Глава 1. Увод

Добре дошли в документацията за "*OOP Social Network*". Този документ предоставя преглед на основната идея на приложението, подробности за избраната архитектурата, използваните класове, както и техните функционалности.

1.1. Описание и идея на проекта

Проектът представлява конзолно приложение, което зарежда данни за дадена социална мрежа или създава такива, ако не съществуват, и поддържа множество команди, които потребителят може да въведе.

Идеята на приложението е да наподобява социалната мрежа "*Reddit*", където потребители могат да създават теми, дискусии, да коментират, да оценяват коментари и да отговарят на тях, като това се отразява на техните потребителски точки.

Всичко това се случва под парадигмата на обектно-ориентираното програмиране, като грешки от различен характер биват обработвани с цел запазване на данните цели.

1.2 Цел и задачи на разработката

Основната цел на разработката е приложението да следва стриктно основите на обектно-ориентираното програмиране, използвайки в своето устройство общоприетите добри практики при изработката на проект, базиран на тази парадигма.

Енкапсулацията е внедрена като основен принцип за организация на кода. Всички "глобални" статични променливи и обекти са стриктно енкапсулирани в подходящи класове, като членовете данни, които са от решаващо значение за коректното функциониране на класовете, са дефинирани като “private” или “protected”. Това не само че осигурява по-добра защита на данните, но и предотвратява директния достъп до тях от външния свят.

Абстракцията е успешно приложена, позволявайки на потребителите да използват интерфейсите на всички класове без да се нуждаят от подробности за тяхната вътрешна реализация. Това значително опростява използването на приложението и позволява лесна поддръжка и разширяване на функционалността му.

Полиморфизмът и наследяването не са използват в имплементацията на това приложение, тъй като бизнес класовете “*User*”, “*Topic*”, “*Post*”, “*Comment*”, “*Reply*”, “*Reaction*” нямат смислово приложение на двата принципа. Нито един от класовете не притежава “*Is a*” смислова връзка, а използва “*Has a*”. Например “*Topic*” съдържа множество обекти от “*Post*”, а “*Post*” съдържа “*Comment*” и т.н. Подвеждащо е, че “Reply” всъщност би могъл да се счита за “*Comment*” и да бъде наследяващ клас, но ако това е така, то се създава безкрайно създаване на инстанции, тъй като става кръговидна йерархия от обекти, без дъно.

1.3. Структура на документацията

Глава 1. Увод

1.1. Описание и идея на проекта

1.2. Цел и задачи на разработката

1.3. Структура на документацията

Глава 2. Проектиране

2.1. Обща архитектура

2.1.1. Персонализирани контейнери

2.1.2. Бизнес класове

2.1.3. Сервизни класове

2.1.4. Други класове

Глава 3. Заключение

3.1. Обобщение на изпълнението на началните цели.

3.2. Насоки за бъдещо развитие и усъвършенстване.

Използвана литература

"Boyer Moore Algorithm for Pattern Searching." GeeksforGeeks, GeeksforGeeks, 11 Mar. 2024, www.geeksforgeeks.org/boyer-moore-algorithm-for-pattern-searching/.

Глава 2. Проектиране

2.1. Обща архитектура

2.1.1 Персонализирани контейнери (*Custom container classes*)

String: Представлява управление на символни низове с динамична памет в C++. Съдържа конструктори, деструктор, оператори за присвояване и сравнение, както и методи за работа със символни низове като търсене на поднизове, преобразуване в число и проверка за цифрови символи. За метода “find(const char* substr)” е използван алгоритъмът на Бойер-Мур за търсене на подтекст в низ.

Vector: шаблонен клас, който реализира динамичен масив от обекти от различен тип. Предлага основни операции като добавяне на елементи, промяна на размера на масива, изтриване на елементи, достъп до елементи чрез оператори за индексиране и други.

2.1.2 Бизнес класове (*Business classes*)

Потребител (*User*): Основният участник в системата, който има уникален идентификатор, потребителско име, парола, име, фамилия, точки и права за достъп.

Тема (*Topic*): Основната рамка, която групира постове, коментари и отговори, свързани с определена тематика. Съдържа заглавие, описание, идентификационен номер, идентификационен номер на нейният създател, както и постовете в нея.

Пост (*Post*): Съдържание, споделяно в рамките на дадена тема, състоящо се от заглавие, съдържание, идентификационен номер, идентификационен номер на създателя и, идентификационен номер на темата, към която принадлежи, както и всички коментари, публикувани в поста.

Коментар (*Comment*): Текстов отзив спрямо определен пост, съдържа името на автора си, текст, точки, идентификационен номер, идентификационен номер на поста, към който принадлежи, идентификационен номер на автора си, както и реакциите и отговорите, които са отправени към него.

Отговор (*Reply*): Текстов отзив, предназначен за отговор на конкретен коментар. Съдържа името на автора си, текст, точки, идентификационен номер, идентификационен номер на поста, към който принадлежи, както и идентификационен номер на коментара, на който отговаря.

Реакция (*Reaction*): Интеракция с пост или коментар. Съдържа вида на реакцията - “*upvote*”, “*downvote*” или “*undefined”*, идентификационен номер на потребителя, поставил реакцията, както и позицията си във вектора от реакции, намиращ се в “*Reply*” класа.

2.1.3 Сервизни класове (*Services*)

FileHandler: Управлява всички операции, свързани с файловата система. Този клас се грижи за зареждането на данни от файлове при стартиране на приложението, както и за записването на текущото състояние на данните обратно във файловете при затваряне на приложението или при команда за запазване.

ConsoleInputGetter: Отговаря за получаването на данни от потребителя чрез конзолата. Този клас се грижи за правилното прочитане и интерпретиране на въведените данни, които се предават на *InputValidator* за валидиране.

InputValidator: Валидира въведените данни от потребителя, за да се гарантира, че командите и данните са правилно форматирани, няма символи извън ASCII таблицата, както и че няма да има допуснато препълване на буфера.

PrintHandler: Отговаря за извеждането на информация на конзолата. Този клас форматира и представя данните по подходящ начин, за да осигури ясна и разбираема визуализация на резултатите от изпълнените команди, като е взет предвид проблемът при извеждане на "*nullptr*" и е решен чрез метода "safePrint(const char* label, const String& value);"

ObjectFactory: Отговаря за създаването на нови обекти. Този клас предоставя методи за създаване на потребители, теми, постове, коментари, отговори и реакции, като гарантира, че всички обекти са правилно инициализирани и подготвени за използване, като също така пази идентификационните номера за последно създадените обекти от клас "User" и "Topic", като те изискват тези данни, за да се зададат техните идентификационни номера коректно, с едно повече от предишният създаден обект.

CommandsHandler: Този клас управлява изпълнението на команди, въведени от потребителя. Поддържа списък с команди и техните описания, които потребителят може да използва. Проверява дали е възможно дадена команда да бъде изпълнена и пренасочва изпълнението на програмата на мястото, където се съдържа логиката за дадената команда. При евентуално хвърлено изключение от някоя от тях, го прихваща и обработва, освен в случая на "*bad_alloc*", което е специфично и бива обработено в "*Application"* класа.

2.1.4 Други класове

SocialNetwork: Този клас управлява основните операции и съдържа логиката за повечето команди за социалната мрежа. Той съдържа данни за текущите потребители, теми и постове, както и методи за взаимодействие с тях. Основните му функции включват регистрация, влизане и излизане от системата, създаване и управление на теми и постове, както и коментиране и гласуване на коментари.

CurrentData: Този клас се грижи за енкапсулиране на инстанцията на заредената социална мрежа. Освен това държи флаг, указващ дали са направени промени. Предоставя методи за получаване и задаване на текущата мрежа и състоянието на промените.

Configuration: Този клас дефинира константите, които контролират максималните дължини на различни символни низове. Включени са максималните дължини за имена на потребители, пароли, заглавия на теми и постове, текстове и други. Също така, класът съдържа методи за изчисляване на максималния брой цифри, които могат да бъдат въведени за входящи данни. Тези константни членове служат за предотвратяване на препълване на буфера и се използват в методите на "*ConsoleInputGetter*" класа.

Application: Този клас представлява централната част на приложението. Използва се за управление на основния цикъл на програмата, който обработва командите на потребителя чрез конзолния интерфейс и задейства тяхното изпълнение. Също така в него се прихваща критичното изключение "*bad_alloc*", при чийто случай се задейства извънредно запазване на данните в отделен файл с цел максимално запазване.

Глава 3. Заключение

3.1. Обобщение на изпълнението на началните цели.

Проектът "*OOP Social Network*" е успешно реализиран като конзолно приложение, което моделира функционалността на социална мрежа подобна на "Reddit". Основната цел на проекта беше да се следват основните принципи на обектно-ориентираното програмиране, като енкапсулация, абстракция и принципите за управление на данни с динамична дължина, както и добрите практики за писане на обектно-ориентиран код.

3.2. Насоки за бъдещо развитие и усъвършенстване.

Проектът "OOP Social Network" има потенциал за значително усъвършенстване чрез въвеждане на семантика на преместване за по-ефективно управление на ресурсите. Допълнително разширение на функционалността може да включва изтриване на потребителски профили, имплементиране на чат функция, монетизиране на платформата, добавяне на визуален интерфейс, интегриране на база данни за по-добро управление на данните и подобряване на сигурността чрез допълнителни мерки за защита на личната информация на потребителите.