ДИЗАЙН ПАТЕРНИ

ПІДГОТУВАЛА ЛОБУР ЮЛІЯ ПП-32

ОПИС ПРОГАМИ

Розроблений веб-додаток на Angular отримує цитати програмістів з зовнішнього API, відображає їх у списку та дозволяє користувачам додавати їх до списку "вибраних". При кліку на цитату відкривається модальне вікно з детальною інформацією. У програмі використовуються дизайн патерни, такі як Фабрика для створення об'єктів цитат, Синглтон для управління сервісами, та Спостерігач для реалізації оновлень при зміні стану додатка. Це забезпечує зручний інтерфейс та ефективну організацію коду.

ВИКОРИСТАНІ ДИЗАЙН ПАТЕРНИ

01 FACTORY PATTERN

02 SINGLETON PATTERN

03 OBSERVER PATTERN

FACTORY PATTERN

Патерн "Фабрика" використовується для створення об'єктів без прив'язки до конкретного класу. Він дозволяє делегувати процес створення об'єктів до окремого класу, який відповідає за їх ініціалізацію.

```
import { Quote } from '../models/quote.model';

export class QuoteFactory {
    public static createQuote(data: any): Quote {
        return {
            id: data.id,
            author: data.author,
            quote: data.quote
        };
    }
}
```

- Спрощує створення об’єктів
- Знижує залежність від конкретних класів
- Покращує підтримку та розширюваність

SINGLETON PATTERN

Патерн "Синглтон" забезпечує наявність лише одного екземпляра класу в усьому додатку.

```
// створюю Singleton сервіс
private constructor() { }

public static getInstance(): FavoritesService {
  if (!FavoritesService.instance) {
    FavoritesService.instance = new FavoritesService();
  }
  return FavoritesService.instance;
}
```

- Гарантує єдине джерело для певних об’єктів
- Полегшує управління ресурсами
- Запобігає створенню зайвих об’єктів

OBSERVER PATTERN

Патерн "Спостерігач" дозволяє об'єктам спостерігати за змінами в іншому об'єкті, отримуючи оновлення, коли стан спостережуваного об'єкта змінюється.

- Зменшує зв’язок між компонентами
- Дозволяє гнучко реагувати на зміни стану
- Використовується для реалізації подій та оновлення інтерфейсів

```
@Injectable({
  providedIn: 'root',
})
export class FavoritesService {
  private static instance: FavoritesService;
  private favoriteQuotes: Quote[] = [];
  private favoritesSubject = new BehaviorSubject<Quote[]>([]);

  private constructor() { }

  public static getInstance(): FavoritesService {
    if (!FavoritesService.instance) {
      FavoritesService.instance = new FavoritesService();
    }
    return FavoritesService.instance;
  }

  // дизайн патерн!!! - Observer
  // додаю цитату у вибране
  addFavorite(quote: Quote): void {
    if (!this.favoriteQuotes.find((q) => q.id === quote.id)) {
      this.favoriteQuotes.push(quote);
      this.favoritesSubject.next(this.favoriteQuotes);
    }
  }

  // видаляю цитату з вибраного
  removeFavorite(quote: Quote): void {
    this.favoriteQuotes = this.favoriteQuotes.filter((q) => q.id !== quote.id);
    this.favoritesSubject.next(this.favoriteQuotes);
  }
```

ДЯКУЮ ЗА УВАГУ