УчИМИ Ученически институт по математика и информатика

ДВАДЕСЕТ И ПЕТА УЧЕНИЧЕСКА СЕКЦИЯ
на Пролетната конференция на СМБ
УС'25
31 март – 3 април 2025 г., Варна

ТЕМА НА ПРОЕКТА

"RosePad" - мултиплатформен редактор

(https://github.com/TMG8047KG/RosePad)

Автор:

Александър Ганчев Горанов
ПМГ "Акад. Иван Гюзелев" - Габрово
12 клас

Научен ръководител (консултант):

Галя Неделчева, старши учител,
ПМГ "Акад. Иван Гюзелев" - Габрово

РЕЗЮМЕ

RosePad е лек и интуитивен текстови редактор, в който са реализирани най-често използваните функционалности за обработка на текстови файлове. Основни характеристики на продукта са неговата мултиплатформеност и разпространението му под свободен лиценз GPL и FOSS (Free and Open Source Software). Разработката е базирана на framework Tauri, а за frontend и backend-модулите основно са използвани framework React, езиците TypeScript и Rust. Избраните програмни средства и технологии имат висока степен на интегрираност помежду си, избягват се колизии между елементи и модули и се гарантира оптимално ниво на защитеност и сигурност.

Тези характеристики и възможности на RosePad обуславят приложимост и разпространение в съвременния мултиплатформен свят.

SUMMARY

RosePad is a lightweight and intuitive text editor that implements the most commonly used functionalities for processing text files. The main characteristics of the product are its multi-platform nature and its distribution under the free license GPL and FOSS (Free and Open Source Software).The development is based on the Tauri framework, with the frontend and backend modules primarily utilizing the React framework and the TypeScript and Rust programming languages. The chosen tools and technologies are highly integrated, preventing conflicts between elements and modules while ensuring an optimal level of security and protection.

These characteristics and capabilities make RosePad highly applicable and widely distributed in today's cross-platform world.

УВОД

RosePad е мултиплатформен редактор за създаване и редактиране на файлове, като най-специфична негова характеристика е възможността да работи на различни платформи и ОС. Основни цели на проекта са:

- да създаде open source среда за обработване на текстови файлове (създаване, редактиране, запазване), която работи на различни платформи
- да реализира възможността за обработка (import) на файлове и от формати, различни от специфичния за редактора

За реализиране на тези цели, в милтиплатформения редактор RosePad са интегрирани основните функционалности, необходими за обработване на текст. В процеса на създаването му беше нужно и са решени редица задачи. Масово в ИТ-света са разпространени редактори, които работят на избрана операционна система, но в съвременния мултиплатформен живот това води до някои неудобства (например съвместимост между различни файлови формати, различна визия на редакторите и др.). Тези факти поставят пред RosePad задачата разработката да предлага възможности за интегриране (import) на файлове от други формати, като се запазва начинът на обработка на файловете и след вградено разпознаване и вградена настройка, в редактора те се третират по един и същ начин. Друга важна задача при реализацията на проекта е да се постигне унифицирана, еднаква визия на мултиплатформения редактор при използването му на различни платформи. Тук влиянието на различните операционни системи налагат нуждата от програмиране на отделни модули, с които се постига унифицирана визия на редактора.

В настоящата разработка изложението е представено в 4 части:

Раздел 1: Описание на приложната област

В тази част е направен анализ на пет ключови фактори за развитието на текстовите редактори и наличните крос-платформени решения. Направена е и класификация на този тип софтуер според обхвата им.

Раздел 2: Описание на програмната система / програмното осигуряване

В този раздел основно внимание е отделено на изискванията за функционалност на съвременните текстови редактори. От голямата палитра изисквания най-съществени са: възможността за манипулиране на данни, оформлението на страници в документа, спецификации на шрифта, възможността за добавяне на някои елементи (графики, малки таблици, номерирани списъци и др.). В съвременните текстови редактори стандартни

изисквания вече са вграждането на помощ за правопис и граматика и възможността за използване на бързи клавиши.

Раздел 3: Избор на програмно-технически средства

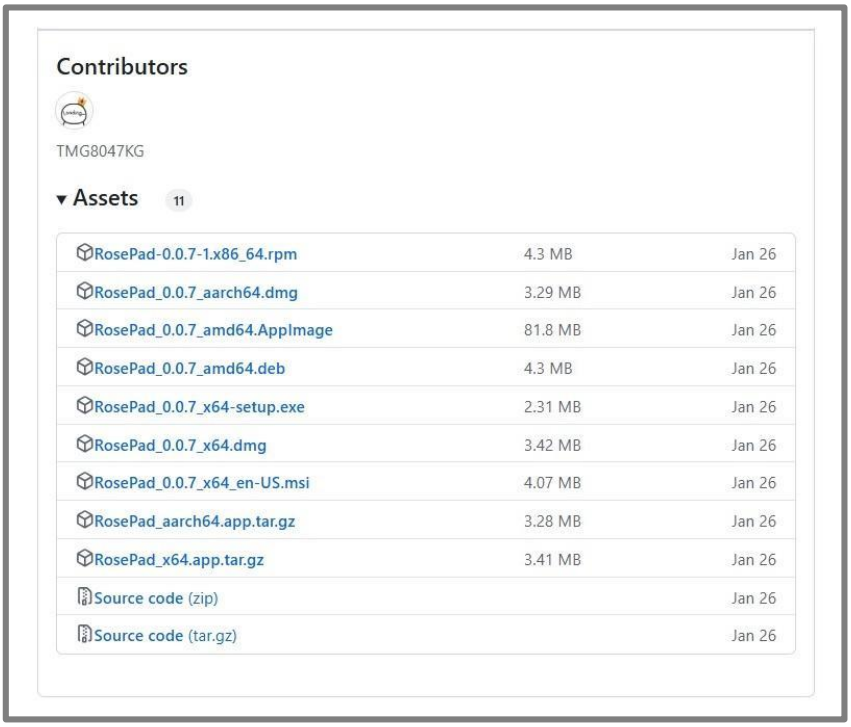
В тази част са представени възможни софтуерни средства, на които може да се базира софтуерното решение за RosePad. Разгледани са характеристики на два framework-а: Tauri и Electron и е посочен избора на Tauri за базов framework. Неговият избор обуславя решението за следващите елементи: React и TypeScript за frontend-частта и езикът Rust за backend-реализацията.

Раздел 4: Описание на реализацията (архитектура на системата и интерфейс).

В раздела са представени схеми, алгоритми и обяснения за структурата и използването на RosePad. Отделно са разгледани характерни елементи при създаване на заглавната страница, използването на външен за редактора файл, описание на работния екран на редактора и настройките му.

В заключителната част на разработката акцентът е върху специфичните характеристики на мултиплатформения редактор и авторския принос (иновация) при реализация на проекта.

RosePad е разработен под свободен лиценз Creative Commons (GPL-3.0 License) и е достъпен на адрес:
https://github.com/TMG8047KG/RosePad



снимка 1: Екран на github release с инсталационните файлове на различни ОС

ИЗЛОЖЕНИЕ

РАЗДЕЛ 1: Описание на приложната област

Текстовите редактори са фундаментален инструмент използван в ежедневието. Хората най-често използват редактори за писане на текст, оформянето му, добавяне на изображения и други действия. Има разработени редица текстови редактори, които обаче са зависими от платформата, на която работят. А в съвременния дигитален свят потребителите използват едновременно различни платформи и значението и необходимостта от крос-платформените продукти нараства. Това се дължи на няколко ключови фактора:

1. Глобализация и дистанционна работа: С увеличаването на глобализацията и възможностите за дистанционна работа, хората често работят с колеги или клиенти, които използват различни операционни системи (Windows, macOS, Linux). Крос-платформените текстови редактори позволяват съвместна работа без значение от използваната платформа.

2. Мобилност и гъвкавост: Много хора използват различни устройства – лаптопи, таблети, смартфони – и искат да могат да продължат работата си безпроблемно на всяко от тях. Крос-платформените редактори предлагат тази гъвкавост.

3. Облачни услуги и синхронизация: Облачните услуги като Google Drive, Dropbox и OneDrive улесняват синхронизацията на документи между различни устройства и платформи. Това прави крос-платформените текстови редактори още по-привлекателни.

4. Потребителски предпочитания: Потребителите все повече търсят решения, които са лесни за използване и не изискват сложни настройки или инсталации. Крос-платформените текстови редактори често предлагат интуитивни интерфейси и лесна интеграция с други приложения.

5. Отворени стандарти: Поддръжката на отворени файлови формати като Markdown, OpenDocument Format (ODF) и други насърчава съвместимостта между различните платформи и приложения.

Някои от най-популярните крос-платформени текстови редактори са:

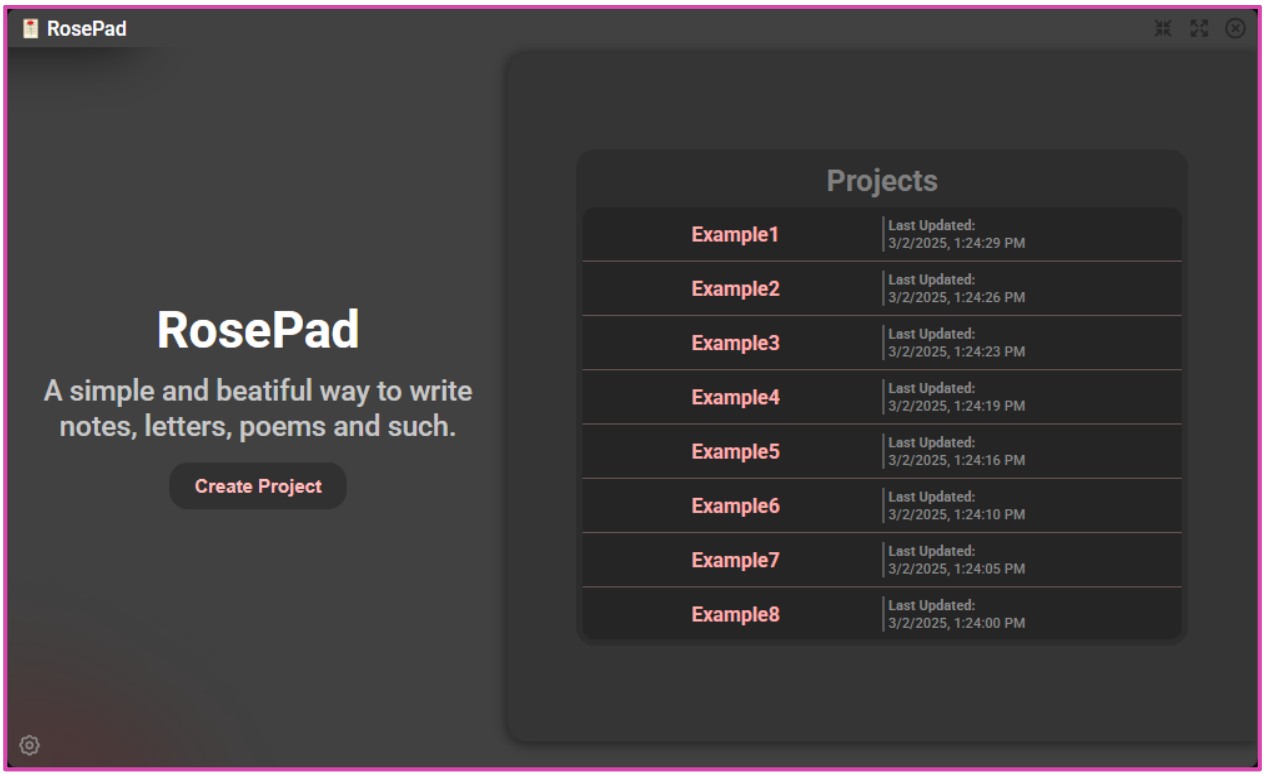
- Google Docs: Част от Google Workspace, този онлайн редактор предлага мощни функции за сътрудничество и е достъпен от всяко устройство с интернет връзка.

- Microsoft Office Online: Включва Word Online, Excel Online и PowerPoint Online, които също са част от Microsoft 365 и предлагат добра съвместимост с настолните версии на тези приложения.
- LibreOffice: Отворен код офис пакет, който включва Writer (текстов редактор), Calc (електронни таблици) и Impress (презентации). Той е наличен за Windows, macOS и Linux.
- OnlyOffice: Друг отворен код офис пакет, който предлага подобни функционалности на LibreOffice и е достъпен за различни платформи.

Тези тенденции показват, че крос-платформените текстови редактори ще продължат да бъдат важна част от ежедневната работа на много хора, особено с нарастващата нужда от гъвкавост и мобилност.

От друга страна може да се направи класификация на текстови редактори според обхвата им. Така те се разделят на два типа:

1. Леки редактори (като Notepad, Vim, Sublime Text) - бързи и минималистични, но с ограничена функционалност за форматиране.
2. Комплексни платформи (като Microsoft Word, Google Docs, MS Word 365) - богати на функции, но тежки, платено-зависими и често свързани с конкретна операционна система (ОС).



снимка 2: Начален екран при стартиране на редактора

Основна цел на RosePad е да създаде лек текстов редактор, който комбинира лекота на използване и реализация на най-нужните функционалности за оформяне на текст. Не на последно място предимство на продукта е неговата 100% прозрачност - изцяло под лиценз Creative Commons (MIT License)

РАЗДЕЛ 2: Описание на програмната система / програмното осигуряване

В началото, при възникването си, изискванията към компютърните текстови редактори са били минимални - основно да запазват и редактират текст. Към съвременните продукти има много повече изисквания за функционалност. Тях основно можем да групираме в следните насоки:

2.1. Обработване на данни

Възможността за манипулиране на текста в документ остава основна за текстообработващ редактор, като тя се реализира чрез основните дейности вмъкване, изрязване и поставяне и копиране на текст. По-усъвършенстваните възможности могат да включват обвиване на думи, при което програмата автоматично преминава към следващия ред, след като е запълнен текущия ред с текст. Възможността за бързо писане, редактиране и преместване на текст е това, което прави текстообработващите програми толкова ценни инструменти за всеки потребител на компютър.

2.2. Оформление на страниците

Повечето софтуери за текстообработка позволяват на потребителите да променят оформлението на създавания от тях документ. Това може да включва промяна на размера на страницата, ориентация на страницата, полетата и отстъпите, добавяне на колони, възможности за огледално оформление (като вариант за предварителна подготовка за печат на файл). Тя може да включва и възможност за създаване на заглавия и подзаглавия, както и на варианти за номериране на страници.

2.3. Спецификации на шрифта

Друга стандартна функция на текстообработващите програми е възможността за промяна на вида на шрифтовете в документа. Повечето програми дават възможност на потребителите да удебеляват, правят курсив и подчертават текста, да използват цветово отличаване, както и да променят размера на шрифта. Обикновено се предлагат различни

шрифтове, които помагат на потребителите да създадат по-еднообразен и по-лесен за четене документ.

2.4. Графична система

Основните текстообработващи програми предоставят на потребителите възможност за добавяне на елементарни таблици, графики или номерирани списъци, но тези с по-разширени възможности предлагат по-големи възможности за добавяне и оформяне на такъв тип визуални елементи. Някои редактори имат функционалност за вграждане на илюстрации, по-сложни графики, директни линкове към интернет-адреси и дори видеоклипове. Тези графики могат да бъдат създадени в самия софтуер за текстообработка или да бъдат създадени в друга програма и след това копирани.

2.5. Помощ за правопис и граматика

Напоследък почти всички съвременни редактори (особено cloud-базираните) предоставят възможност за проверка на правописа, както и основна проверка на граматиката (най-вече на английски език). Друга съвременна възможност, която предоставят редакторите са предложенията за избор на думи, която се реализира чрез речник за търсене или друг вариант на организация като база данни.

2.6 Бързи клавиши (настройки)

С течение на времето се наложи практиката на използване на бързи клавиши за най-често прилаганите операции. Те дават възможност на потребителя да оптимизира работата си, да се концентрира върху същността в създавания файл и да минимизира занимаването си с технически действия. За функционалността на избрана клавишна комбинация има нещо като вид неписана схема, конвенция, но няма строго дефинирани функционалности дори за най-използваните клавишни комбинации. Затова в RosePad се използват стандартните (by default) комбинации за бързи клавиши.

При използване на редактора има възможност за проверка на правописа в текст, като за това се използват опциите на операционната система или браузъра. Този начин на реализация интегрираността на редактора с настройки на операционната система като се запазва мултиплатформеността на проекта.

РАЗДЕЛ 3: Избор на програмно-технически средства

Основните програмно-техническите средства на RosePad са Framework - като основа за изграждане на продукта, Frontend - като базова организация за реализиране на UI (User Interface) и Backend - като основна техника за запазване и управление на съдържанието.

3.1 Framework

Изборът на технически средства за създаването на RosePad минава през разучаване на възможностите и технологиите за създаване на Web-Based приложение. Едно от главните изисквания е да бъде възможно най-леката основа, на която е изградена платформата, докато паралелно с това да има възможност за cross-platform solutions и създаване на желания продукт. След проучване и анализ на предимства, възможности и недостатъци на различни framework-системи, авторът редуцира избора до две опции:

а) Tauri

Tauri v2 е модерен framework за разработка на десктоп приложения, който позволява създаването на крос-платформени приложения (Windows, macOS, Linux) с web технологии (HTML, CSS, JavaScript). Друго предимство на този софтуер е интегрираната опция за backend логика с Rust. Тя се отличава с малък размер на приложенията, висока сигурност и производителност, както и интеграция с операционната система. Tauri v2 е лесен за използване, поддържа популярни уеб frameworks като React и Vue, предлага лесни и интуитивни възможности за разширяване чрез плъгини, което позволява последващо развитие на проекта RosePad.

б) Electron

Electron е популярен framework за разработка на десктоп приложения, който предоставя възможност за създаване на съвременни крос-платформени (Windows, macOS, Linux) приложения с уеб технологии като HTML, CSS и JavaScript. Той използва Chromium за рендиране на потребителския интерфейс и Node.js за backend логиката, което позволява на разработчиците да използват познати и утвърдени уеб технологии за създаване на десктоп приложения. Electron е известен с широката си употреба в приложения като Visual Studio Code, Slack и Discord.

След анализ на функционалностите, използваемостта и производителността на двете софтуерни системи, авторът избра Tauri като подходящия избор за създаването на мултиплатформения редактор RosePad.

3.2 Frontend

Изборът на Framework Tauri предоставя голям избор от Frontend опции и софтуер за разработчици, включително и добавянето на наш по избор framework. За разработката на frontend частта е избран React в комбинация с TypeScript.

React е JavaScript библиотека за създаване на потребителски интерфейси (UI). Основни технологии и модули, които той предлага са:

- ✓ Компоненти: Многократно използваеми части от кода, които улесняват организирането и поддръжката.
- ✓ Виртуално DOM: Оптимизира актуализациите на интерфейса за по-добра производителност.
- ✓ JSX: Позволява писане на HTML-подобен код в JavaScript, което прави кода по-четлив.
- ✓ Еднопоточни приложения (SPA): Поддържа създаване на приложения без презареждане на страницата.
- ✓ Еднопосочен поток на данни: Данните се предават от родителски към дъщерни компоненти, което прави приложението по-предсказуемо.
- ✓ Голяма общност: Богата екосистема от библиотеки и инструменти.

Основни предимства на React са: лесен и интуитивен за научаване, лек, с висока производителност, възможности на многократна употреба на компонентите.

Използването на React поставя следния проблем - концепциите могат да са сложни и неясни за начинаещи и могат да създадат трудности за тях. От друга страна бърза еволюция изисква чести актуализации и използване на по-нови версии.

3.3 Backend

За реализация на backend-частта на RosePad е използван езикът Rust. Неговите предимства са че, той е съвременен системен програмен език, фокусиран върху безопасност, производителност и паралелизъм. Друг негов позитив е комбинацията на скоростта на езици като C/C++ с модерни функции, които предотвратяват често срещани грешки (например препълване на буфера и проблеми с управлението на паметта). Rust използва система за собственост (ownership) и проверки по време на компилация, което го прави безопасен без нужда от garbage collector.

Предимствата на Rust - лекота и бързина и възможностите на операционните системи, на които мултиплатформеният редактор работи правят RosePad-файловете малки като размер. Това е още едно предимство на продукта и резултат от удачните решения за софтуерни средства за разработка.

РАЗДЕЛ 4: Описание на реализацията (архитектура на системата и интерфейс).

Проектът е реализиран след внимателно планиране на нужните функции за достигане на желания резултат. Винаги се има в предвид и изискването, че дизайнът не трябва да бъде натоварващ за окото, отчитайки и леснотата на ползване на RosePad.

4.1 Създаването на заглавната/стартовата страница

Създаването на първоначалната страница, която се показва при стартиране на приложението, се състои от custom titlebar с цел да избегне многообразния изглед, породен избора на операционните системи и да бъде унифицирана визията на RosePad независимо от платформата, на която работи. Елементи на top-бар лентата са логото, името и стандартните бутони за затваряне, минимизиране и максимизиране. Основния панел съдържа кратък описателен текст на приложението, бутон за създаване на нов проект и лист с всички проекти направени или отворени от приложението. В него е поставен и бутонът за настройките на приложението.

снимка 3: Схема на дизайна на главната страница

При кликването на бутона за създаване на нов проект се показва поле за въвеждане на името на проекта, след което програмата минава през процес, проверяващ наличието на запазена директория на проектите, която се избира от потребителя. В случай, че не бъде такава, се отваря файловия мениджър на операционната система, където по подразбиране предлага системната папка за документи, ако има такава. Тази възможност на мултиплатформения редактор е зависима от файловата система. След избора на директорията се проверява дали в нея вече съществува проект със същото име. Ако няма приложението вече преминава към създаване на самия файл на проекта и зареждането на редактора, като запазва пътя (path) на файла, името на проекта, както и името с неговото разширение в паметта на сесията.

4.2. Отваряне на външен файл

При отварянето на текстов файл или външен проект чрез RosePad, се взема аргумента на файла при отварянето на приложението (който е пътя на самия файл). Стойността на аргумента се добавя в списъка с проектите и чрез нея се отваря. Специфично за мултиплатформения редактор е, че файловете, които не са RosePad проекти запазват разширението си в името. След тази последователност от вградени действия външният проект се отваря като файл, създаден с RosePad.

4.3. Работен екран на редактора

снимка 4: Текстов файл на редактора с опции за оформяне на текст

Екранът на мултиплатформеният редактор се състои от titlebar, който съдържа логото и името на приложението, последвано от името на проекта който, е отворен в момента. Бутонът за настройките се намира до бутоните за манипулиране на прозореца. Следващ елемент на екрана е панелът, в който е разположен sidebar-а и редактора като в sidebar-а са поставени бутонът за връщане към главната страница и бутоните за запазване на проекта.

В полето на редактора при селектиране на текст се активират опциите под селекцията, от които може да се манипулира избрания текст чрез натискане на бутоните за форматиране в менюто. На този етап менюто се състои от бутони за: подчертаване, задраскване, удебеляване и промяна на цвета на текста.

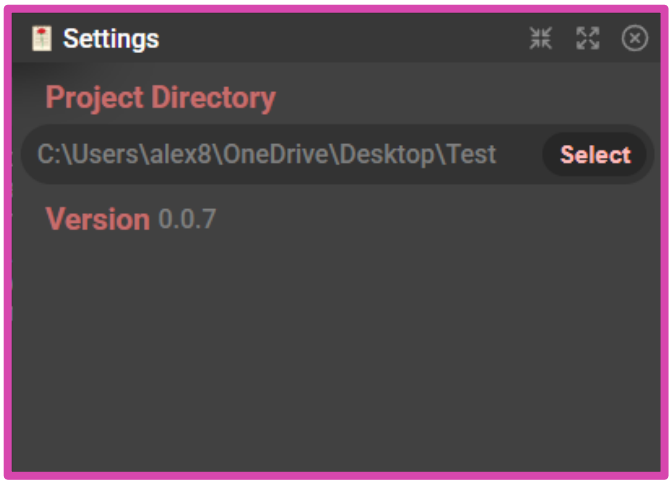
снимка 5: Йерархична схема на елементите на редактора в RosePad

Форматирането на текста става като селекцията бива обградена от стилов блок и след това маха или заменя тези, които вече съществуват в селекцията.

Настройките се състоят от:

✓ опция за промяна на директорията на проектите;

✓ версията на приложението.



снимка 6: Секцията за настройки на RosePad

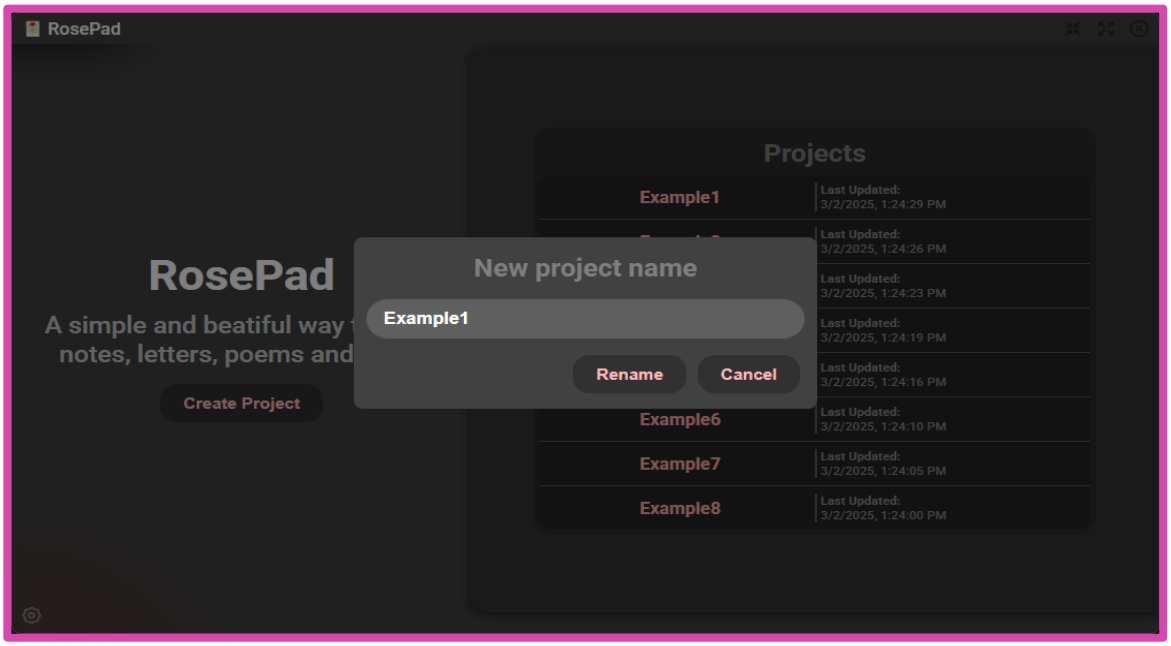
В отделен прозорец се отварят настройките и структура на документа. Това дава възможност на потребителя да вижда промените в реално време за планирани функции за оформление.

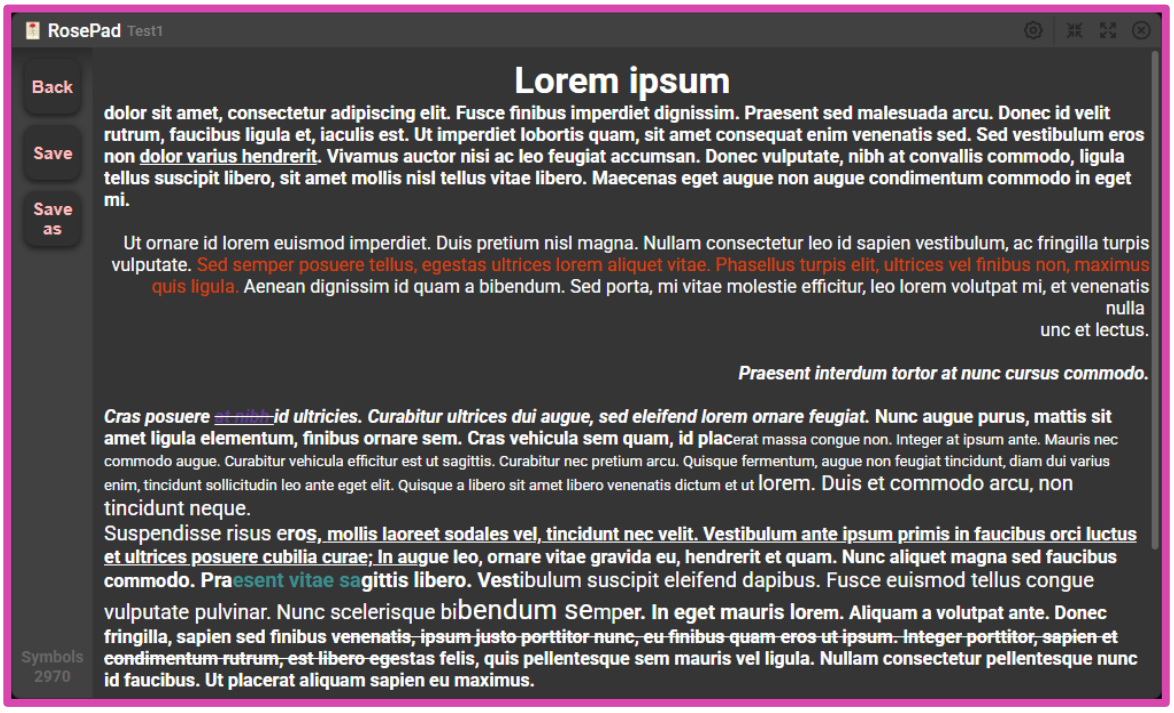
4.4. Интеграция

RosePad има интегрирана система за показване на статуса на потребителя в Discord по време на ползване на приложението. То работи като се свързва с Discord API за контрол на Discord приложение (бот). Това става чрез Client Token на Discord приложението и след това то позволява на RosePad да променя или добавя статус (Rich Presence) в Discord профила на потребителя.

ЗАКЛЮЧЕНИЕ

RosePad е мултиплатформен текстови редактор, който е разработен под open source лиценз и работи под най-използваните съвременни операционни системи.





снимка 7: Екрани при създаване на нов проект и варианти за оформление на текст

Софтуерният продукт е реализиран с използване на съвременни технологии, платформи и езици: Tauri Framework, React (JS библиотека), TypeScript, виртуална DOM-структура, езикът Rust.

Създаденият редактор RosePad постигна поставените цели, като реално създава текстов редактор, който може да работи на най-разпространените операционни системи – Windows, macOS, Linux-дистрибуции. В процеса на реализация са решение следните основни задачи:

✓ Разработване на функционалност, включваща най-необходимите действия при оформление на текст във файл. В редактора това са промяна на шрифт, височина (кегел), подравняване, варианти на шрифта (удебелен, наклонен, подчертан), избор на цвят.

✓ Разработване на система за интегриране. В разработката тя е направена чрез модул за отваряне и вмъкване (import) на файл с друг текстов формат. Характерна особеност на мултиплатформения редактор е, че след обработване файлът може да бъде запазен в оригиналния си формат или в rosepad-формат.

✓ Постигане на унифицираност – върху различните операционни системи визията на редактора е еднаква, като са отчетени и отработени спецификите на ОС. По един и същ начин върху различните платформи изглеждат секциите с команди на редактора, с настройки, екраните при създаване на нов или отваряне на вече съществуващ проект.

✓ Интегриране с комуникационни платформи – на този етап е разработена интеграция с Discord (конкретно с Discord API), като там RosePad може да добавя статус в профила на потребителя.

Архитектурата и избраните програмни средства позволяват лесно добавяне на нови функционалности и модули, без да се получават тежки колизии. В процес на разработка и добавяне на нови функционалности, като най-близки до завършен етап са:

- ✓ опция за community made plugins за разширяване на опциите и възможностите на редактора от потребителя;
- ✓ модул за настройване на дизайна по избор на потребителя (цветови template, изглед и др);
- ✓ функционалности за по-силни и развити опции за форматиране на текста и добавяне на още видове елементи (по-сложни таблици, диаграми, схеми);
- ✓ модул за промяна на режима от notepad-like на word-like (от безразмерно поле на А4 листи).

- ✓ развиване на интеграция с AI, например правопис, структура на документ и др.

Мултиплатформеният редактор RosePad (версия 1.0) е реално работещ под най-използваните операционни системи Windows, macOS. Linux-базирани ОС. Архитектурата и структурата му позволяват по-нататъшното му развитие и добавяне на нови функционалности, а неговият open source лиценз е възможност за широкото му разпространение и използване и перспектива за развитие на общност от потребители.

Използвани източници

https://v2.tauri.app (последно посетен на 01.03.2025)

https://developer.mozilla.org/en-US/ (последно посетен на 01.03.2025)

https://react.dev/ (последно посетен на 01.03.2025)

https://flowbite.com/icons/(последно посетен на 01.03.2025)

https://docs.rs/discord-ipc-rp/0.1.1/discord_ipc_rp/ (последно посетен на 03.03.2025)

Приложения

Приложение 1: Програмен код на отваряне на външен файл

Приложение 2: Програмен код на създаване на проект

Програмен код на отваряне на външен файл

```rust
#[tauri::command]
async fn get_args() -> Vec<String> {
    let mut arg_list = vec![];
    for arg in env::args() {
        arg_list.push(arg);
    }
    return arg_list;
}
```

```javascript
const openedFromFile = async () => {
    if(!path){
      path = await pathFromOpenedFile(); //full path (aka with
/file_name.extension)
      const exists = await projectExists(path);
      if(path){
        const splitPath = path.split(/[/\\]/g);
        let name = splitPath[splitPath.length-1];
        const project = name.split(".");
        sessionStorage.setItem("path", path);
        if(project[1] == "rpad"){
          name = project[0] //file_name
        }
        sessionStorage.setItem("name", name);
//file_name.extension
        sessionStorage.setItem("projectName", name);
        if(!exists) await addProject(name, path);
        await rpc_project(name, path);
        navigator(`/editor/${name}`);
      }
    }
  }
  openedFromFile();
```

Програмен код на фрагмента за създаване на проект

```
const handleCreateProject = async (name: string) => {
    const raw = await readTextFile(settingsFile, { baseDir:
BaseDirectory.AppConfig })
    const data = JSON.parse(raw)
    let dir = data.projectPath;
    console.log(dir);
    if(dir == "null" || !dir){
      dir = await selectDir()
    }
    if(data.projects.some((projectName: { name: string; }) =>
projectName.name === name)){
      return alert("A project with this name already
exists!");
    }
    await createProject(dir, name);
    setIsModalOpen(false);
    navigator(`/editor/${name}`);
  };
```

```
async function createProject(dir:string, name:string) {
  const filePath = `${dir}\\${name}.rpad`;
  const file = await create(filePath);
  await file.close();
  await rpc_project(name, filePath);
  sessionStorage.setItem("name", `${name}`); //file_name
  sessionStorage.setItem("projectName", name);
  sessionStorage.setItem("path", filePath) //with extension
  addProject(name, filePath);
}
```

```
export async function addProject(name:string, path:string|URL)
{
    const rawJson = await readTextFile(settingsFile, {
baseDir: BaseDirectory.AppConfig });
    let settings = JSON.parse(rawJson);
    let now = new Date();
    settings.projects.push({
        name: `${name}`,
        last_updated: `${now.toLocaleString()}`,
        path: `${path}`
    })
    let json = JSON.stringify(settings, null, 2);
    await writeTextFile(settingsFile, json, { baseDir:
BaseDirectory.AppConfig })
}
```