МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Секція інформаційно-комунікаційних технологій

ВИПУСКНА РОБОТА БАКАЛАВРА

на тему:

«Додаток-асистент для онлайн гри на мові JavaScript»

Завідувач
випускаючої кафедри **Довбиш А. С.**

Керівник роботи **Власенко О. В.**

Студента групи ІН-62 **Перепелиця Б. В.**

СУМИ 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютерних наук

Затверджую_____

Зав. кафедрою Довбиш А.С.

"_____" _____2012 г.

ЗАВДАННЯ
до випускної роботи

Студента четвертого курсу, групи ІН-62 спеціальності "Інформатика" денної форми навчання Перепелиці Богдана Віталійовича.

Тема: " **Додаток-асистент для онлайн гри на мові JavaScript** "

Затверджена наказом по СумДУ

№_____ от _____2020 г.

Зміст пояснювальної записки: 1) інформаційний огляд методів розробки настільних додатків та постановка завдання; 2) аналіз та вибір методів вирішення поставленого завдання, а також вирішення проблеми розповсюдження настільного додатку; 3) опис та аналіз результатів роботи.

Дата видачі завдання "_____"_____2020 г.

Керівник випускної роботи _____ Власенко О.В.

Завдання приняв до виконання _____ Перепелиця Б.В.

РЕФЕРАТ

Записка: 44 стор., 17 рис., 1 додаток, 8 джерел.

Об’єкт дослідження — настільні додатки.

Мета роботи — дослідження і розробка настільного додатку на мові JavaScript із використанням голосових порад, а також розповсюдження додатку із використанням інсталятора.

Результати — розроблено настільний додаток для гри League of Legends із використанням таких інструментів як Java Script, Type Script, electronjs, Angular та ін. Розроблено інсталятор на основі мови NSIS, за допомогою якого розповсюджується готовий додаток.

ДОДАТОК-АСИСТЕНТ ДЛЯ ОНЛАЙН ГРИ НА МОВІ JAVASCRIPT

ЗМІСТ

ВСТУП

Зараз ми не уявляємо своє життя без персонального комп'ютера, бо за його допомогою шукаємо інформацію, спілкуємося з друзями, дивимося фільми, а також граємо в ігри. Кіберспорт у наш час є невід'ємною частиною життя багатьох людей. Школярі, студенти а також досить дорослі люди грають як для розваги, так і на професійному рівні. Але не всі однаково швидко адаптуються до того чи іншого виду ігор, тому що це вимагає від людини багато часу та зусиль, а також запам'ятовування великої кількості інформації.

Аліса, Сірі, Алекса, Кортана – все це жіночі імена голосових асистентів від найбільших світових компаній. Вони можуть викликати вам таксі, показати найкоротший шлях до роботи, ввімкне потрібну музику і попередити коли буде дощ. Наразі голосові асистенти займають важливе місце в житті людей, тому що вони допомагають скоротити час на виконання повсякденних речей.

У даний час існує багато компаній, які займаються розробкою допоміжного ПО для гравців комп'ютерних ігор, з яких Mobalytics, Overwolf та багато інших. Зараз ця сфера активно розвивається, тому що на неї є попит. Наприклад, Overwolf надає можливість створювати власні додатки, використовуючи їх API, які після детального тестування будуть доступні до завантаження у їх каталозі. Ця компанія наразі є найбільшим агрегатором подібних додатків на ринку. Але, незважаючи на велику кількість ігрових додатків, поки що жоден не надає функціоналу асистента.

Мета випускної роботи – створити голосового асистента як настільний додаток, який допоможе початківцям адаптуватися до нової гри, а також вже досвідченим гравцям нагадає про важливі речі або події, про які вони могли забути. Наразі іще немає подібних асистентів, тому **робота є актуальною**.

1. ІНФОРМАЦІЙНИЙ ОГЛЯД

1.1. Поняття настільного додатку

Настільний додаток – це програма, яка працює на вашому робочому столі. У нашому випадку це клієнт-серверний додаток, тому що він буде отримувати частину даних від веб-сервера. Настільні додатки зручні тим, що вони можуть взаємодіяти із іншими програмами на комп'ютері користувача та отримувати від них необхідну інформацію.

Найбільш поширеною при розробці додатків є electron.js бібліотеку, архітектура додатку буде виглядати наступним чином:

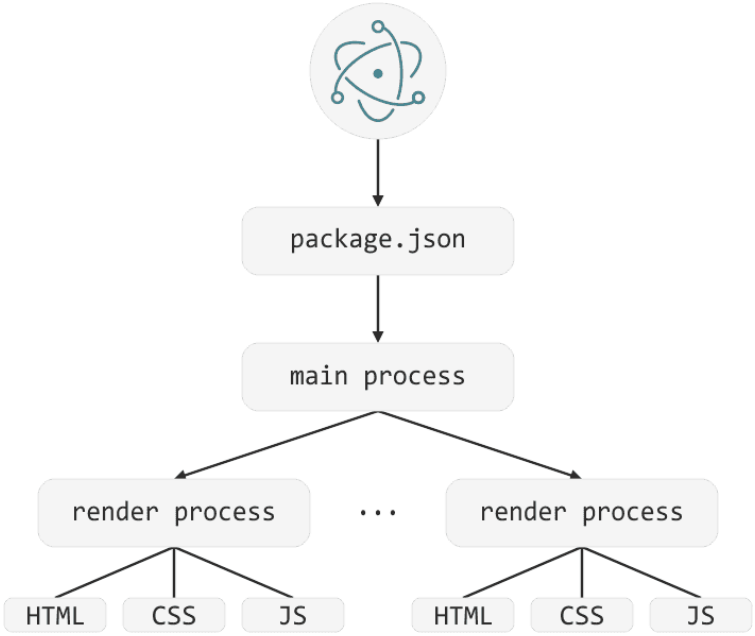

Рисунок 1.1 – Архітектура Electron.js [9]

В electron-додатку існує два типи процесів: основний процес і процес візуалізації. Основний процес (main process) – це процес, який запускається через package.json. Скрипт, запущений в основному процесі, відображає користувацький інтерфейс, тобто GUI, використовуючи веб-сторінки. В electron-додаток завжди має лише один main процес. Він керує процесами візуалізації.

Процес візуалізації (renderer process) відповідний за запуск користувацького інтерфейсу додатку, або, іншими словами, веб-сторінки, що є екземпляром webContents (відповідає за рендеринг та керування веб-сторінкою). Всі API інтерфейси DOM, API-інтерфейси node.js і підмножина API-бібліотек Electron доступні в даному процесі.

1.2. Альтернативи Electron.js.

Для розробки настільних додатків використовують безліч технологій, серед яких NW.js, Qt (Qml), GTK та інші. Зупинимося детальніше на кожному з них.

1.2.1. NW.js.

NW.js, раніше відомий як проект «node-webkit», являє собою додаток, заснований на Chromium і Node.js. Існує можливість писати власні програми в HTML і Javascript за допомогою NW.js. Він також дозволяє безпосередньо звертатися до модулів io.js з DOM і дозволяє використовувати новий спосіб написання власних програм з усіма веб-технологіями. Він створений і розроблений в технологічному центрі Intel з відкритим вихідним кодом.

Має повну підтримку API Node.js і всіх його сторонніх модулів. Хороша продуктивність: io.js і WebKit працюють в одному потоці: виклики функцій виконуються просто; об'єкти знаходяться в одній купі і можуть просто посилатися один на одного; Простота упаковки і поширення додатків. Доступно в Linux, Mac OSX і Windows.

Переваги та недоліки:

1. Захист вихідного коду. У Electron немає будь-якого механізму для захисту вашого вихідного коду. Asar важко назвати захистом, враховуючи що це простий tar-архів і фактично будь-який користувач може «розпакувати» вашу програму як звичайний архів і отримати доступ до всіх ресурсів і вихідного коду.

NW.js дозволяє зібрати виконуваний файл із захистом через V8 Snapshot. Дане рішення не компілює JavaScript в машинний код (як стверджує

документація) і не забезпечує повну безпеку вихідного коду. По суті це просто дуже добре обфускований код. Але якщо єдина альтернатива – залишити вихідний код абсолютно відкритим, то багато розробників віддадуть перевагу V8 Snapshot, навіть з огляду на втрату приблизно 30% продуктивності.

2. Час запуску. Додаток на Electron запускається помітно швидше як на Windows, так і на OSX. Навіть якщо відключити V8 Snapshot, NW.js-додаток завантажується набагато повільніше [1].

3. Open Source. Свого часу Electron зробив сміливий крок, підтримавши IO.js в момент застою розробки Node.js. Це означає, що Electron в великій мірі прагне підтримувати передові можливості JavaScript, в той час як NW.js більше орієнтується на підтримку сумісності (принаймні в теорії).

Також не можна не помітити високу швидкість розробки Electron. Більше сотні коммітів в тиждень, по десять релізів на місяць. Команда розробників активно відповідає на питання на github. З іншого боку, NW.js все ще знаходиться в версії 0.15, а github-документація здається досить застарілою. Наприклад, регулярно можна побачити згадки назви «node-webkit», хоча проект був перейменований кілька років тому.

Висновок: Electron працює швидше та має набагато детальнішу документацію, ніж NW.js. Відсутність захисту вихідного коду компенсується тим, що використовується TypeScript для розробки, який у подальшому компілюється у обфускований JavaScript.

1.2.2. Qt (QML)

Qt – це потужний набір інструментів для написання кросплатформених додатків. У останніх версіях даного фреймворка з'явився компонент Qt Quick для створення програмного забезпечення з використанням декларативної мови опису розмітки QML. Qt в загалі і QML зокрема – це перевірені, що знайшли своє застосування практично у всіх сферах – від embedded до програмних оболонок операційних систем.

Характерна особливість – використання метаоб'єктного компілятора – попередньої системи обробки вихідного коду. Для того щоб розширити стандартні можливості використовується система плагінів, які розміщуються безпосередньо в панелі візуального редактора. Також є можливість розширити звичайні функції віджетів, пов'язані з їх розміщенням на екрані, відображенням, перемальовуванням при зміні розмірів вікна.

Комплектується візуальним середовищем розробки графічного інтерфейсу Qt Designer, який дозволяє створювати діалоги і форми з використанням режиму WYSIWYG. Також у складі Qt є Qt Linguist – графічна утиліта, яка спрощує локалізацію і переклад програми на багато мов; і Qt Assistant – довідкова система Qt, яка дозволяє створювати кросплатформену довідку для розробленого на основі Qt програмного забезпечення. Починаючи з версії 4.5.0 в комплект включене середовище розробки Qt Creator, що містить довідку, редактор коду, графічні засоби Qt Designer і надає можливість налагоджувати додатки.

Основною мовою програмування в Qt є C++ (в QML можна використовувати javascript). Однак для Qt і QML є біндінги для інших мов програмування (наприклад C#). Однак офіційно підтримується тільки інтеграція з Python. Все інше – сторонні реалізації [2].

Переваги:

- швидкість роботи;
- мале використання ресурсів.

Недоліки:

- складність розробки;
- неможливість використання web компонентів;
- велика вага додатків.

1.2.3. GTK

Бібліотека для побудови призначеного для користувача інтерфейсу, яка починалася як частина проекту GIMP і згодом була виділена в окремий проект.

Є інструмент Glade для швидкої розробки призначеного для користувача інтерфейсу. Основною мовою для розробки GTK є C, проте є біндінг для багатьох популярних мов програмування. Більш того, з використанням C# біндінгу створювалася MonoDevelop – одна з найпотужніших IDE для розробки під C#. Однак розробка майже припинилася [3].

1.3. Постановка задачі

Необхідно розробити настільний додаток-асистента, який буде допомагати гравцям із адаптацією до нових ігор.

Функціонал передбачений для цієї програми:

- голосові подари під час гри;
- нагадування про ігрові події;
- поради про купівлю предметів;
- поради про прокачування здібностей;
- оверлей з інформацією та субтитрами;
- графік цінності команд.

1.4. Розповсюдження програми

Для розповсюдження програми буде використано веб-інсталлер – це невеликий пакет, який автоматично визначає і завантажує тільки компоненти, необхідні для конкретної платформи. Для створення інсталлера у electron використовується electron-builder. Його конфігурують у файлі package.json під кожну операційну систему окремо. Для Linux та Mac це буде deb пакет, а для Windows – NSIS веб-інсталлер.

Також для того щоб у користувача працював потрібний нам функціонал – необхідно конфігурувати NSIS інсталлер таким чином, щоб він встановив необхідні для роботи програми пакети, наприклад Microsoft C++ Build Tools.

Аналогами NSIS є Inno Setup, WiX Toolset та ін.

1.4.1. NSIS

NSIS – широко відомий інструментарій для створення інсталяційних пакетів в середовищі Windows. Відмінні риси – компактність, масштабованість, підтримка плагінів і локалізацій. З інших можливостей NSIS: підтримка сценаріїв, різних варіантів установки (включаючи кілька проектів усередині одного інсталятора), створення веб-інсталяторів.

Дистрибутиви NSIS створюються на основі сценаріїв. Змінні, функції повністю контролюють як інсталяцію, так і деінсталяцію продукту. За допомогою скриптів можна додавати файли і директорії, вносити зміни до реєстру, редагувати текстові і двійкові файли, створювати патчі, управляти змінними середовища і навіть звертатися до Windows API (за допомогою розширень). Якщо виникає необхідність розширити можливості інсталятора, можна додати певні функції. Модулі пишуться на C, C++, Delphi або інших мовах. Скомпілювати інсталятор можна за допомогою інструменту makensis, попередньо обравши тип компресії – ZLib, BZip2 або LZMA. Крім того, сам по собі інсталятор займає небагато місця – всього 34 КБ [4].

1.4.2. Inno Setup

Inno Setup – інструментарій для створення інсталяторів на базі сценаріїв. Серед головних особливостей продукту –- робота з усіма версіями ОС Windows (Windows 2000 і вище), розширена підтримка 64-бітових додатків, настроюються типи установки, вбудований препроцесор і потужна мова сценаріїв на основі Pascal.

Сценарії мають структуру, характерну конфігураційним файлів ini-формату, тому код цілком добре читаємо і зручний в редагуванні. Документ розділений на секції, і кожна з них відповідає за певну задачу інсталятора відповідно. Всього є два типи секцій - з параметрами і «директива-значення».

Розробникам, знайомим з Pascal, буде на руку той факт, що Inno використовує схожу мову – RemObjects Pascal Script. Сценарії відповідають за додавання нових опції, створення інтерфейсу, виклик файлів або бібліотек, налаштовують дії (custom actions) і т. д. За замовчуванням в Inno Setup входить компілятор сценаріїв Compiler IDE.

З інших особливостей Inno Setup – тиха інсталяція та деінсталяція. Користувачеві доступні різні типи установки і локалізації на вибір. Установник може бути зашифрований, містити цифровий підпис або захищений паролем. До дистрибутива може застосовуватися bzip2- або LZMA / LZMA2-компресія. Інтерфейс – в стилі 2000 / XP, інших варіантів не передбачено.

1.4.3. WiX Toolset

WiX (Windows Installer XML) – набір інструментів для створення інсталяторів (Windows Installer) з використанням специфікації XML. З особливостей: інтерфейс командного рядка, інтеграція з IDE, автоматизація процесів, підтримка базових і розширена підтримка Windows Installer.

Ядро WiX складають компілятор (candle), компонувальник (light), бібліотекар (lit), декомпілятор (dark), бутстраппер інсталяції (burn) та багато інших програм. З їх допомогою можна створювати пакети інсталяції .msi, модулі злиття .msm, патчів .msp.

Створення інсталятора складається з декількох основних етапів: розробка сценарію, його перевірка, обробка препроцесором, компіляція і компонування. Для складання інсталятора та автоматизації в цілому використовується платформа MSBuild. Автоматизація – одна з головних переваг WiX при роботі з досить ресурсоємними проектами.

Вихідний код має формат XML (розширення файлу wxs). Сценарій описує всі етапи установки, починаючи з опису проекту закінчуючи діями після установки. Редагувати його можна в будь-якому редакторі з підтримкою даного формату.

2. ВИБІР МЕТОДУ РІШЕННЯ

2.1. Особливості архітектури

Програма буде складатися з трьох частин:

- main процес – основний скрипт, який буде запускати програму;
- два renderer процеси – один для основного вікна програми, другий для оверлея;
- python бекенд.

Для комунікації renderer процесів із бекендом буде використовуватися HTTP протокол. Зв’язок процесів між собою буде відбуватися через IPC (Inter-Process Communication).

2.2. Програмні засоби основного процесу

У функції основного процесу входить запуск програми, керування процесами візуалізації, взаємодію між процесами та робота з даними із гри. Для даного процесу ми будемо використовувати такі технології як NodeJS, electron.js та IPC.

Electron є конкурентоспроможним рішенням при розробці кроссплатформенних додатків. Не вимагає від команди розробки знання таких мов програмування як C++ та Python, дозволяє створювати додатки, використовуючи багаті можливості веб-технологій HTML, CSS, JS, а також Node.js. Electron з коробки надає різноманітний функціонал по роботі з вікном програми. Також великою перевагою фреймворка є те, що цей проект поширюється як open source і має активну ком'юніті. Головним недоліком Electron є велике споживання оперативної пам'яті і велику вагу навіть мінімального додатку.

2.2.1. Electron

Electron (раніше відомий як atom shell) – фреймворк, розроблений GitHub. Дозволяє розробляти нативні графічні додатки для настільних операційних систем за допомогою веб-технологій. Фреймворк включає в себе Node.js для роботи з back-end і бібліотеку рендеринга з Chromium [5].

На базі Electron побудований не тільки текстовий редактор для програмістів Atom, а й такі програмні продукти для розробників, як Visual Studio Code, Light Table (починаючи з версії 0.8), Ionic Lab, Avocode, REPL-консоль Mancy для фреймворків Node.js і Meteor.js, Mongotron - GUI-менеджер для MongoDB. Крім того, на основі цього фреймворка написано клієнтську програму чату Slack, Skype, Discord, десктопний клієнт WordPress і багато іншого.

2.2.2. NodeJS

Node або Node.js – програмна платформа, заснована на движку V8 (здійснює трансляцію JavaScript в машинний код), що перетворює JavaScript з вузькоспеціалізованого мови в мову загального призначення. Node.js додає можливість JavaScript взаємодіяти з пристроями введення-виведення через свій API (написаний на C++), підключати інші зовнішні бібліотеки, написані на різних мовах, забезпечуючи виклики до них з JavaScript-коду. Node.js застосовується переважно на сервері, виконуючи роль веб-сервера, але є можливість розробляти на Node.js і десктопні віконні додатки (за допомогою NW.js, AppJS або Electron для Linux, Windows і macOS) і навіть програмувати мікроконтролери (наприклад, tessel і espruino). В основі Node.js лежить подієво-орієнтоване і асинхронне (або реактивне) програмування з неблокуючим введенням/виведенням [6].

2.2.3. IPC

Взаємодія між процесами (англ. Inter-Process Communication, скорочено англ. IPC) – це набір засобів, який дозволяє обмінюватися повідомленнями між

процесами. Для взаємодії процесів, які виконуються на одному комп'ютері (під керуванням однієї операційної системи) використовують (взаємодія забезпечується ядром операційної системи, в якій виконуються процеси):

- сигнали – асинхронні повідомлення, які сповіщають про подію (надзвичайну ситуацію), на яку процес має відреагувати виконанням наперед визначеною функцією або командою, тощо;
- неіменовані й іменовані канали для передачі даних у вигляді синхронних (очікуваних) повідомлень; відправка повідомлення відбувається аналогічно запису в файл, отримання – як читання даних з файлу, якщо канал порожній – процес, який чекає на дані, призупиняється до тих пір, поки дані не надійдуть;
- черги повідомлень – пакети даних, що передаються між процесами та доводять до відома отримувача інформацію про надходження пакету;
- сегменти подільної пам'яті – засіб, що дозволяє кільком процесам сумісно використовувати (поділяти) фрагмент оперативної пам'яті з метою обміну даними; відправлення даних відбувається шляхом запису в пам'ять, отримання – читанням з пам'яті [7].

2.3. Програмні засоби процесів візуалізації

Завданням даних процесів є відображення інформації, тому буде використано стек технологій HTML + SCSS + Angular 8. Також замість JavaScript буде використано TypeScript для типізації коду, щоб з ним було легше працювати. Проміжні дані для цих процесів будуть зберігатися у localStorage та Cookies. Angular використовується для прискорення роботи інтерфейсу.

2.3.1. TypeScript

TypeScript – мова програмування, яка була представлена Microsoft в 2012 році і позиціонується як засіб розробки веб-додатків, що розширює можливості JavaScript. TypeScript є зворотньо сумісним з JavaScript і компілюється в останній. Фактично, після компіляції програму на TypeScript можна виконувати

в будь-якому сучасному браузері або використовувати спільно з серверною платформою Node.js. Код експериментального компілятора, який транслює TypeScript в JavaScript, поширюється під ліцензією Apache. Його розробка ведеться в публічному репозиторії через сервіс GitHub.

TypeScript відрізняється від JavaScript можливістю явного статичного призначення типів, підтримкою використання повноцінних класів (як в традиційних об'єктно-орієнтованих мовах), а також підтримкою підключення модулів, що покликане підвищити швидкість розробки, полегшити читабельність, рефакторинг і повторне використання коду, допомогти здійснювати пошук помилок на етапі розробки і компіляції, і, можливо, прискорити виконання програм.

2.3.2. Angular 2+

Angular (версія 2 і вище) – це відкрита і безкоштовна платформа для розробки веб-додатків, написана мовою TypeScript. Angular – це повністю переписаний фреймворк від тої ж команди, яка написала AngularJS.

Спочатку створювався як інша версія AngularJS. Angular 2 був переписаний з нуля на TypeScript, обладнаний іншою архітектурою і не є зворотньо сумісним з AngularJS, у зв'язку з чим для попередження плутанини було вирішено розвинути його як окремий фреймворк, нумерація версії, якого починається з 2.

Фреймворк працює з HTML, що містить додаткові атрибути, які описуються директивами, і пов'язує введення або виведення області сторінки з моделлю, яка представляє собою звичайні змінні JavaScript. Значення цих змінних задаються вручну або витягуються з статичних або динамічних JSON-даних. Angular дотримується MVC-шаблону проектування і заохочує слабкий зв'язок між поданням, даними і логікою компонентів. Використовуючи впровадження залежності, Angular переносить на клієнтську сторону такі класичні серверні служби, як видозалежні контролери. Отже, зменшується навантаження на сервер і веб-додаток стає легше.

2.4. Програмні засоби бекенду

Завданням даного процесу є редагування та зберігання даних про поради, а також формування тригерів для ігрових подій. Для даної задачі буде використано стек Python + Django. Зберігати голосові поради будемо у Google Cloud. Для зберігання текстових порад та інформації про користувачів будемо використовувати базу даних PostgreSQL.

Даний стек підходить найкраще, тому що завдяки Django маємо готовий інструмент для адміністрування порад.

2.4.1. Python

Python – високорівнева мова програмування загального призначення, яка орієнтована на підвищення продуктивності розробника і простоту коду. Синтаксис ядра Python є мінімалістичним, але водночас стандартна бібліотека включає великий обсяг корисних функцій.

Python підтримує багато парадигм програмування, з яких: структурне, об'єктно-орієнтоване, функціональне, імперативне і аспектно-орієнтоване. Також містить у собі такі функції як підтримка багатопотокових обчислень, автоматичне керування пам'яттю, динамічна типізація, механізм обробки виключень, високорівневі структури даних. Підтримується розбиття програм на модулі, які, в свою чергу, можуть об'єднуватися в пакети.

Еталонною реалізацією Python є інтерпретатор CPython, що підтримує більшість активно використовуваних платформ. Він поширюється під вільною ліцензією Python Software Foundation License, що дозволяє використовувати його без обмежень в будь-яких додатках, включаючи пропрієтарні. Є реалізація інтерпретатора для JVM з можливістю компіляції, CLR, LLVM, інші незалежні реалізації. Проект PyPy використовує JIT-компіляцію, яка значно збільшує швидкість виконання Python-програм [8].

2.4.2. Django

Django – вільний фреймворк для веб-додатків на мові Python, який має в основі шаблон проектування MVC. Проект підтримується організацією Django Software Foundation.

Сайт на Django будується з одного або декількох додатків, які рекомендується робити ізольованими. Це одна з основних архітектурних відмінностей цього фреймворка від багатьох інших (наприклад, Ruby on Rails). В основі фреймворка лежить принцип – DRY (англ. Do not repeat yourself), тобто максимальне перевикористання кодової бази. Також, на відміну від інших фреймворків, обробники URL в Django можно конфігурувати явно за допомогою регулярних виразів.

Для роботи з базою даних Django використовує власний ORM, де модель даних описується класами Python, і по ній генерується схема бази даних. Веб-фреймворк Django використовується в таких великих і відомих сайтах, як Instagram, Disqus, Mozilla, The Washington Times, Pinterest, YouTube, Google та ін.

Django використовується у якості веб-компонента в різних проектах, таких як Graphite – система спостереження і побудови графіків, FreeNAS – вільна реалізація системи для зберігання та обміну файлами і ін.

Архітектура Django схожа на «Модель-Вид-Контролер» (MVC). Контролер класичної моделі MVC майже відповідає рівню, який в Django називається «Вид», а презентаційна логіка «Уявлення» реалізована в Django за допомогою рівня Шаблонів. Через це архітектуру Django часто називають «Модель-Шаблон-Подання» (MTV).

Первісна розробка Django як засобу для роботи блогів та ресурсів новин досить сильно вплинула на його архітектуру: він надає ряд засобів, які допомагають у швидкій розробці веб-сайтів інформаційного характеру. Наприклад, розробнику не потрібно створювати контролери та сторінки для адміністративної частини сайту, тому що Django має вбудований додаток для керування вмістом, який можна

включити в будь-який сайт, розроблений із використанням Django, і який може керувати відразу декількома сайтами на одному сервері.

Адміністративний додаток дозволяє керувати будь-якими об'єктами наповнення сайту та протоколюючи всі скоєні дії для запобігання непередбачених ситуацій, і надає інтерфейс для керування користувачами і групами (з пооб'єктним призначенням прав).

2.4.3. PostgreSQL

PostgreSQL – вільна об'єктно-реляційна система управління базами даних (СУБД). Існує в реалізаціях для безлічі UNIX-подібних платформ, включаючи AIX, різні BSD-системи, HP-UX, IRIX, Linux, macOS, Solaris / OpenSolaris, Tru64, QNX, а також для Microsoft Windows.

Сильними сторонами PostgreSQL вважаються:

- високопродуктивні і надійні механізми транзакцій і реплікації;
- розширювана система вбудованих мов програмування: в стандартному постачанні підтримуються PL / pgSQL, PL / Perl, PL / Python і PL / Tcl; додатково можна використовувати PL / Java, PL / PHP, PL / Py, PL / R, PL / Ruby, PL / Scheme, PL / sh і PL / V8, а також є підтримка завантаження модулів розширення на мові C;
- наслідування;
- можливість індексування геометричних об'єктів і наявність базується на ній розширення PostGIS;
- вбудована підтримка слабоструктурованих даних в форматі JSON з можливістю їх індексації;
- розширюваність (можливість створювати нові типи даних, типи індексів, мови програмування, модулі розширення, підключати будь-які зовнішні джерела даних).

PostgreSQL базується на мові SQL і підтримує багато з можливостей стандарту SQL: 2011.

3. ПРАКТИЧНА РЕАЛІЗАЦІЯ

3.1. Проектування бази даних.

У проекті буде використано PostgreSQL. ER-діаграма бази даних виглядає наступним чином (рис. 3.1.):

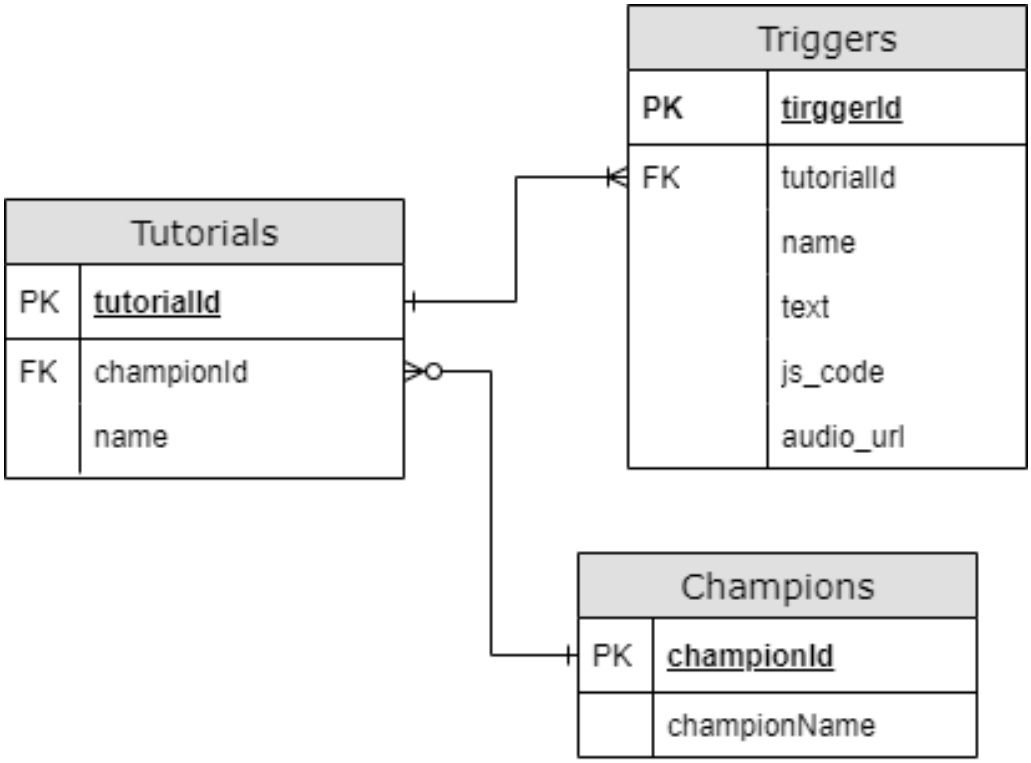

Рисунок 3.1 – База даних додатку

Через те що було використано Python + Django, то база даних створена у вигляді ORM.

3.2. Дизайн та структура додатку.

Перед початком роботи необхідно встановити асистент (рис. 3.2.):

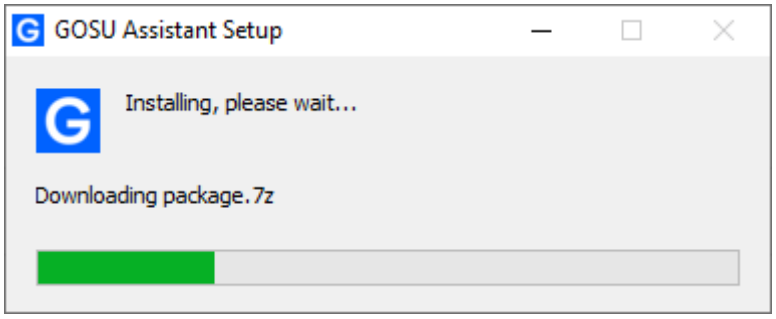

Рисунок 3.2 – Процес установки додатку

Так як використовується nsis web installer, то інсталятор займає до 1 mb. Вже в процесі інсталяції web installer завантажує потрібні програмі дані. Після установки додатку з'являється екран завантаження (рис. 3.3.) і після цього стартова сторінка (рис. 3.4.):

Рисунок 3.3 – Процес завантаження додатку

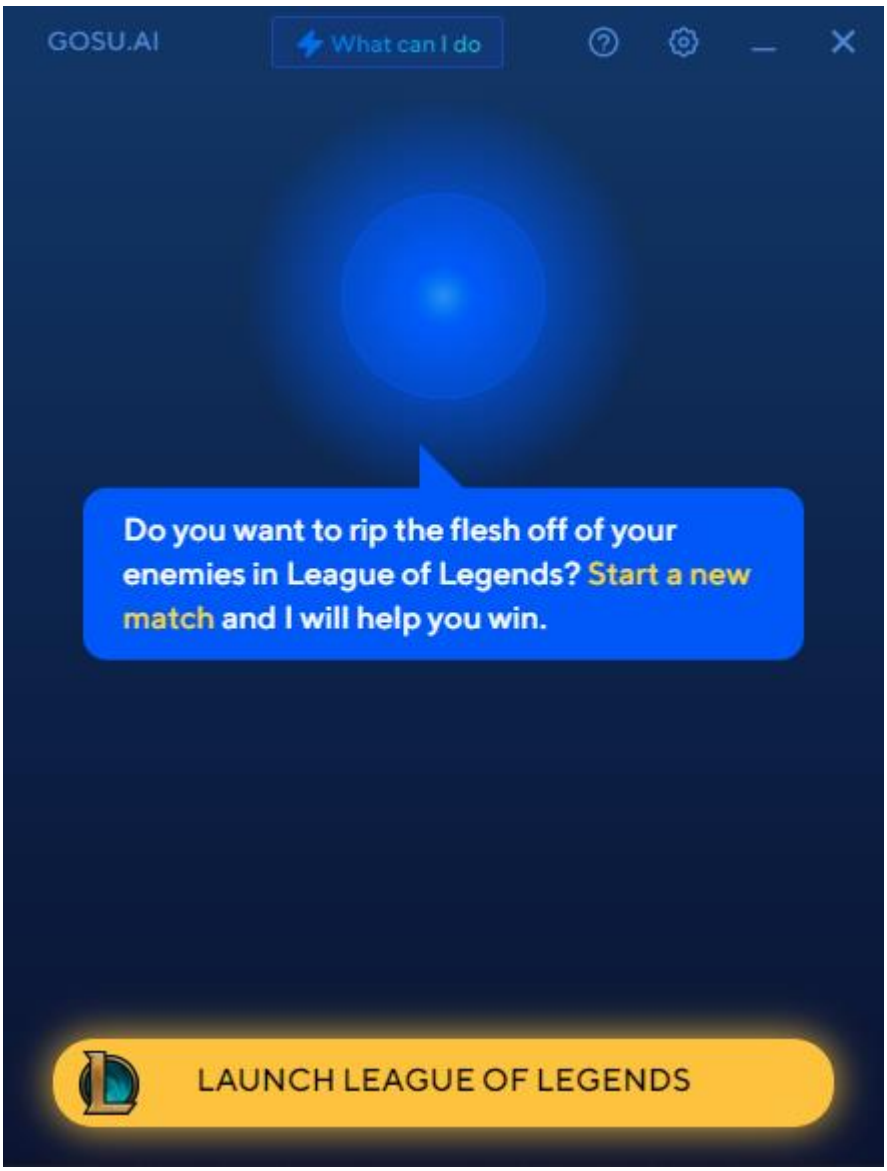

Рисунок 3.4 – Стартова сторінка додатку

За допомогою цієї сторінки існує можливість відкрити гру, згорнути або закрити додаток, змінити налаштування та подивитися можливості додатку.

Сторінка налаштувань виглядає наступним чином (рис. 3.5.-3.6):

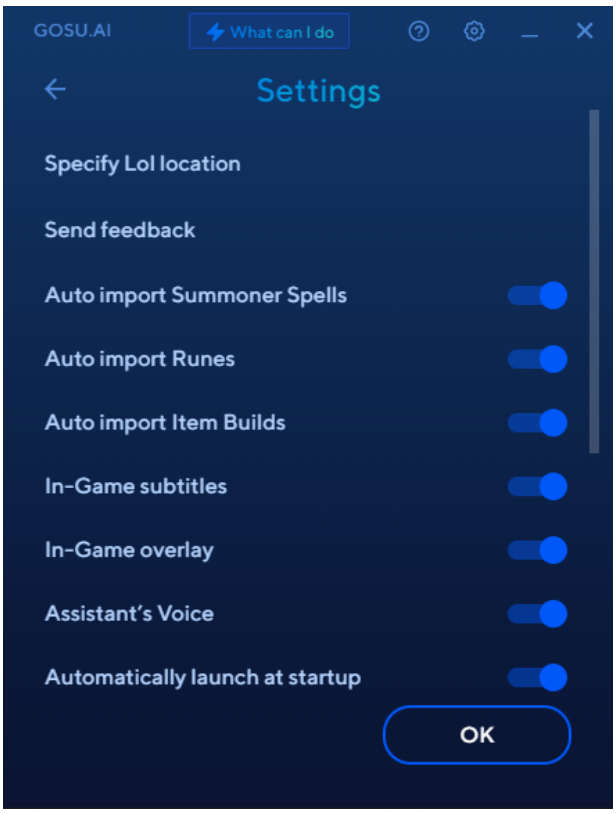

Рисунок 3.5 – Сторінка налаштувань

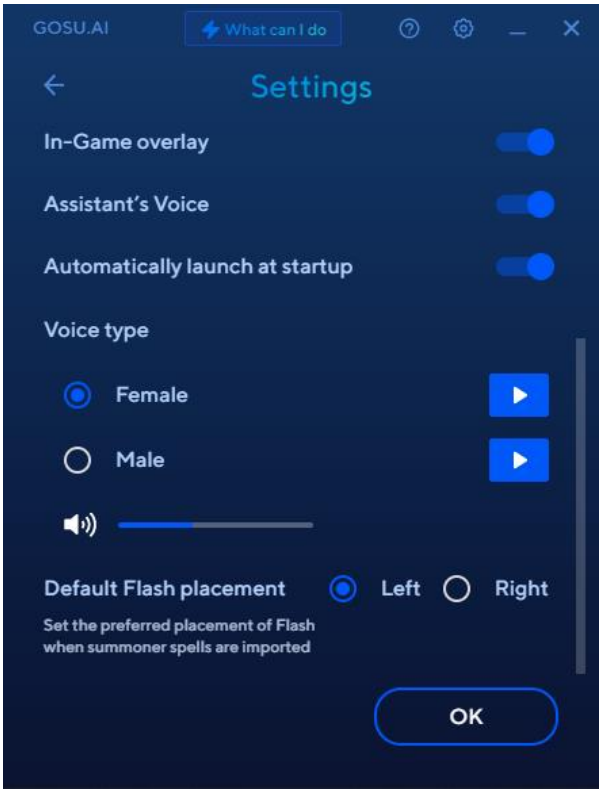

Рисунок 3.6 – Сторінка налаштувань

На даній сторінці можена налаштувати такі параметри додатку як, гучність голосових порад та ін. Сторінка можливостей виглядає наступним чином (рис. 3.7.):

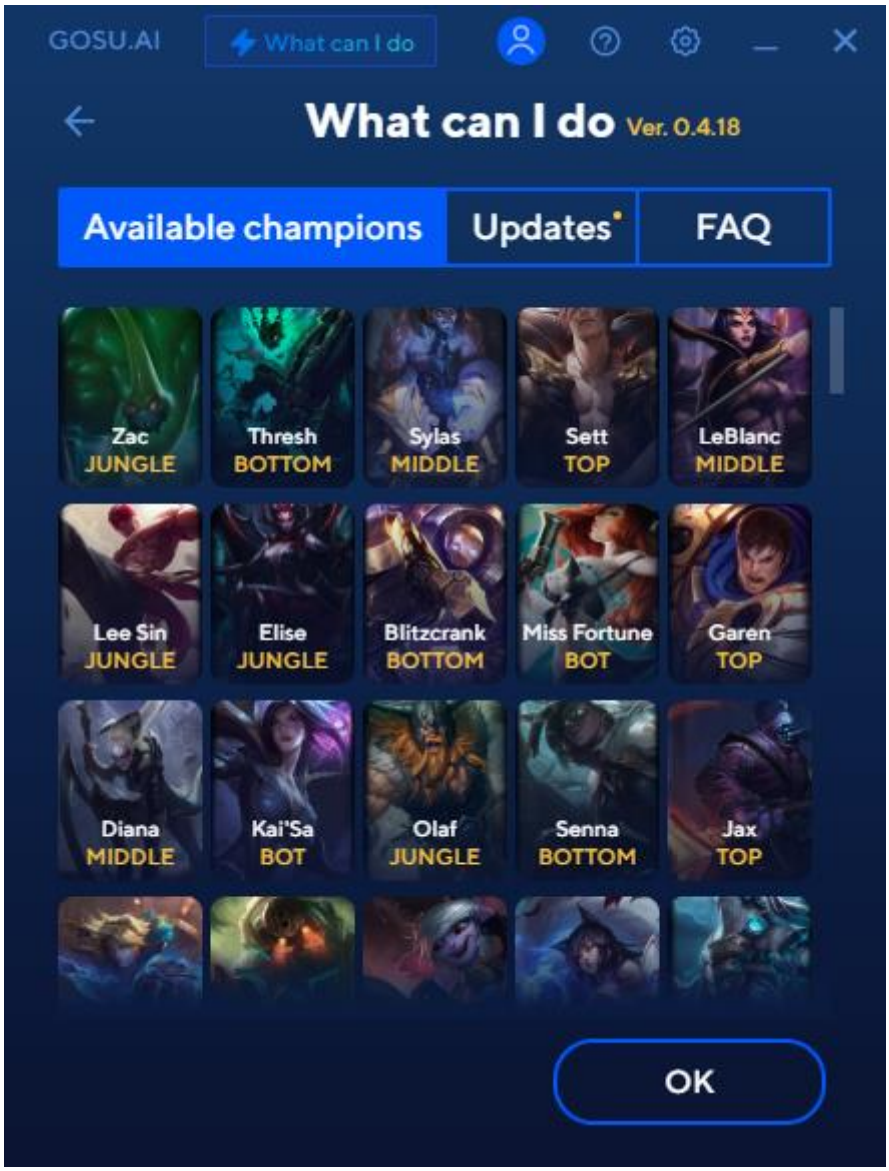

Рисунок 3.7 – Сторінка можливостей додатку

На цій сторінці доступний список чемпіонів, для яких додаток має поради, сторінку оновлень (рис. 3.8.), та допомоги (рис. 3.9.).

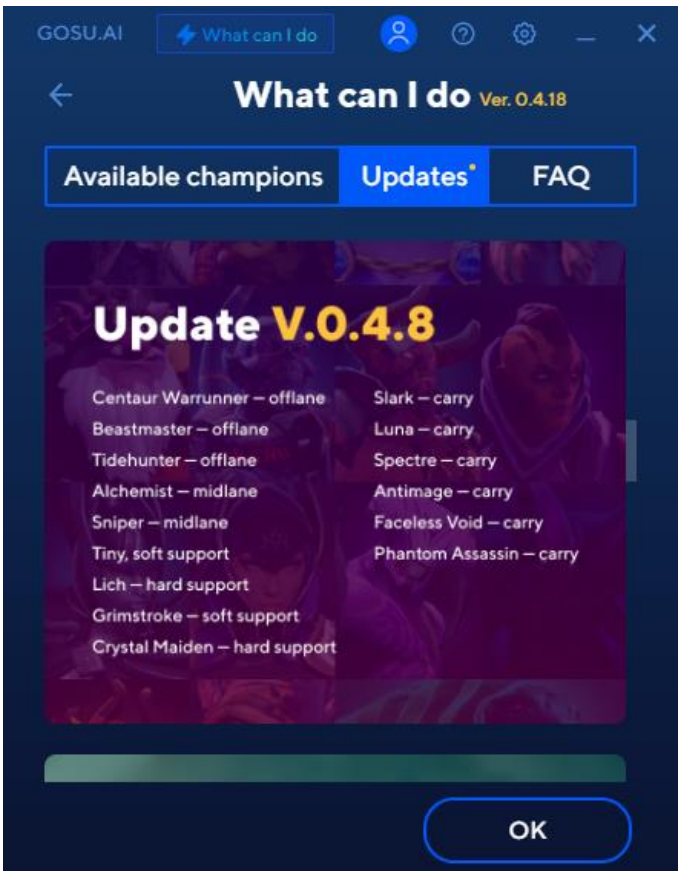

Рисунок 3.8 – Сторінка оновлень

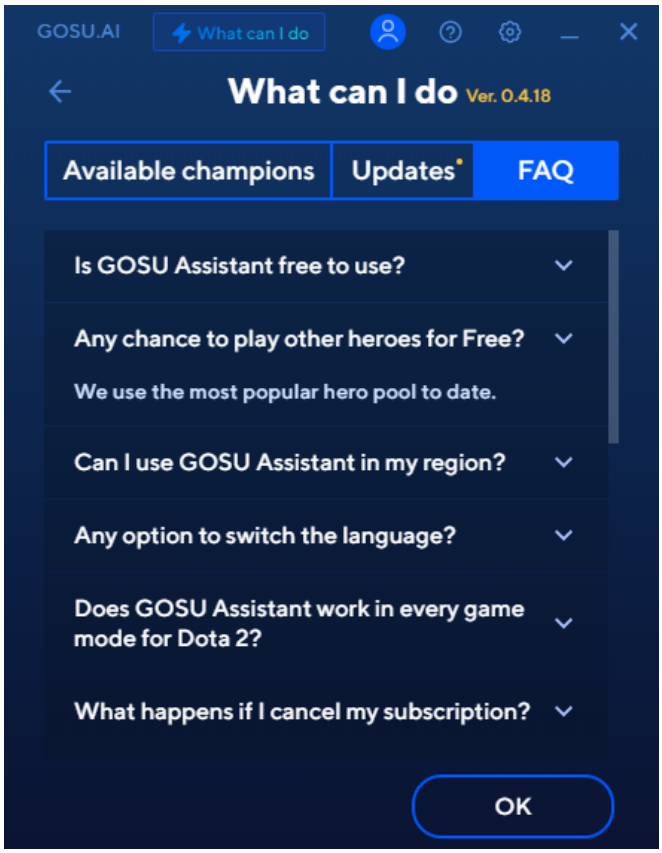

Рисунок 3.9 – Сторінка допомоги

Після запуску гри основна сторінка змінюється (зникає кнопка запуску гри та з’являється список доступних чемпіонів (рис. 3.10.).

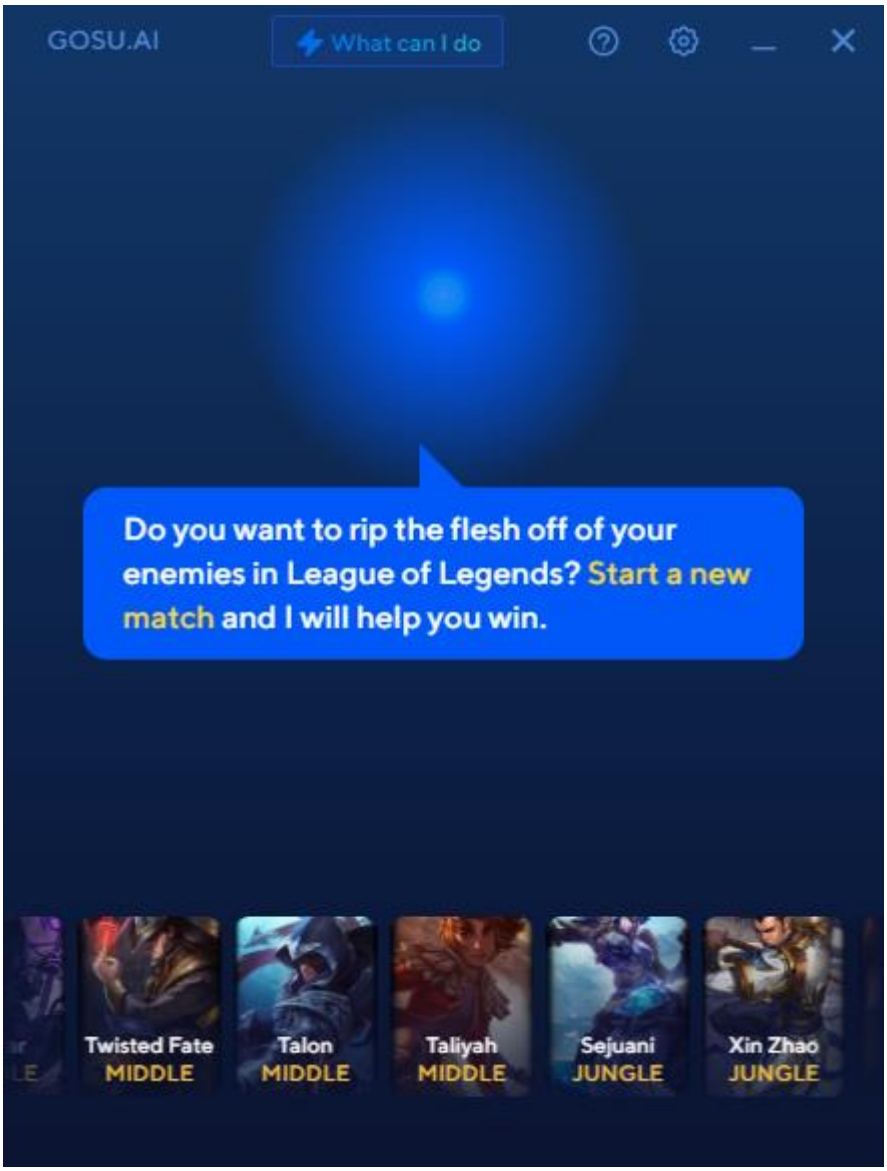

Рисунок 3.10 – Головна сторінка після запуску гри

У самій грі з’являється оверлей, на якому відображаються субтитри до голосових порад (рис. 3.11.).

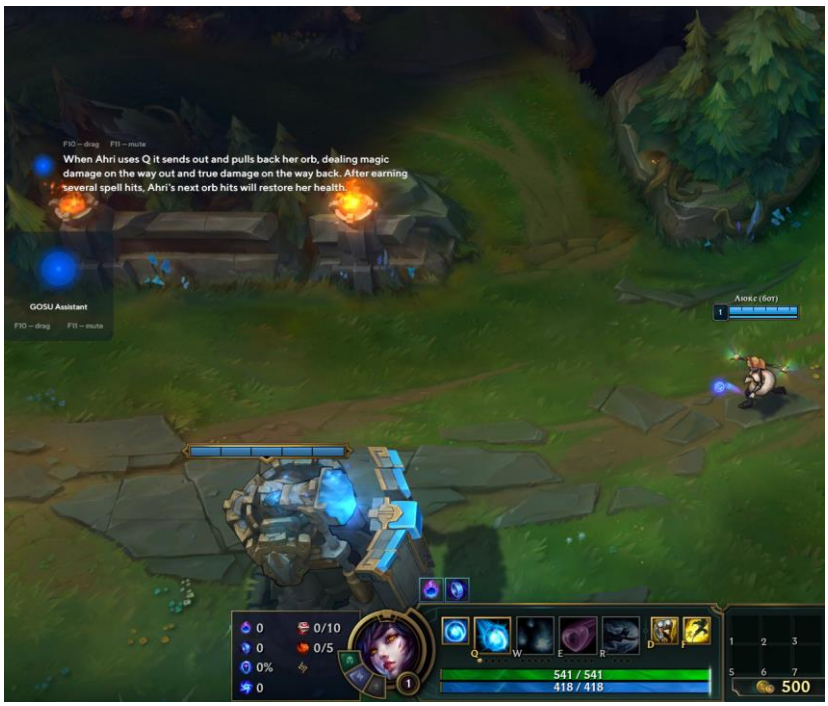

Рисунок 3.11 – Оверлей із субтитрами

При натисканні на F11 можна вимкнути голос асистента, а при натисканні F10 – перейти у режим редагування (рис 3.12) та змінити положення елементів оверлея.

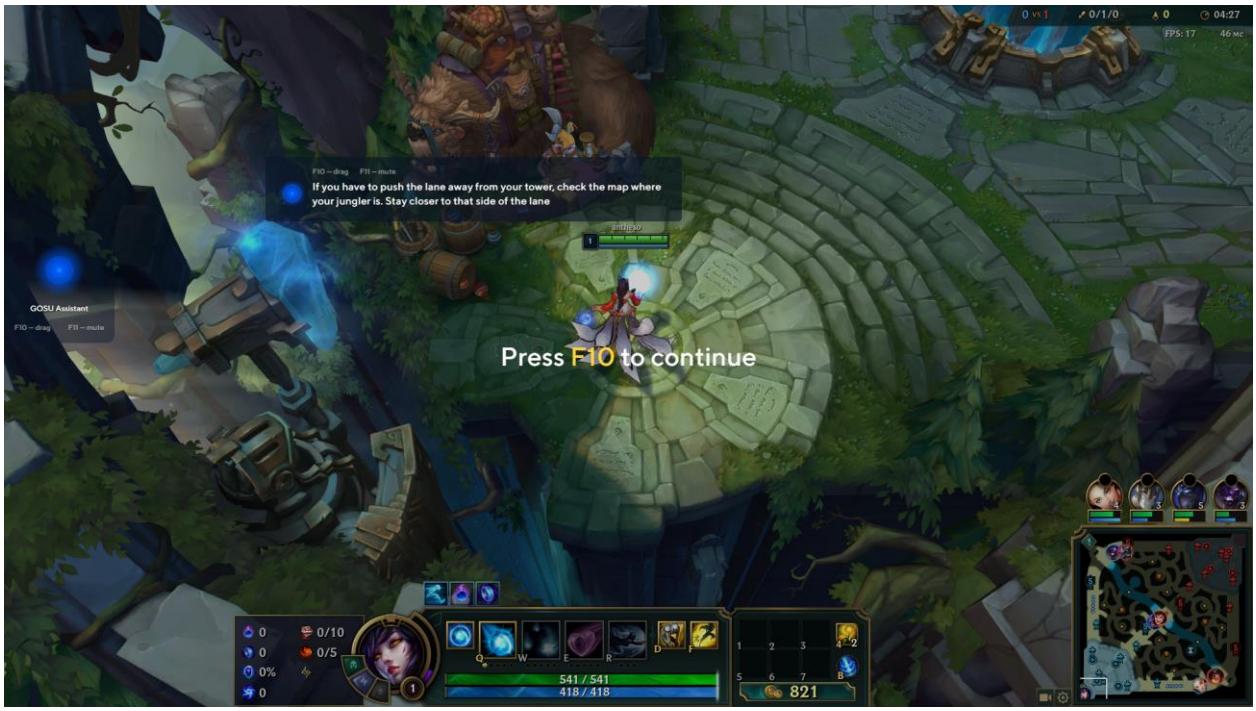

Рисунок 3.12 – Оверлей у режимі редагування

3.3. Панель адміністрування голосових порад

У проекті використовується панель адміністрування Django. Головна сторінка панелі адміністрування виглядає наступним чином (рис. 3.13):

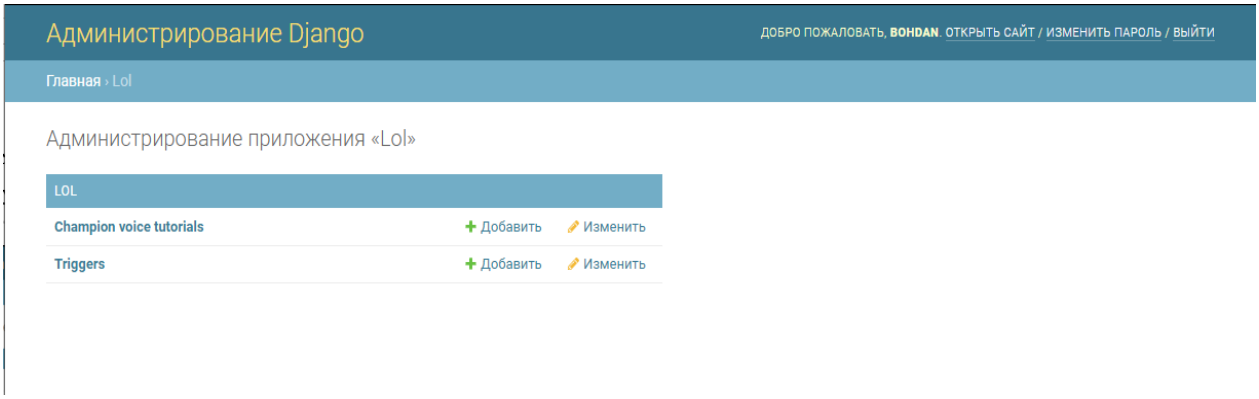

Рисунок 3.13 – Головна сторінка панелі адміністрування

На сторінці Champion voice tutorials (рис. 3.14) можна створити новий обрати посібник або редагувати існуючий (рис. 3.15):

Рисунок 3.14 – Сторінка посібника

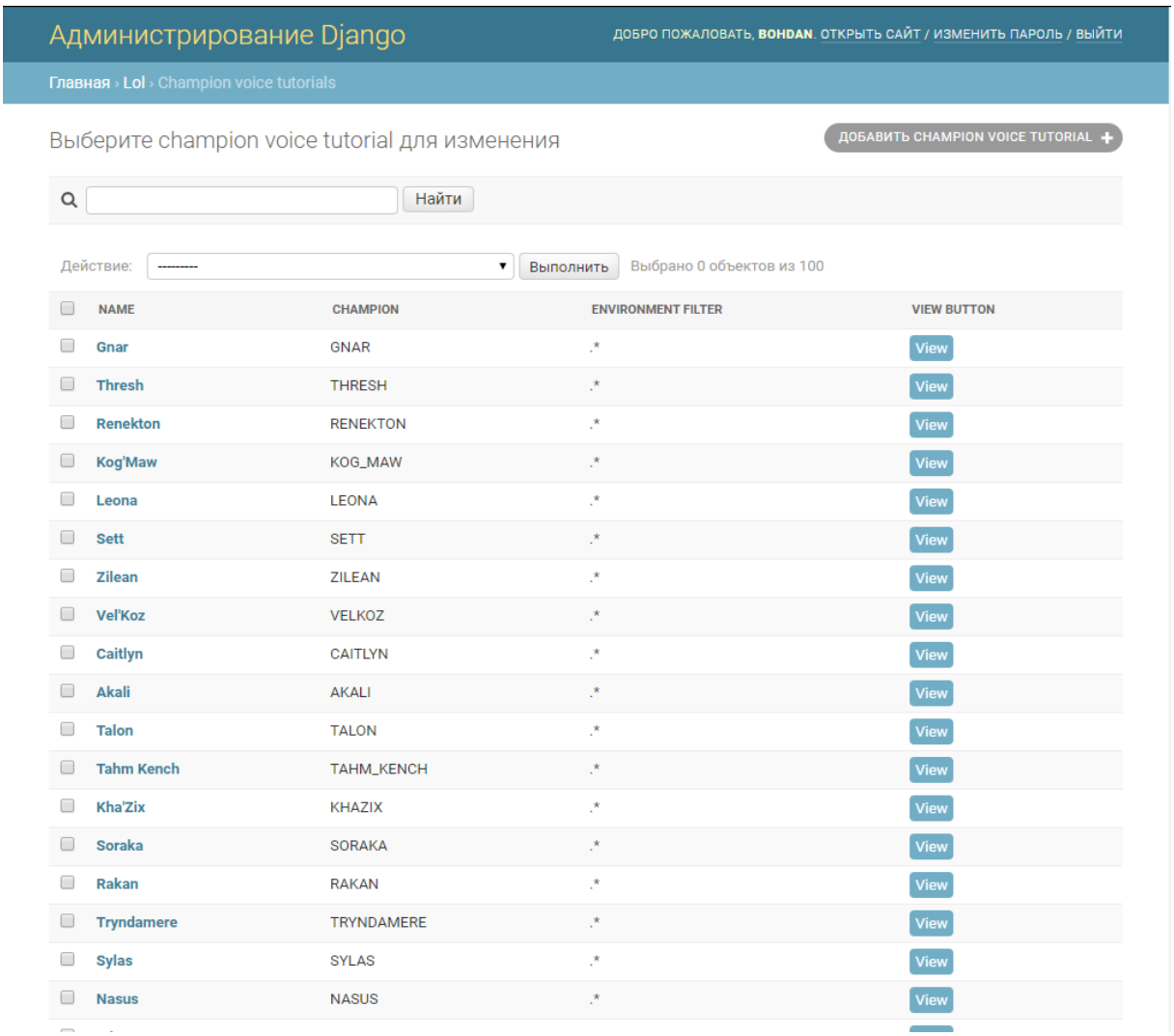

Рисунок 3.15 – Сторінка Champion voice tutorials

На сторінці тріггера (рис. 3.16) є можливість його редагувати. Аудіо файл буде згенеровано автоматично після збереження.

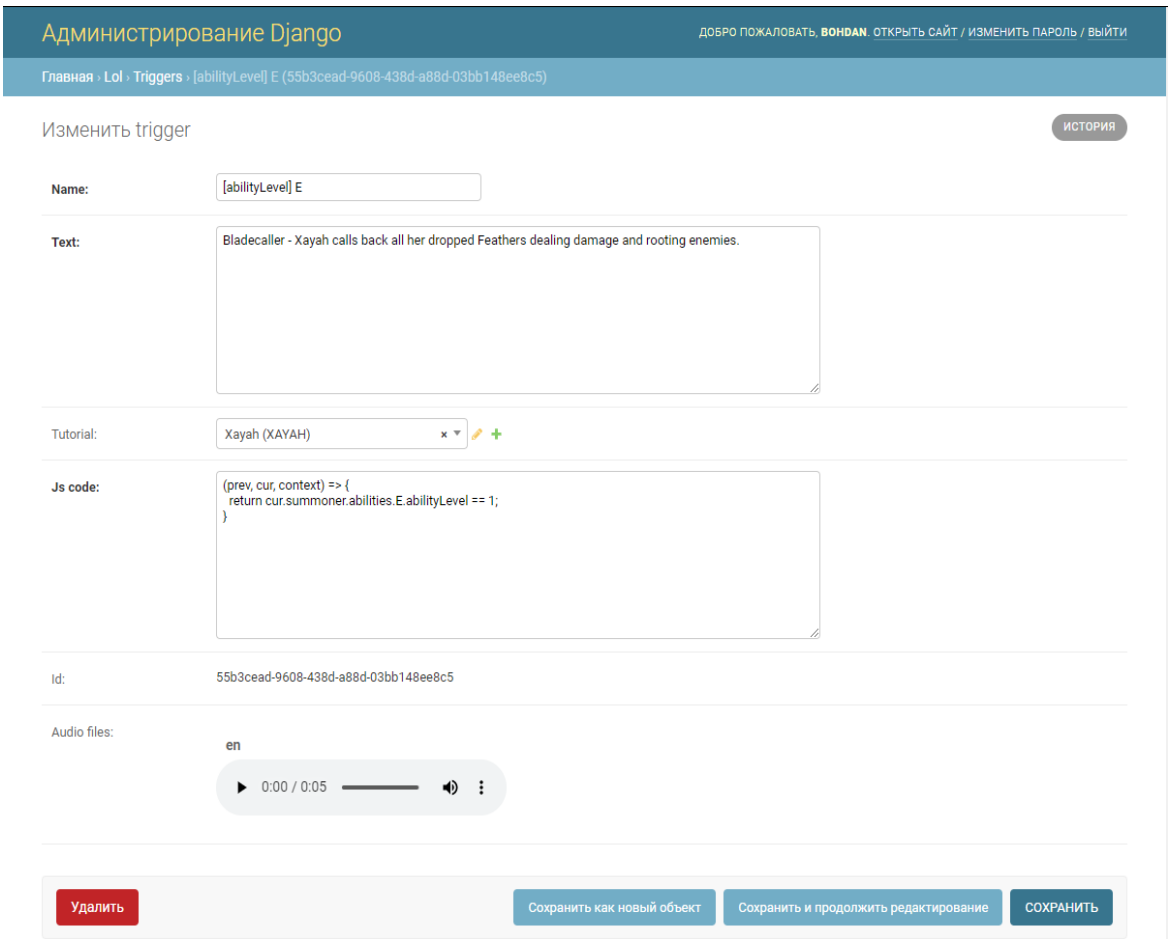

Рисунок 3.16 – Сторінка редагування тріггера

В результаті роботи було створено функціонуючий настільний додаток, який може надавати користувачу поради для кращої гри, а також адміністративну панель, за допомогою якої можна налаштовувати та адмініструвати поради.

ВИСНОВКИ

Під час виконання випускної роботи було виконане наступне:

1. було зроблений огляд сучасної літератури за обраною тематикою роботи та виконана поставка задач щодо розробки;
2. на основі літературного огляду було обрано у якості інструментів для розробки мову програмування JavaScript та бібліотеку Electron.js через простоту використання;.
3. для створення панелі адміністрування було обрано мову програмування Python та фреймворк Django тому що вони надають готовий функціонал для розв’язання задач подібного типу;
4. для розповсюдження та інсталяції додатку використовується NSIS web installer, тому що він надає можливість кастомізації та швидкої інсталяції додатку.

Результатом роботи є розроблений додаток-асистент для гри League of Legends. Знайти та завантажити додаток можна на сайті компанії gosu.ai. Додаток виконує свої функції та знаходиться у вільному доступі.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Alessandro Benoit. NW.js Essentials. – Packt Publishing Ltd, 2015-05-25. – 192 с. – ISBN 9781785287008.

2. Ж. Бланшет, М. Саммерфилд. Qt 4: Программирование GUI на C++. 2-е дополненное издание. – М.: Кудиц-пресс, 2008. – 736 с. – ISBN 978-5-91136-059-7.

3. GTK Reference Manual [Електронний ресурс]. – Режим доступу: URL: https://www.opennet.ru/docs/RUS/gtk-reference/.

4. NSIS Users Manual [Електронний ресурс]. – Режим доступу: URL: https://nsis.sourceforge.io/Docs/.

5. Roy Sutton. Desktop Targets // Enyo: Up and Running: Build Native-Quality Cross-Platform JavaScript Apps. – 2-nd ed.. – O'Reilly, 2015. – С. 83. – 100 с. – ISBN 9781491921203.

6. Брэд Дейли, Брендан Дейли, Калеб Дейли. Разработка веб-приложений с помощью Node.js, MongoDB и Angular: исчерпывающее руководство по использованию стека MEAN = Web Development with Node and Express. – 2-е изд.. – Санкт-Петербург: «Диалектика-Вильямс», 2020. – 656 с. – ISBN 978-5-6040044-8-7.

7. Stevens, Richard. UNIX Network Programming, Volume 2, Second Edition: Interprocess Communications. Prentice Hall, 1999. ISBN 0-13-081081-9

8. A. Sweigart. Core Python Applications Programming. – 2012. – 888 с. – ISBN 978-0-1326782-0-9.

9. [Електронний ресурс] https://www.ssa-frontend.com/blog/building-desktop-applications-with-electron

Додаток А

Файл electron/main.ts

```
import {
  app,
  BrowserWindow,
  Tray,
  Menu,
  ipcMain,
  globalShortcut,
  Event
} from 'electron';
import * as path from
'path';
import * as url from
'url';
import * as keytar
from 'keytar';
import * as log from
'electron-log';
import { machineIdSync
} from 'node-machine-
id';

import {
BehaviorSubject } from
'rxjs';

import {
  LolConnector,
  AppLogger,
  AppUpdater,
  DeepLogin,
  AppWidget,
  Analytic
} from './utils';

const gameLaunchState$
= new
BehaviorSubject(false)
,
  analytic = new
Analytic(),
  iconPath =
app.isPackaged ?
path.join(app.getAppPa
th(), '..', 'build',
'icon.ico') :
path.join(app.getAppPa
th(), 'build',
'icon.ico');

let win:
BrowserWindow,
  isQuiting = false,
  tray: Tray,
  updater: AppUpdater,
  logger: AppLogger,
  deepLogin:
DeepLogin,
  widget: AppWidget,
  connector:
LolConnector;

const prepareGlobal =
() => {
    global['version']
= app.getVersion();

global['machineId'] =
machineIdSync();
    global['keytar'] =
keytar;
    global['reqToLol']
= (apiUrl: string) =>
connector.makeRequestT
oLol(apiUrl);

global['reqToMatch'] =
(apiUrl: string) =>
connector.makeRequestT
oMatch(apiUrl);
  },

  startLolConnector =
() => {

connector.on('gameLaun
chState', (state:
boolean) => {
      if
(win.webContents) {

gameLaunchState$.next(
state);

win.webContents.send('
gameLaunchState',
state);
      }
    });

    connector.start();
  },

  createWindow = () =>
{
    prepareGlobal();

    win = new
BrowserWindow({
      width: 250,
      height: 300,
      webPreferences:
{

nodeIntegration: true,

backgroundThrottling:
false
      },
      icon: iconPath,
      frame: false,
      resizable: false
    });


analytic.sentAnalytic(
'app-start');

    logger = new
AppLogger(log);
    updater = new
AppUpdater(win,
logger);
    deepLogin = new
DeepLogin(win,
logger);
    widget = new
AppWidget(win);
    connector = new
LolConnector(logger);


startLolConnector();
    tray = new
Tray(iconPath);


tray.setContextMenu(Me
nu.buildFromTemplate([
      {
        label: 'Show
App', click: () =>
win.show()
      },
      {
        label: 'Quit',
click: () => {
          isQuiting =
true;
          app.quit();
        }
      }
    ]));

    tray.on('double-
click', () =>
win.show());
```

```
globalShortcut.register('F11', () => {
      if (win.webContents.isDevToolsOpened()) {
        win.webContents.closeDevTools();
      } else {
        win.webContents.openDevTools();
      }
    });

    win.loadURL(
      url.format({
        pathname: path.join(__dirname, `/../../dist/index.html`),
        protocol: 'file',
        slashes: true
      })
    );

    win.webContents.setZoomFactor(1);

    deepLogin.deepLink = process.argv.slice(1).toString();

    win.on('closed', () => {
      connector.stop();
      win = null;
    });

    win.on('minimize', (event: Event) => {
      event.preventDefault();
      win.hide();
    });

    win.on('close', (event: Event) => {
      if (!isQuiting) {
        event.preventDefault();
        win.hide();
        event.returnValue = false;
      }
    });

    win.webContents.on('crashed', (event: Event) => {
      analytic.sentAnalytic('app-crash', event);
      app.relaunch();
      app.exit();
    });

    win.on('unresponsive', () => {
      analytic.sentAnalytic('app-crash');
      app.relaunch();
      app.exit();
    });

    process.on('uncaughtException', (event: Error) => {
      analytic.sentAnalytic('app-crash', event);
      app.relaunch();
      app.exit();
    });

    updater.checkForUpdatesAndNotify();
  },

  prepareApp = () => {
    app.on('second-instance', (_, argv: string[]) => {
      deepLogin.deepLink = argv.slice(1).toString();
      if (win) {
        if (win.isMinimized()) {
          win.restore();
        }
        win.focus();
      }
    });

    app.on('ready', createWindow);

    app.on('activate', () => {
      if (win === null) {
        createWindow();
      }
    });

    app.on('before-quit', () => isQuiting = true);

    app.setAsDefaultProtocolClient('gosu-desktop');

    app.commandLine.appendSwitch('autoplay-policy', 'no-user-gesture-required');

    app.on('open-url', (event: Event, urlVal: string) => {
      event.preventDefault();
      deepLogin.deepLink = urlVal;
    });

    app.on('will-quit', () => {
      analytic.sentAnalytic('app-quit');
      globalShortcut.unregisterAll();
    });

    ipcMain.on('getGameLaunchState', () => win.webContents.send('gameLaunchState', gameLaunchState$.getValue()));
  };

if (!app.requestSingleInstanceLock()) {
  app.quit();
} else {
  prepareApp();
}
```

Файл electron/widget.ts

```
import {
BrowserWindow, ipcMain
} from 'electron';
import * as ioHook
from 'iohook';
import * as activeWin
from 'active-win';
import * as path from
'path';

import { Subscription,
timer, from } from
'rxjs';
import { switchMap }
from 'rxjs/operators';

import { processNames
} from './lol-
connector';

export class AppWidget
{

  public win:
BrowserWindow;
  private
ignoreMouseEvents =
true;
  private intervalSub:
Subscription;
  private
isWindowActive =
false;

  get isActive():
boolean {
    return
!this.ignoreMouseEvent
s;
  }

  constructor(
    private parentWin:
BrowserWindow
  ) {
    ioHook.on('keyup',
(evt) => {
      if (evt.rawcode
=== 121) {

this.ignoreMouseEvents
=
!this.ignoreMouseEvent
s;

this.win.webContents.s
end('widgetActive',
this.isActive);


this.win.setIgnoreMous
eEvents(this.ignoreMou
seEvents);

        if
(!this.isActive) {

this.win.setFocusable(
false);

this.win.blur();
        } else {

this.win.setFocusable(
true);
        }
      }

      if (evt.rawcode
=== 120) {

this.win.webContents.s
end('playerMute');
      }
    });


ipcMain.on('widgetOpen
', (_, routeUrl:
string) =>
this.open(routeUrl));

ipcMain.on('widgetClos
e', () =>
this.close());

this.proxyToWindow('wi
dgetMessage');

this.proxyToWindow('gl
adosActive');
  }

  private
open(routeUrl:
string): void {
    if (this.win) {
      this.close();
    }


this.checkWinActivity(
);

    this.win = new
BrowserWindow({
      parent:
this.parentWin,
      show: false,
      frame: false,
      resizable:
false,
      fullscreen:
true,
      transparent:
true,
      modal: true,
      webPreferences:
{

nodeIntegration: true
      },
      alwaysOnTop:
true
    });


this.win.on('closed',
() => {
      this.win = null;

this.isWindowActive =
false;

this.ignoreMouseEvents
= true;
      ioHook.stop();

      if
(this.intervalSub) {

this.intervalSub.unsub
scribe();
      }
    });


this.win.once('ready-
to-show', () => {

this.parentWin.hide();

ioHook.start(false);
    });


this.win.loadURL(`file
://${path.join(__dirna
me,
```

```
`/../../../dist/index.
html`)}#${routeUrl}`);

this.win.setIgnoreMous
eEvents(this.ignoreMou
seEvents);
    //
this.win.webContents.o
penDevTools();
  }

  private close():
void {
    if (this.win) {

this.win.close();
    }
  }

  private
checkWinActivity():
void {
    this.intervalSub =
timer(0, 2000).pipe(
      switchMap(() =>
from(activeWin()))

).subscribe((window:
any) => {
      if (!this.win) {
        return;
      }

      const
isWinActive =
this.parentWin.isFocus
ed()
        ? false
        :
[...processNames,
'gosu.ai.exe',
'electron.exe'].includ
es(window &&
window.owner.name);

      if
(this.isWindowActive
!== isWinActive) {
        if
(isWinActive) {
          if (window
&& window.bounds &&
JSON.stringify(window.
bounds) !==
JSON.stringify(this.wi
n.getBounds())) {

this.win.setBounds(win
dow.bounds);
          }

this.win.show();
        } else {

this.win.hide();
        }
      }


this.isWindowActive =
isWinActive;
    });
  }

  private
proxyToWindow(channel:
string): void {

ipcMain.on(channel,
(_event, ...args: any)
=> {
      if (!this.win) {
        return;
      }


this.win.webContents.s
end(channel, ...args);
    });
  }

}
```

Файл lol-connector.ts

```
import { EventEmitter
} from 'events';

import * as path from
'path';
import * as
processList from
'process-list';
import * as
LockfileParser from
'lol-lockfile-parser';
import * as request
from 'request';
import * as chokidar
from 'chokidar';

import { AppLogger }
from './logger';

export const
processNames =
['League of
Legends.exe'];
const lockfileParser =
new LockfileParser();

interface
ILockfileData {
  protocol: 'http' |
'https';
  port: number;
  password: string;
}

interface
ILolCredentials
extends ILockfileData
{
  username: string;
  address: string;
}

const ENABLE_LOGGING =
true;
const lolMatchApiURL =
'https://127.0.0.1:299
9';

export class
LolConnector extends
EventEmitter {
  private dirPath:
string;
  private
processWatcher:
NodeJS.Timeout;
  private
lockfileWatcher: any;
  private
doesClientLaunched =
false;
  private
lolCredentials:
ILolCredentials;
  private
launcherWorking =
false;

  constructor(
    private logger:
AppLogger
  ) {
    super();
  }

  public start(): void
{
```

```
this.initProcessWatcher();
  }

  public stop(): void {

this.clearProcessWatcher();

this.clearLockfileWatcher();
  }

  public makeRequestToLol(url: string): Promise<any> {
    return new Promise((resolve, reject) => {
      if (!this.lolCredentials) {
        reject('No LOL credentials!');
        return;
      }

      if (!this.launcherWorking) {

reject('Launcher doesnt opened!');
        return;
      }

      if (ENABLE_LOGGING) {

this.logger.logInfo('makeRequestToLol');
      }

      request({
        uri: `https://${this.lolCredentials.address}:${this.lolCredentials.port}${url}`,
        method: 'GET',

rejectUnauthorized: false,
        headers: {

'Authorization': 'Basic ' + new Buffer(this.lolCredentials.username + ':' + this.lolCredentials.password).toString('base64')
        }
      }, (error: any, response: request.Response, body: any) =>

this.processResponce(error, response, body).then(data => resolve(data), err => reject(err)));
    });
  }

  public makeRequestToMatch(url: string): Promise<any> {
    return new Promise((resolve, reject) => {
      if (!this.launcherWorking) {

reject('Launcher doesnt opened!');
        return;
      }

      if (ENABLE_LOGGING) {

this.logger.logInfo('makeRequestToMatch');
      }

      request({
        uri: `${lolMatchApiURL}${url}`,
        method: 'GET',

rejectUnauthorized: false
      }, (error: any, response: request.Response, body: any) =>

this.processResponce(error, response, body).then(data => resolve(data), err => reject(err)));
    });
  }

  private processResponce(error: any, response: request.Response, body: any): Promise<any> {
    return new Promise((resolve, reject) => {
      const clearBody = JSON.parse(body && body.trim());

      if (response && response.statusCode === 200) {
        if (ENABLE_LOGGING) {

this.logger.logInfo(clearBody);
        }


resolve(clearBody);
      } else {
        if (ENABLE_LOGGING) {
          if (error) {

this.logger.logInfo(error);
          }

this.logger.logInfo(clearBody);
        }


reject(clearBody);
      }
    });
  }

  private getLCUPathFromProcess(): Promise<string> {
    return new Promise(resolve => {
      const snapshot = processList.snapshot;
      const INSTALL_REGEX = /"--install-directory=(.*?)"/;
      const launcherProcessName = 'LeagueClientUx.exe';


snapshot('cmdline', 'name').then((list: Record<string, string>[]) => {
        const isGameRunning =
```

```
list.some(item =>
processNames.includes(
item.name));
        if
(this.doesClientLaunch
ed !== isGameRunning)
{

this.emit('gameLaunchS
tate', isGameRunning);

this.doesClientLaunche
d = isGameRunning;
        }

        return
list.find(item =>
item.name ===
launcherProcessName);
      }).then((proc:
Record<string,
string>) => {

this.launcherWorking =
!!proc;

        if
(this.dirPath ||
!this.launcherWorking)
{
          resolve();

          return;
        }
        const parts =
proc.cmdline.match(INS
TALL_REGEX) || [];


resolve(parts[1]);
      });
    });
  }

  private async
initProcessWatcher():
Promise<void> {
    try {
      const lcuPath =
await
this.getLCUPathFromPro
cess();
      if (lcuPath) {
        this.dirPath =
lcuPath;

this.initLockfileWatch
er();
      }
      if
(!this.processWatcher)
{

this.processWatcher =
setInterval(this.initP
rocessWatcher.bind(thi
s), 1000);
      }
    } catch (e) {
      return
console.error(e);
    }
  }

  private
clearProcessWatcher():
void {

clearInterval(this.pro
cessWatcher);
  }

  private
initLockfileWatcher():
void {
    if
(this.lockfileWatcher)
{
      return;
    }

    const lockfilePath
=
path.join(this.dirPath
, 'lockfile');
    if
(ENABLE_LOGGING) {

this.logger.logInfo(lo
ckfilePath);
    }

this.lockfileWatcher =
chokidar.watch(lockfil
ePath, {
disableGlobbing: true
});


this.lockfileWatcher.o
n('add', () =>
this.onFileCreated(loc
kfilePath));

this.lockfileWatcher.o
n('change', () =>
this.onFileCreated(loc
kfilePath));

this.lockfileWatcher.o
n('unlink', () =>
this.onFileRemoved());
  }

  private
clearLockfileWatcher()
: void {
    if
(this.lockfileWatcher)
{

this.lockfileWatcher.c
lose();
    }
  }

  private
onFileCreated(lockPath
: string): void {

lockfileParser.read(lo
ckPath).then((data:
ILockfileData) => {
      const result:
ILolCredentials = {
        protocol:
data.protocol,
        address:
'127.0.0.1',
        port:
data.port,
        username:
'riot',
        password:
data.password
      };

this.lolCredentials =
result;


this.emit('connect');
    }).catch((e:
Error) =>
console.error(e));
  }

  private
onFileRemoved(): void
{

this.emit('disconnect'
);
  }

}
```

Файл angular/main.ts

```
import { enableProdMode } from '@angular/core';
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';

import { AppModule } from './app/app.module';
import { environment } from './environments/environment';

if (environment.production) {
  enableProdMode();
}

platformBrowserDynamic().bootstrapModule(AppModule)
  .catch(err => console.error(err));
```

Файл angular/index.html

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">

<title>GosuDesktop</title>
  <base href="./">

  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
  <script>
    var global = global || window;
    var Buffer = Buffer || [];
    var process = process || {
      env: { DEBUG: undefined },
      version: []
    };
  </script>
</head>
<body>
  <gosu-root>
    <div class="preloader">
      <div class="preloader__content">
        <div class="preloader__light"></div>
        <p class="preloader__title">GOSU — AI Assistant</p>
        <div class="preloader__wrapper">
          <span class="preloader__status">Launching</span>
          <span class="dot">.</span>
          <span class="dot">.</span>
          <span class="dot">.</span>
        </div>
      </div>
    </div>
    <style>
      .preloader {
        width: 100%;
        height: 100%;
        background: linear-gradient(180deg, #193869 3.72%, #0E1C3F 76.08%, #0D1533 100%);
        display: flex;
        align-items: center;
        justify-content: center;
      }

.preloader__light {
        background: radial-gradient(50% 50% at 50% 50%, #5DBBEE 0%, #0062FF 15.62%, rgba(13, 21, 51, 0) 100%);
        width: 140px;
        height: 140px;
        margin-bottom: 10px;
      }

.preloader__title {
        font-size: 16px;
        line-height: 19px;
        color: #699FF4;
        text-shadow: 0px 1px 1px $black;
        margin: 0 0 35px;
      }

.preloader__status {
        background: linear-gradient(90deg, #0062FF 0%, #53E19E 100%), #0062FF;
        background-clip: text;
        -webkit-background-clip: text;
        -webkit-text-fill-color: transparent;
        font-size: 16px;
        line-height: 20px;
        margin: 0;
        text-align: center;
      }

.preloader__wrapper {
        display: flex;
        justify-content: center;
```

```
        align-items:
flex-end;
      }
      .dot {
        color:
#53E19E;
        animation:
loader 1.5s infinite
ease-in-out;
      }
      .dot:nth-
child(2) {
        animation-
delay: 0.5s;
      }
      .dot:nth-
child(3) {
        animation-
delay: 0.25s;
      }
      @keyframes
loader {
        0% {
          opacity: 1;
        }
        50% {
          opacity: 0;
        }
        100% {
          opacity: 1;
        }
      }
    </style>
  </gosu-root>
</body>
</html>
```

Файл angular/styles.scss

```
// Font faces

@font-face {
  font-family:
'CircularPro';
  font-style: normal;
  font-weight: normal;
  src:
url('./assets/fonts/ci
rcular-pro-book.eot');
  src:
url('./assets/fonts/ci
rcular-pro-
book.eot?#iefix')
format('embedded-
opentype'),
url('./assets/fonts/ci
rcular-pro-
book.woff2')
format('woff2'),
url('./assets/fonts/ci
rcular-pro-book.woff')
format('woff');
}

@font-face {
  font-family:
'CircularPro';
  font-style: normal;
  font-weight: 700;
  src:
url('./assets/fonts/ci
rcular-pro-bold.eot');
  src:
url('./assets/fonts/ci
rcular-pro-
bold.eot?#iefix')
format('embedded-
opentype'),
url('./assets/fonts/ci
rcular-pro-
bold.woff2')
format('woff2'),
url('./assets/fonts/ci
rcular-pro-bold.woff')
format('woff');
}

@font-face {
  font-family:
'Graphik Regular';
  src:
url('./assets/fonts/Gr
aphik-Regular-
Web.eot');
  src:
url('./assets/fonts/Gr
aphik-Regular-
Web.eot#iefix')
format('embedded-
opentype'),
url('./assets/fonts/Gr
aphik-Regular-
Web.woff2')
format('woff2'),
url('./assets/fonts/Gr
aphik-Regular-
Web.woff')
format('woff'),
url('./assets/fonts/Gr
aphik-Regular-
Web.ttf')
format('truetype');
  font-weight: normal;
  font-style: normal;
}

@font-face {
  font-family:
'Graphik Semibold';
  src:
url('./assets/fonts/Gr
aphik-Semibold-
Web.eot');
  src:
url('./assets/fonts/Gr
aphik-Semibold-
Web.eot#iefix')
format('embedded-
opentype'),
url('./assets/fonts/Gr
aphik-Semibold-
Web.woff2')
format('woff2'),
url('./assets/fonts/Gr
aphik-Semibold-
Web.woff')
format('woff'),
url('./assets/fonts/Gr
aphik-Semibold-
Web.ttf')
format('truetype');
  font-weight: normal;
  font-style: normal;
}

@font-face {
  font-family:
'Graphik Super';
  src:
url('./assets/fonts/Gr
aphik-Super-Web.eot');
  src:
url('./assets/fonts/Gr
aphik-Super-
Web.eot#iefix')
format('embedded-
opentype'),
url('./assets/fonts/Gr
aphik-Super-
Web.woff2')
format('woff2'),
url('./assets/fonts/Gr
aphik-Super-Web.woff')
format('woff'),
url('./assets/fonts/Gr
aphik-Super-Web.ttf')
format('truetype');
  font-weight: normal;
  font-style: normal;
}

@font-face {
```

```
  font-family:
'TTNorms';
  src:
url('./assets/fonts/37
8659_0_0.eot');
  src:
url('./assets/fonts/37
8659_0_0.eot?#iefix')
format('embedded-
opentype'),
url('./assets/fonts/37
8659_0_0.woff2')
format('woff2'),
url('./assets/fonts/37
8659_0_0.woff')
format('woff'),
url('./assets/fonts/37
8659_0_0.ttf')
format('truetype');
  font-weight: 700;
  font-style: normal;
}

@font-face {
  font-family:
'TTNorms';
  src:
url('./assets/fonts/TT
Norms-Medium.eot');
  src:
url('./assets/fonts/TT
Norms-
Medium.eot?#iefix')
format('embedded-
opentype'),
url('./assets/fonts/TT
Norms-Medium.woff2')
format('woff2'),
url('./assets/fonts/TT
Norms-Medium.woff')
format('woff'),
url('./assets/fonts/TT
Norms-Medium.ttf')
format('truetype');
  font-weight: normal;
  font-style: normal;
}

// App global styles

html,
body {
  height: 100%;
  font-family:
'TTNorms', sans-serif;
  font-variant-
ligatures: none;
  text-rendering:
optimizeLegibility;
  scroll-behavior:
smooth;
  margin: 0;
  letter-spacing:
0.004em;

  .ps--active-y >
.ps__rail-y,
  .ps--active-x >
.ps__rail-x {
    background-color:
transparent
!important;
  }
}

input,
button {
  font-family:
'TTNorms', sans-serif;
  font-variant-
ligatures: none;
}

* {
  margin: 0;
  padding: 0;
  box-sizing: border-
box;
}

.text-yellow {
  color: #FFD300;
}

.item-image {
  width: 20px;
  height: 20px;
  display: inline-
block;
  background-size:
100%;
  background-position:
center;
  vertical-align: sub;

  &_large {
    width: 40px;
    height: 40px;
  }
}
```

Файл angular/app.module.ts

```
import { BrowserModule
} from
'@angular/platform-
browser';
import { NgModule }
from '@angular/core';
import {
HttpClientModule }
from
'@angular/common/http'
;
import { FormsModule }
from '@angular/forms';

import {
NgxElectronModule }
from 'ngx-electron';

import { AppRouting }
from './app.routing';
import { AppComponent
} from
'./app.component';
import { CoreModule }
from
'./core/core.module';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    AppRouting,
    NgxElectronModule,
    CoreModule,
    HttpClientModule,
    FormsModule
  ],
  bootstrap: [
    AppComponent
  ]
})
export class AppModule
{ }
```

Файл angular/app.component.ts

```
import {
  Component,
ChangeDetectionStrateg
y,
  HostListener,
  AfterViewInit
} from
'@angular/core';
import {
NavigationEnd, Router
} from
'@angular/router';

import { Observable }
from 'rxjs';
import {
  filter,
  take,
  mapTo,
  tap
} from
'rxjs/operators';

import {
ElectronService } from
'ngx-electron';

import { HitService }
from '@gosu-
core/services';

@Component({
  selector: 'gosu-
root',
  templateUrl:
'./app.component.html'
,
  styles: [`
    :host {
      display: block;
      height: 100%;
    }
  `],
  changeDetection:
ChangeDetectionStrateg
y.OnPush
})
export class
AppComponent
implements
AfterViewInit {

  public
isRouteResolved:
Observable<boolean> =
this.router.events.pip
e(
    filter(e => e
instanceof
NavigationEnd),
    take(1),
    mapTo(true)
  );
  public
updatesLoading =
false;

  constructor(
    private router:
Router,
    private hit:
HitService,
    electron:
ElectronService
  ) {
    if
(!electron.remote) {
      return;
    }

console.log(electron.r
emote.getGlobal('logs'
));

electron.ipcRenderer.o
n('message', (_evt,
info) =>
console.log(info));

electron.ipcRenderer.s
end('getUpdateStatus')
;

electron.ipcRenderer.o
n('updateStatus', (_,
status) =>
this.updatesLoading =
status);
  }


@HostListener('click',
['$event']) public
onClick(evt:
MouseEvent): void {

this.hit.click(evt);
  }

  public
ngAfterViewInit():
void {

this.router.events.pip
e(
      filter((event)
=> event instanceof
NavigationEnd),
      tap(() =>
this.hit.pageview())
    ).subscribe();
  }

}
```

Файл angular/app.component.html

```
<gosu-preloader
[updating]="updatesLoa
ding"

*ngIf="updatesLoading
|| !(isRouteResolved |
async); else outlet">
</gosu-preloader>
<gosu-busy></gosu-
busy>

<ng-template #outlet>
  <router-
outlet></router-
outlet>
</ng-template>
```

Файл angular/app.routing.ts

```
import { NgModule }
from '@angular/core';

import { Routes,
RouterModule } from
'@angular/router';

import {
LayoutComponent } from
'@gosu-
core/components';
```

```
const routes: Routes =
[
  {
    path: '',
    component:
LayoutComponent,
    children: [
      {
        path: '',
        loadChildren:
'./+glados/glados.modu
le#GladosModule'
      },
      {
        path:
'paywall',
        loadChildren:
'./+paywall/paywall.mo
dule#PaywallModule'
      },
      {
        path:
'settings',
        loadChildren:
'./+settings/settings.
module#SettingsModule'
      }
    ]
  },
  {
    path: 'widget',
    loadChildren:
'./+widget/widget.modu
le#WidgetModule'
  }
];

@NgModule({
  imports: [

RouterModule.forRoot(r
outes, {
      useHash: true
    })
  ],
  exports: [
    RouterModule
  ]
})
export class
AppRouting { }
```

Файл angular/widget.component.ts

```
import {
  Component,
  HostBinding,
  OnInit,

ChangeDetectionStrateg
y,
  ChangeDetectorRef,
  NgZone,
  ViewChild,
  ElementRef,
  AfterViewInit,
  OnDestroy,
  HostListener
} from
'@angular/core';

import {
ElectronService } from
'ngx-electron';

import {
PlayerService,
SettingsService,
IAssistantMessage }
from '@gosu-
core/services';

@Component({
  selector: 'gosu-
widget',
  templateUrl:
'./widget.component.ht
ml',
  styleUrls:
['./widget.component.s
css'],
  changeDetection:
ChangeDetectionStrateg
y.OnPush
})
export class
WidgetComponent
implements OnInit,
AfterViewInit,
OnDestroy {

  public isColumn =
false;

@HostBinding('class.ac
tive') public isActive
= false;

@ViewChild('widgetAssi
stant', { static:
false }) public
widgetAssistant:
ElementRef;

@ViewChild('widgetSubt
itles', { static:
false }) public
widgetSubtitles:
ElementRef;
  public widgetMsg =
'';
  public widgetItems =
[];

  private activeElem:
HTMLElement;

  get hasSubtitles():
boolean {
    return
this.settings.userSett
ings.subtitles;
  }

  constructor(
    private electron:
ElectronService,
    private cd:
ChangeDetectorRef,
    private playerSrv:
PlayerService,
    private zone:
NgZone,
    private settings:
SettingsService
  ) { }

  public ngOnInit():
void {
    if
(!this.electron.remote
) {
      return;
    }


this.electron.ipcRende
rer.on('widgetActive',
(_evt, isActive:
boolean) => {
      this.isActive =
isActive;
      if (!isActive) {

this.saveSettings();
      }
```

```
this.cd.detectChanges(
);
    });


this.electron.ipcRende
rer.on('gladosActive',
(_evt, status:
boolean) => {

this.zone.runTask(()
=> {

this.playerSrv.playerA
ctive$.next(status);

this.cd.detectChanges(
);
      });
    });


this.electron.ipcRende
rer.on('widgetMessage'
, (_evt, message:
IAssistantMessage) =>
{

this.zone.runTask(()
=> {
        this.widgetMsg
= message.text;
        if
(message.items &&
message.items.length)
{

this.widgetItems =
message.items;
        }

this.cd.detectChanges(
);
      });
    });
  }

  public
ngAfterViewInit():
void {
    const widget =
this.settings.widgetSe
ttings;

    if (widget) {

this.widgetAssistant.n
ativeElement.style.top
= widget.assistantTop;

this.widgetAssistant.n
ativeElement.style.lef
t =
widget.assistantLeft;

this.widgetSubtitles.n
ativeElement.style.top
= widget.subtitlesTop;

this.widgetSubtitles.n
ativeElement.style.lef
t =
widget.subtitlesLeft;
    }
  }

  public
mouseDown(evt:
MouseEvent, item:
HTMLElement): void {

evt.preventDefault();

this.moveAt(evt.pageX,
evt.pageY, item);

    this.activeElem =
item;
  }


@HostListener('documen
t:mousemove',
['$event']) public
mouseMove(evt:
MouseEvent): void {
    if
(this.activeElem) {

this.moveAt(evt.pageX,
evt.pageY,
this.activeElem);
    }
  }

  public mouseUp(evt:
MouseEvent): void {

evt.preventDefault();
    const left =
+this.activeElem.style
.left.replace('px',
''),
      top =
+this.activeElem.style
.top.replace('px',
''),
      height =
+this.activeElem.offse
tHeight,
      width =
+this.activeElem.offse
tWidth;

    if (left < 0) {

this.activeElem.style.
left = '0px';
    }

    if (top < 0) {

this.activeElem.style.
top = '0px';
    }

    if (left + width >
window.innerWidth) {

this.activeElem.style.
left =
`${window.innerWidth -
width}px`;
    }

    if (top + height >
window.innerHeight) {

this.activeElem.style.
top =
`${window.innerHeight
- height}px`;
    }

    this.activeElem =
void 0;
  }

  public moveAt(pageX:
number, pageY: number,
elem: HTMLElement):
void {
    elem.style.left =
pageX -
elem.offsetWidth / 2 +
'px';
    elem.style.top =
pageY -
elem.offsetHeight / 2
+ 'px';
  }

  public
toggleCol(evt:
MouseEvent): void {

evt.preventDefault();
    this.isColumn =
!this.isColumn;
  }

  public
ngOnDestroy(): void {
    if
(this.electron.ipcRend
erer) {

this.electron.ipcRende
```

```
rer.removeAllListeners
('widgetActive');

this.electron.ipcRende
rer.removeAllListeners
('gladosActive');

this.electron.ipcRende
rer.removeAllListeners
('widgetMessage');
    }
  }

  private
saveSettings(): void {
    const settings =
this.settings.widgetSe
ttings;

settings.assistantLeft
=
this.widgetAssistant.n
ativeElement.style.lef
t;

settings.assistantTop
=
this.widgetAssistant.n
ativeElement.style.top
;

settings.subtitlesLeft
=
this.widgetSubtitles.n
ativeElement.style.lef
t;

settings.subtitlesTop
=
this.widgetSubtitles.n
ativeElement.style.top
;


this.settings.setWidge
tSettings(settings);
  }

}
```

Файл angular/widget.component.html

```
<div class="widget"
[class.widget_active]=
"isActive">
  <div
class="widget__assista
nt"
    #widgetAssistant

[class.widget__assista
nt_col]="isColumn"

(mousedown)="mouseDown
($event,
widgetAssistant)"

(mouseup)="mouseUp($ev
ent)">
    <div
class="widget__col">
      <span
class="widget__key">F1
0</span>
      <gosu-glados-eye
class="widget__eye"></
gosu-glados-eye>
      <span
class="widget__text">G
OSU — AI
Assistant</span>
    </div>
    <div
class="widget__col
widget__items"
*ngIf="widgetItems?.le
ngth">
      <div
class="item"
*ngFor="let item of
widgetItems">
        <div
class="item__image"
[style.background-
image]="'url(' + (item
| itemData : 'icon') +
')'"></div>
        <div
class="widget__text
widget__text_small">{{
item | itemData :
'name' }}</div>
      </div>
    </div>
  </div>
  <div
class="widget__subtitl
es"
*ngIf="hasSubtitles"
    #widgetSubtitles

(mousedown)="mouseDown
($event,
widgetSubtitles)"

(mouseup)="mouseUp($ev
ent)">
    <gosu-text-write
[text]="widgetMsg"></g
osu-text-write>
  </div>
</div>
```

Файл angular/widget.component.scss

```
@import
'app.settings';

.widget {
  position: absolute;
  left: 0;
  top: 0;
  width: 100%;
  height: 100%;
  overflow: hidden;

  &_active {
    background-color:
rgba($black, 0.2);

    .widget__subtitles
{
      background:
rgba($firefly, 0.4);
    }
  }

  &__assistant {
    position:
absolute;
    top: calc(100% -
170px);
    left: 0;
    background:
rgba($firefly, 0.4);
    border-radius:
12px;
    padding: 10px;
    cursor: move;
    user-select: none;
    display: flex;


.widget__col:first-
child {
      margin: 0 8px 0
0;
    }

    .widget__col:only-
child {
      margin: 0;
    }

    &_col {
      flex-direction:
column;


.widget__col:first-
child {
       margin: 0 0
8px 0;
      }


.widget__col:only-
child {
        margin: 0;
      }
    }
  }

  &__subtitles {
    position:
absolute;
    width: 500px;
    top: calc(100% -
300px);
    left: calc(50% -
250px);
    color: $white;
    cursor: move;
    user-select: none;
    text-align:
center;
    padding: 20px;
  }

  &__eye {
    width: 80px;
    height: 80px;
    margin-bottom:
12px;
  }

  &__col {
    position:
relative;
    width: 150px;
    height: 150px;
    display: flex;
    flex-direction:
column;
    align-items:
center;
    justify-content:
center;
  }

  &__key {
    position:
absolute;
    left: 0;
    top: 0;
    @include font(11,
13);
    color:
$pattensBlue;
    opacity: 0.5;
    letter-spacing:
0.2px;
  }

  &__text {
    @include font(12,
14);
    color: $white;
    text-shadow: 0px
1px 1px $black;
    &_small {
      @include
font(11, 13);
      letter-spacing:
0.2px;
      text-align:
center;
    }
  }

  &__items {
    flex-direction:
row;
    flex-wrap: wrap;
    justify-content:
flex-start;
  }
}

.item {
  width: 50%;
  flex-shrink: 0;

  &__image {
    width: 32px;
    height: 32px;
    background:
url('../../assets/imag
es/heroes/Ahri.png')
no-repeat center;
    background-size:
cover;
    margin: 0 auto
4px;
  }
```