu.
- Moduł `Unix`:
  - I/O na plikach
  - obsługa naciśnięć w terminalu
- Wkład własny:
  - Logika edytora
  - Algorytmy nawigacji
  - Zarządzanie buforem i stanem

Ryzyka

Wyzwania i Rozwiązania

Wyzwanie 1: Wydajność

- **Problem:** Jak edytować 10k linii tekstu bez mutowania pamięci (kopiowania tablic)?
- **Rozwiązanie:** Struktury trwałe (Persistent Data Structures).
- **Implementacja:** Zipper (suwak) - lokalne zmiany są O(1), a historia zmian jest “darmowa”.

Wyzwania i Rozwiązania

Wyzwanie 2: Zarządzanie stanem

- **Problem:** “Vim hell” - czy jestem w trybie Insert? Czy wpisuję komendę?
- **Rozwiązanie:** Algebraiczne Typy Danych (ADT).
- **Efekt:** Niepoprawne stany są niemożliwe do reprezentowania w kodzie.

Pytania

Koniec

Dziękuję za uwagę.