НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
Кафедра автоматизованих систем обробки інформації і управління

«До захисту допущено»
В.о. завідувача кафедри

_____
О.А.Павлов
(підпис) (ініціали, прізвище)
“___” _____ 2019 р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки *6.050103 «Програмна інженерія»*

спеціальність *«Програмне забезпечення систем»*

на тему: Програмне забезпечення підтримки діяльності тестувальника

Виконав: студент 4 курсу, групи ІП-52

Ільїна Марія Дмитрівна _____
(прізвище, ім’я, по батькові) (підпис)

Керівник *старший викладач Халус О.А.* _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Консультант з графічної документації *доц. к.т.н. Ліщук К.І.* _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент *доц. каф. ТК, к.т.н., доц.Пасько В.П.* _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019 року

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проект	2	
2	А4	КПІ.ІП-5207.045440.01.81	Пояснювальна записка	60	
3	А4	КПІ.ІП-5207.045440.02.13	Опис програми	12	
4	А4	КПІ.ІП-5207.045440.03.91	Технічне завдання	13	
5	А4	КПІ.ІП-5207.045440.04.51	Програма та методика тестування	10	
6	А4	КПІ.ІП-5207.045440.05.34	Керівництво користувача	6	
7	А3	КПІ.ІП-5207.045440.06.99 КЕ	Креслення вигляду екранних форм	1	
8	А3	КПІ.ІП-5207.045440.06.99 СБД	Схема бази даних	1	
9	А3	КПІ.ІП-5207.045490.06.99 СС	Схема структурна класів програмного забезпечення	1	

Зм.	Арк.	ПІБ	Підп.	Дата	КПІ.ІП-5207.045440.00.90		
Розробн.		Ільїна М.Д.			Відомість дипломного проекту	Літ.	Лист 1 / Листів 1
Керівн.		Халус О.А					
Консульт						КПІ ім.Ігоря Сікорського кафедра АСОІУ гр. ІП-52	
Н/контр.		Ліщук К.І.					
В.о.зав.каф.		Павлов О.А.					

АНОТАЦІЯ

Пояснювальна записка дипломного проекту складається з чотирьох розділів, містить 15 таблиць та 10 рисунків – загалом 60 сторінок.

Об`єкт дослідження: програмне забезпечення підтримки діяльності тестувальника.

Мета дипломного проекту: розробити програмне забезпечення підтримки діяльності тестувальника.

У першому розділі проведено аналіз відомих аналогів, відомих технічних рішень, проведено порівняння обох. Також були сформовані функціональні та нефункціональні вимоги до продукту.

У другому розділі вибрали підхід до розробки додатку, розробили архітектуру, спроектували базу даних та описали класи, методи та інтерфейси додатку.

Третій розділ містить опис аналізу якості програми та загальний опис етапів та процесів тестування.

У четвертому розділі описано інструмент щодо розгортання та супровіду даного програмного забезпечення.

Також наведено опис програми, інструкцію користувача, програму та методику тестування, креслення вигляду екранних форм, схему структурну розгортання, схему бази даних.

КЛЮЧОВІ СЛОВА: ПЛАНУВАННЯ, ТЕСТУВАЛЬНИК, ВЕБ-ДОДАТОК, JIRA, CONFLUENCE.

ABSTRACT

The explanatory note of the diploma project consists of four sections, contains 15 tables and 10 sources – a total of 60 pages.

Object of research: software testing software support.

The purpose of the diploma project: to develop software for testing the testosterone.

In the first section, the analysis of known analogues, known technical solutions, a comparison of both were made. These were also formed functional and non-functional product requirements.

In the second section, we chose the approach to develop the application, developed the architecture, designed the database and described the classes, methods and interfaces of the application.

The third section contains a description of the quality analysis of the program and a general description of the stages and processes of testing.

The fourth section describes the deployment and maintenance tool for this software.

Also provided are: description of the program, user's manual, program and test method, drawing of the form of the form, structural deployment diagram, database schema.

KEYWORDS: PLANNING, TESTER, WEB APPLICATION, JIRA, CONFLUENCE.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Драг-енд-Дроп (в перекладі з англійської означає буквально тягни-і-кидай; Бери-і-Кинь) - спосіб оперування елементами інтерфейсу в інтерфейсах користувача (як графічним, так і текстовим, де елементи GUI реалізовані за допомогою псевдографіки) за допомогою маніпулятора «миша» або сенсорного екрану.

SDLC (Software development lifecycle)- це серія з шести основних фаз, через які проходить будь-яка програмна система. Далі ми розглянемо в загальних рисах фази життєвого циклу розробки системи, тримаючи в розумі, що всі системи різні за рівнем складності, необхідним компонентам і очікуваної функціональності.

QA (Англ. Quality Assurance, QA) - це процес або результат формування необхідних властивостей і характеристик продукції в міру її створення, а також - підтримку цих характеристик при зберіганні, транспортуванні та експлуатації продукції. Забезпечення якості визначено в стандарті ISO 9000: 2005 «Системи менеджменту якості. Основні положення і словник як частина менеджменту якості, зосереджена на створенні впевненості в тому, що вимоги до якості будуть виконані.

API (Application Programming Interface) — це набір готових класів, процедур, функцій, структур і констант, що надаються додатком (бібліотекою, сервісом) для використання в зовнішніх програмних продуктах (Вікіпедія). Своїми словами, API надає нам можливість використовувати чужі напрацювання в своїх цілях [10].

ВСТУП

Інформаційні технології заполонили наші життя майже у кожному куточку. Розрахуватись годинником у супермаркеті, вимкнути всю воду та електроенергію в один клік, знаходячись у іншій частині світу, приготувати вишукані страви, лише натиснувши кнопку з режимом - усе навколо вдосконалюється та прагне спростити життя людині.

У нашому сторіччі кожна деталь та кожна дія автоматизованна та має цифровий аналог.

Дипломна робота – результат чотирьох років навчання та сумлінної праці. Кожен хоче показати усі свої навички та зробити роботу, якак максимально була б корисною для людей та в першу чергу для самого дипломованого судента.

Працюючи у сфері інформаційних технологій вже більше двох років на позиції інженера із забеспечення якості програмних продуктів, командою дійшли висновку, що процесу роботи невистачає планеру задач, який збирав би усі потрібні інструменти та інформацію в одному місці, щоб зменшити кількість переходів між вікнами і зробити роботу більш продуктивною.

Отже, цілью дипломної роботи є створення зручного та простого програмного продукту, щоб оптимізувати процес роботи тестувальника, для цього приорітизувати задачі та отримати максимум продуктиності.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

Для того щоб зрозуміти, яким саме повинен бути продукт для покращення процессу роботи інженера з забеспечення якості треба повністю розуміти, якими є його основні задачі та обов'язки. Як протікає цей самий робочий процес.

Почнемо з деяких визначень.

Інженери з забезпечення якості мають на меті допомогти у створенні якісної продукції. Мова йде не про пошук помилок, і не про просте тестування. Основною функцією інженера з забезпечення якості є запобігання дефектам, а отже, забезпечення якості процесу розробки та його результатів.

Дефект - це недосконалий фрагмент коду, який змушує систему не виконувати необхідну функцію. Це не завжди означає, що щось не працює. Він може працювати неправильно.

Так що ж роблять інженери з якості?

Вони:

- виявляють слабкі сторони та невідповідності продукту на всіх етапах проекту;
- допомогають у визначенні вимог проекту;
- надають вичерпну інформацію про рівень якості продукту;
- тестують продукту на всіх етапах життєвого циклу розробки програмного забезпечення (SDLC).

Важливо відзначити, що QA зацікавлені в тому, щоб будь-який продукт був зручним для користувача, будь то функціональний або дизайн. Для цього QA інженери тісно спілкуються з усіма членами команди і постійно звертаються до цих вимог.

Розглянемо всі етапи, на яких бере участь QA, їхні ролі на цих етапах, та ділову цінність, яку приносить їхня робота.

У цій частині наведено, що роблять QA та в який момент вони це роблять. Стадії, які описані, є загальними (і узагальненими) частинами життєвого циклу тестування програмного забезпечення (STLC) зображено на Рисунку 1.1.

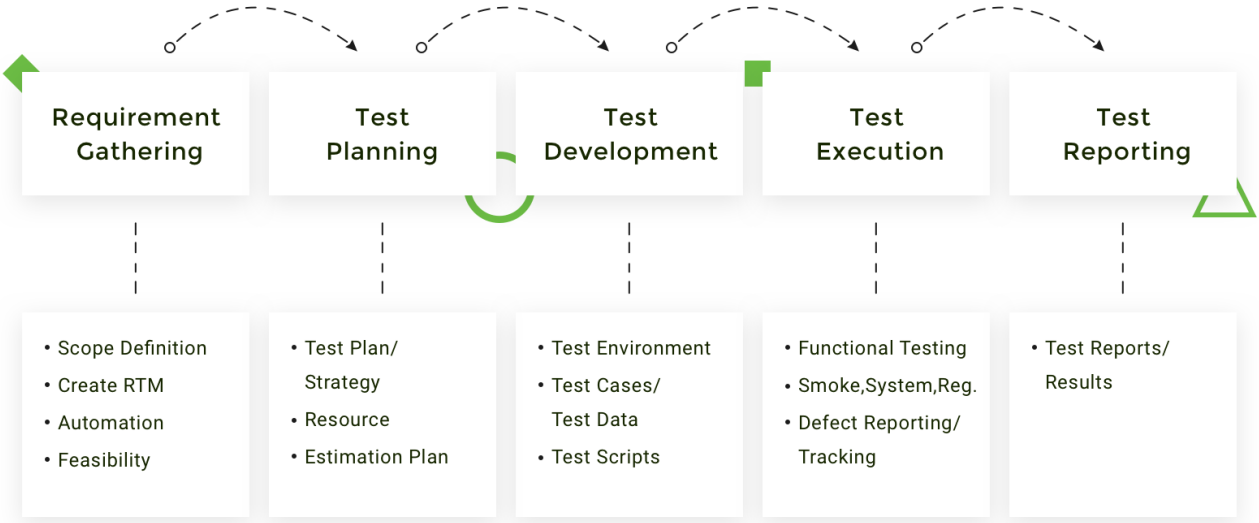

Рисунок 1.1 – Процес роботи QA інженера

Визначимося з тим, що таке оптимізація та планування.

Оптимізація - процес максимізації вигідних характеристик, співвідношень (наприклад, оптимізація виробничих процесів і виробництва), і мінімізації витрат.

Завдання оптимізації сформульована, якщо задані: критерій оптимальності (економічний, технологічні вимоги - вихід продукту, вміст домішок в ньому та інше); варіюють параметри (наприклад, температура, тиск, величини вхідних потоків в процесах переробки гірського і ін. сировини), зміна яких дозволяє впливати на ефективність процесу; математична модель процесу; обмеження, пов'язані з економічними і

конструктивними умовами, можливостями апаратури, вимогами вибухобезпеки і інше.

Планування - оптимальний розподіл ресурсів для досягнення поставлених цілей, діяльність (сукупність процесів), пов'язана з постановкою цілей (завдань) і дій в майбутньому. З точки зору математики, планування - це функція, одним з аргументів якої є час.

Згідно «Webster's New Collegiate Dictionary» планування - це розробка методу для створення або виконання чого-небудь для досягнення мети. Таким чином планування є щоденною, майже несвідомою діяльністю для всіх

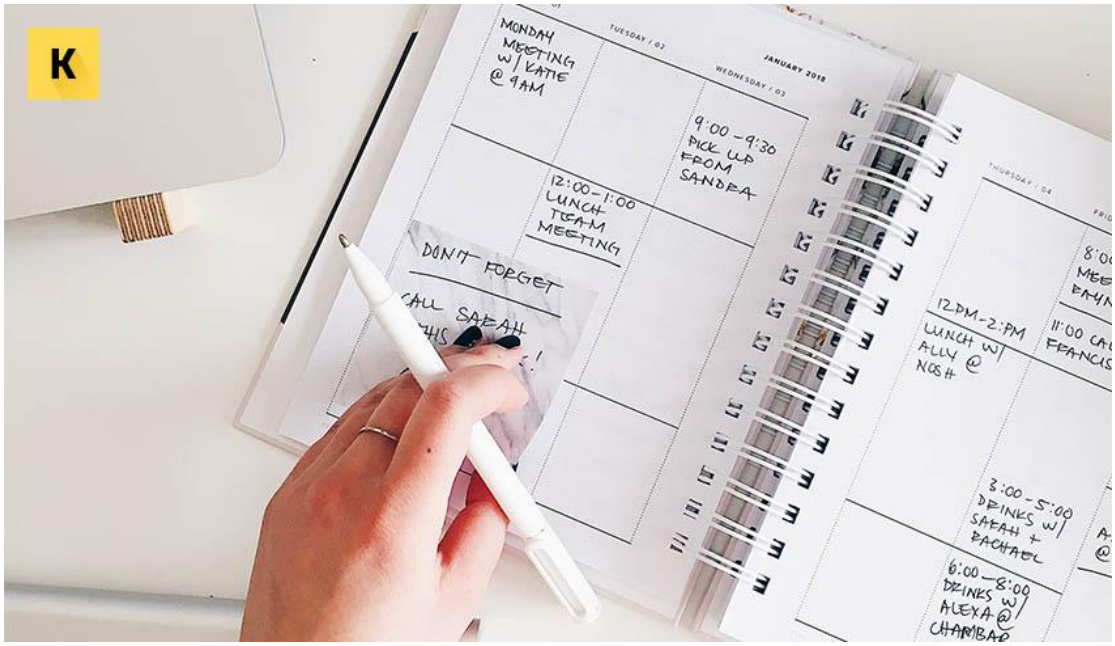

Рисунок 1.2 – Процес планування

1.2 Змістовний опис і аналіз предметної області

Щоденник - зошит для ведення щоденних записок. На рисунку 1.3 зображений розгорнутий планер.

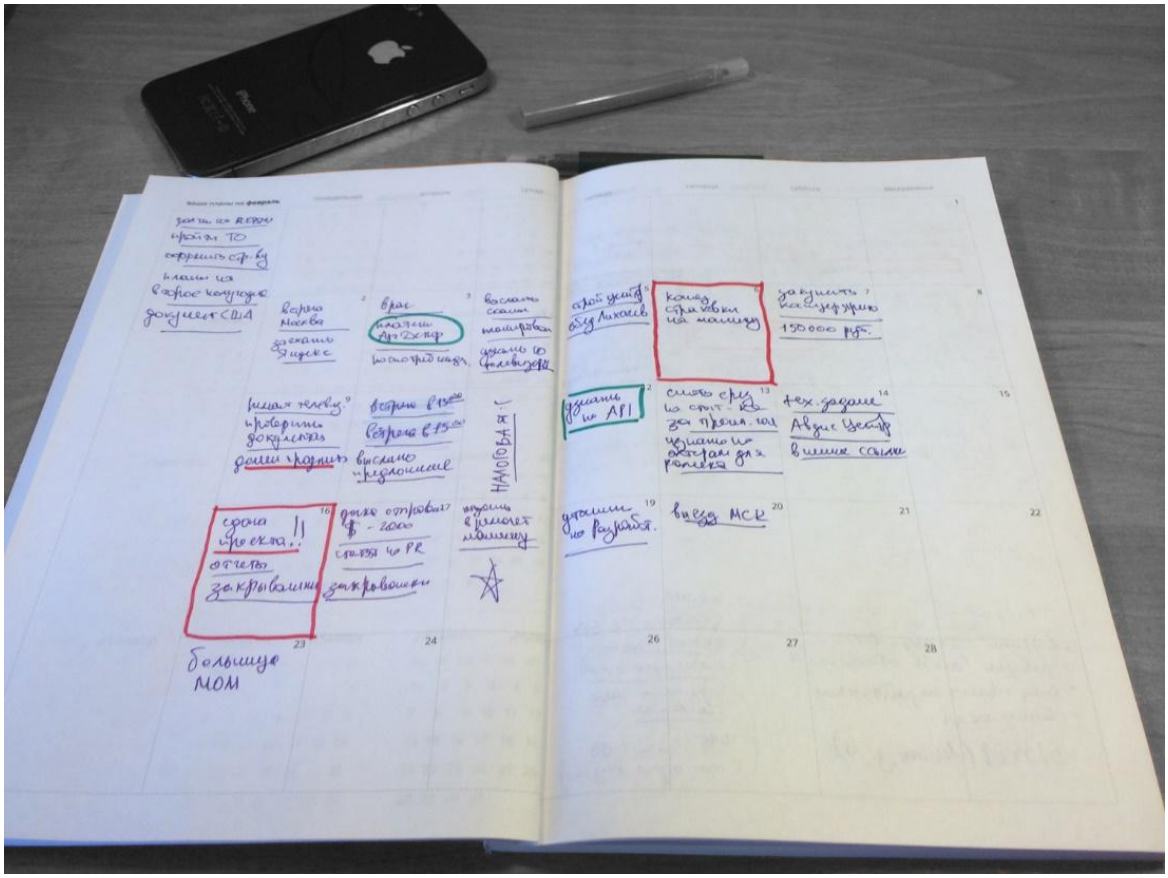

Рисунок 1.3 – Розгорнутий планер

Першу згадку про щоденниу прийнято пов'язувати з Італією 1650 р звідки і взялося перше слово (лат. Agenda: букв. «Речі, які треба зробити»).

У сучасному розумінні - діловий (звичайно) блокнот, призначений для щоденного ведення записів про справи, плани, результати та інших записах, пов'язаних з щоденної ділової діяльністю людини або компанії / організації. Найбільш популярними видами є щоденники у вигляді книги, в різному палітурці. Одним з перших видів щоденників, якими користувалися Сартр, Пікассо, А. С. Куракін, Ван Гог, Аполлінер, Матісс, Гертруда Стайн, Хемінгуей, Брюс Чатвін названий на честь міцного матеріалу, який дозволяв «виживати» паперовому одному в різних умовах разом зі своїм господарем - Чортової шкірою (дослівно англ. Moleskine - хутро крота), молескіновимі робамі могли в давні часи похвалитися ремісники і шукачі золота.

Італійські видавці дуже пишаються тим, що є історичною батьківщиною щоденників, і довгий час італійське місто Бергамо разом з

адміністративним центром цієї провінції - Міланом, вважався не тільки європейським, а й світовим центром виготовлення найпопулярніших марок щоденників.

З кінця XVII століття і до цього дня, наявність щоденника свідчило про значущість даної особи, а також мало на увазі те, що в щоденнику було що записувати. Це могли бути боргові зобов'язання, справи, податкові відомості і накази розпорядникам. Тому сам блокнот виготовлявся з усіма регаліями і гербами, властивими прізвища - за спеціальним замовленням палітурних справ майстрами. Найчастіше переплітався в шкіру, з великою кількістю позолоти і містив у своєму первісному вигляді персональні дані власника, де вказано кому він належить, куди і за яку винагороду слід повернути щоденник в разі його втрати.

У наш час щоденник, і традиція його використання, набув широкого поширення, а сам принцип ведення списку справ і планування часу переріс у цілі напрямки по управлінню часом (Тайм-менеджмент, Проектний-менеджмент).

В даний час щоденники набувають найрізноманітніші форми виконання, які залежать від можливостей видавця. Зустрічаються навіть щоденники-розмальовки. Та й значимість Італійських виробників, як законодавців моди в даному питанні, втратила силу. В СНД вже існують національні виробники, які не поступаються, а іноді і перевершують за якістю виконання, автентичним засновникам традиції планування.

Незважаючи на всю різноманітність - найпопулярнішим залишається щоденник у вигляді книги в благородній твердій палітурці, виконаним в м'якій шкірі, як правило забезпечений додатковою інформацією для комфортної роботи його власника (коди телефонного зв'язку, таблиці відстаней між місцевими обласними центрами, конвертація одиниць виміру, державні і міжнародні свята, і т. п.), а також (тільки для датованих

щоденників) підготовлений для ведення щорічного, щомісячного, щотижневого та щоденного розкладу. Що, втім, не заважає існуванню недатованих щоденників, дати і події в яких проставляє сам власник.

Найбільш традиційні формати щоденників: А5, А4. В цьому плані кожен визначає для себе найбільш зручні розміри:

Щоденник А4 - 21 * 29см - настільний варіант, тому що носити такий щоденник на зустрічі і ділові переговори некомфортно.

Щоденник А5 - 15 * 21 см - найпопулярніший формат, тому що дозволяє легко носити щоденник в портфелі або ділової сумці.

Крім розмірів, виробництво щоденників передбачає вибір їх внутрішнього наповнення. Папір для них найчастіше беруть щільністю 70-80 гр / м2, колірна гамма найчастіше біла або кремова.

З розвитком бізнес-середовища в різних країнах, щоденник, з персонального продукту, що продається в роздрібних мережах, перетворюється в діловій сувенір, який прийнято дарувати від імені компанії найкращим клієнтам або партнерам. Для цього рекламні агентства освоїли різні методи брендування і ексклюзивного виконання щоденників, з метою підкреслити імідж компанії, яка здійснює такий подарунок, а також цінність подарунка, і шанобливе ставлення до особи, якій такий подарунок був піднесений.

В наші дні щоденники використовуються все рідше і витісняються іншими більш універсальними засобами для планування особистого і робочого часу, починаючи від схожих по використанню щоденників-планерів, до мобільних додатків, що включають в себе планувальники завдань і доручень.

У щоденнику ділова людина детально розписує свою діяльність: роботу і дозвілля. На кожній сторінці всю інформацію можна фіксувати по годинах і хвилинами. Все це робиться для того, щоб не забувати про важливі

заходів. У щоденнику можна коротко законспектувати важливу інформацію, щоб нічого не забути. Найчастіше це відноситься до робочої інформації. Людина з щоденником виглядає набагато акуратніше, ніж з листами. Якщо партнер по бізнесу бачить, що у його співрозмовника є якісний щоденник, то це говорить йому про надійність. Щоденник можна піднести як подарунок.

Він відноситься до числа універсальних: підходить для всіх людей і різних випадків. Тому діловій людині просто не обійтися без нього. Підійде така річ і для тих, хто любить розписувати своє життя дуже докладно. Схожої річчю є блокнот.

Найчастіше зустрічається формат А5, і такий варіант прекрасно підійде для ділової поїздки і для переговорів. Різновидом щоденників вважаються записні книжки. Для ділових людей найбільше підходять шкіряні обкладинки.

У наш вік автоматизації усе оцифровується і щоденники, планери для нотаток стають анологовими та більш зручними для вікорстання та доступу.

1.3 Аналіз успішних IT-проектів

1.3.1 Аналіз відомих технічних рішень

Оскільки сфера інформаційних технологій розвивається достатньодовгий проміжок часу, існує достатня кількість необхідних технічних рішень для того,щоб зробити веб-застосунок для планування найбільш комфортним для користування. Основними використаними технологіями є : драг-енд-дроп, інтеграція зі сторонніми сервісами, бібліотека для календаря.

Драг-енд-Дроп як спосіб оперування елементами інтерфейсу в інтерфейсах користувача (як графічним, так і текстовим, де елементи GUI реалізовані за допомогою псевдографіки) за допомогою маніпулятора «миша» або сенсорного екрану.

Спосіб реалізується шляхом «захоплення» (потримавши головною (першої, частіше лівої) кнопки миші) відображуваного на екрані комп'ютера об'єкта, програмно доступного для подібної операції, і переміщенні його в інше місце (для зміни розташування) або «кидання» його на інший елемент (для виклику відповідного, передбаченого програмою, дії). По відношенню до вікон (також здатним до переміщення подібним способом) цей термін зазвичай не вживається.

Базовими діями і найпростішими прикладами драг-енд-дроп дій є: переміщення об'єкта, переміщення об'єкта з панелі в панель, хоча в сучасних операційних системах драг-енд-дроп набув широкого застосування і є одним з головних способів взаємодії з комп'ютером в графічному інтерфейсі користувача .

Об'єктами для переміщення можуть бути наступні елементи інтерфейсу: значки (іконки) Робочого столу, плаваючі панелі інструментів, ярлики програм в панелі завдань (починаючи з Win XP), елементи. Переміщатися об'єкти можуть як в межах деякої певної області, в межах одного вікна, між панелями одного вікна, так і між різними вікнами.

Подія перетягування повинно ініціюватися будь-якою дією користувача. Найчастіше цим дією є натискання лівої кнопки миші на елементі (подія це називається МаусДаун), який може бути переміщений в своєму контейнері. Деякі компоненти володіють власними подіями початку драг-енд-дроп.

Модуль calendar дозволяє надрукувати собі календарик (а також містить деякі інші корисні функції для роботи з календарями).

calendar.Calendar (firstweekday = 0) - клас календаря. firstweekday - перший день тижня (0 - понеділок, 6 - неділя).

Методи:

– iterweekdays () - итератор днів тижня, починаючи з firstweekday;

– itermonthdates (year, month) - итератор для місяця month року year. Повертає всі дні цього місяця (як об'єкти datetime.date), а також дні до і після цього місяця до повного тижня;

– itermonthdays2 (year, month) - як itermonthdates, тільки дні повертаються не як datetime.date об'єкти, а кортежі (номер дня, номер дня тижня);

– itermonthdays (year, month) - як itermonthdates, тільки дні повертаються не як datetime.date об'єкти, а номери днів;

– monthdatescalendar (year, month) - список тижнів у місяці. Тиждень – список з 7 об'єктів datetime.date;

– monthdays2calendar (year, month) - як monthdatescalendar, але об'єкти - кортежі (номер дня, номер дня тижня);

– monthdayscalendar (year, month) - як monthdatescalendar, але об'єкти – номери днів;

– calendar.TextCalendar (firstweekday = 0) - клас для генерації текстового календаря.

Методи:

– formatmonth (theyear, themonth, w = 0, l = 0) - повертає календар на місяць у вигляді рядка, з шириною колонки w і висотою l;

– prmonth (theyear, themonth, w = 0, l = 0) - друкує календар на місяць.

– formatyear (theyear, w = 2, l = 1, c = 6, m = 3) - повертає календар на рік; з m колонок, шириною дати w, висотою тижні l і кількістю прогалин між місяцями c;

– pryear (theyear, w = 2, l = 1, c = 6, m = 3) - друкує календар на рік.

– calendar.HTMLCalendar (firstweekday = 0) - клас для генерації HTML календаря.

Методи:

– formatmonth (theyear, themonth, withyear = True) - календар на місяць

у вигляді HTML таблиці. Якщо withyear True, номер року буде включений в заголовок;

– formatyear (theyear, width = 3) - календар на рік у вигляді HTML таблиці. width - кількість місяців в ряду;

– formatyearpage (theyear, width = 3, css = "calendar.css", encoding = None) - календар на рік у вигляді повноцінної HTML сторінки, з підключенням файлу css (який ви можете створити самі), і в кодуванні encoding;

– calendar.LocaleTextCalendar (firstweekday = 0, locale = None) – дозволяє створити текстовий календар з назвами на рідній мові;

– calendar.LocaleHTMLCalendar (firstweekday = 0, locale = None) – дозволяє створити HTML календар з назвами на рідній мові.

Оскільки API REST базується на відкритих стандартах, можна скористатися будь-якою мовою веб-розробки для доступу до API. Проте, якщо ви використовуєте Java, найпростіший спосіб розпочати роботу з JIRA REST API - це завантажити JIRA REST Java Client (JRJC) і використовувати її як бібліотеку у вашому додатку. Для інших мов зверніться до дому API JIRA REST для прикладів коду.

Структура REST URI

API REST API JIRA забезпечують доступ до ресурсів (об'єктів даних) через URI шляхи. Щоб використовувати API REST, ваша програма зробить HTTP-запит і розбере відповідь. API JIRA REST використовує JSON як свій формат зв'язку, а стандартні методи HTTP, такі як GET, PUT, POST і DELETE (див. Описи API нижче, для яких методи доступні для кожного ресурсу). URI для ресурсу API REST API JIRA мають таку структуру:

http: // host: port / context / rest / api-name / api-version / назва-ресурсу

Наразі доступно два назви API, про які йтиметься нижче:

auth - для операцій, пов'язаних з аутентифікацією, і

api - для всього іншого.

Поточна версія API - 2. Однак існує також символьна версія, яка називається останньою, яка вирішується до останньої версії, що підтримується даним екземпляром JIRA. Наприклад, якщо ви хочете отримати представлення JSON випуску JRA-9 з публічного відстеження проблем у Atlassian, ви отримаєте доступ до:

https://jira.atlassian.com/rest/api/latest/issue/JRA-9

Існує документ WADL, який містить документацію для кожного ресурсу в JIRA REST API. Він доступний тут.

Розширення в API REST

Щоб мінімізувати мережевий трафік і використання процесора на сервері, API JIRA REST іноді використовує техніку, яка називається розширенням. Коли ресурс REST використовує розширення, то частини цього ресурсу не будуть включені до відповіді JSON, якщо явно не запитано. Спосіб запиту цих фрагментів для включення полягає у використанні параметра розширення запиту.

Параметр розгорнутих запитів можна використовувати для визначення списку об'єктів, розділених комами, які потрібно розширити, ідентифікуючи кожне з них за назвою. Наприклад, додавання імен розкриття = renderedFields до URI запитів вимагає включення в відповідь імен перекладених полів і значень поля, що відображаються в HTML. Продовжуючи наведений вище приклад, ми використаємо таку URL-адресу, щоб отримати цю інформацію для JRA-9:

https://jira.atlassian.com/rest/api/latest/issue/JRA-9?expand=names,renderedFields

Щоб виявити ідентифікатори для кожного об'єкта, подивіться на властивість розширення у батьківському об'єкті. У прикладі JSON, наведеному нижче, ресурс оголошує віджети розширюваними.

```
{
   "expand": "widgets",
   "self": "http://www.example.com/jira/rest/api/resource/KEY-1",
   "widgets": {
      "widgets": [],
      "size": 5
   }
}
```


Можна використовувати позначення крапки, щоб вказати розширення об'єктів в межах іншого об'єкта. Наприклад? Expand = widgets.fringels розширює колекцію віджетів, а також властивість fringel для кожного віджета.

Аутентифікація

Будь-яка аутентифікація, яка працює проти JIRA, буде працювати проти REST API. Переважними методами аутентифікації є OAuth і HTTP Basic (при використанні SSL), які задокументовані в навчальних посібниках JIRA REST API. Інші підтримувані способи включають: файли cookie HTTP і надійні програми.

На сторінці входу використовується аутентифікація на основі cookie, тому, якщо ви використовуєте JIRA у браузері, ви можете викликати REST з Javascript на сторінці і покладатися на аутентифікацію, яку встановив браузер. Абоненти, які бажають відтворити поведінку сторінки входу JIRA (наприклад, для відображення повідомлень про помилки автентифікації користувачам) або викликають POST API REST і SOAP, на ресурс / auth / 1 / session.

1.3.2 Аналіз відомих програмних продуктів

Серед відомих продуктів із цієї області слід відзначити **Leader Task20**

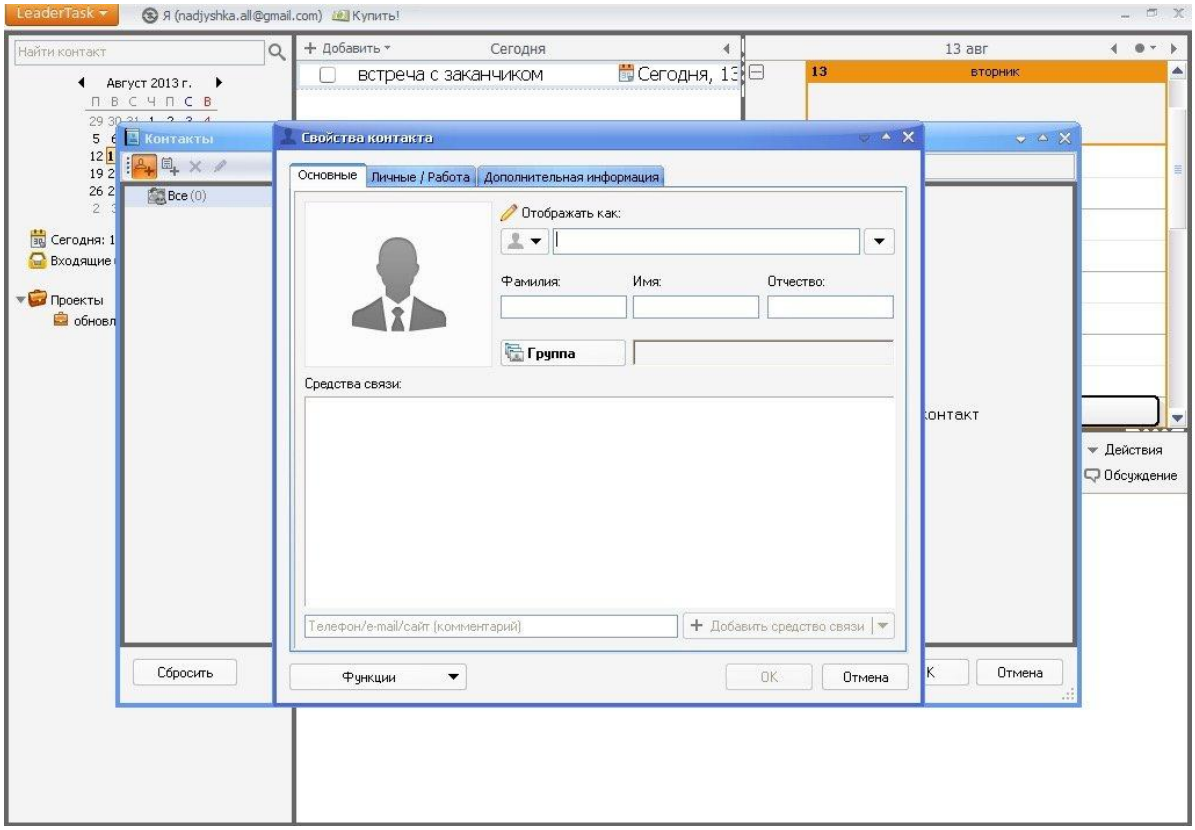

Рисунок 1.4 – Діловий органайзер Leader Task 20

Серед додатків для планування LeaderTask найбільш популярна, перш за все тому, що це - Мультиплатформенне рішення. Користувачам LeaderTask доступні версії для ПК і мобільних платформ - Android, iOS з можливістю синхронізації даних між пристроями.

Після запуску Windows-клієнта LeaderTask значок програми розміщується в системному треї, забезпечуючи таким чином простий і зручний доступ до додатка. Користувачі можуть додавати в базу LeaderTask замітки і завдання. Програма зручна тим, що на одному екрані тут представлені відразу ж списки завдань і заміток, а також список зустрічей, розташований прямо на календарній сітці.

Завдання в LeaderTask можна відразу ж присвоювати проектам, що в подальшому спрощує їх упорядкування та пошук. Десктопна версія програми підтримує drag & drop -завдання можна привласнювати проектам простим перетягуванням їх на список проектів. Точно так само можна призначати

завдання конкретних дат - досить завдання просто перетягнути на потрібну дату або в сітку календаря на потрібний час.

Для кожного завдання можна привласнити цілий ряд додаткових параметрів, в тому числі колір, яким завдання буде виділена в загальному списку. Завдання можуть супроводжуватися текстовим коментарем, до них можна також прикріпити файл. Для складних завдань LeaderTask дозволяє додавати підзадачі. Є можливість задавати пріоритети, налаштовувати нагадування.

Використання фільтрів в LeaderTask дозволить зручно управляти завданнями, проводити пошук потрібних. Фільтри можна будувати на основі декількох критеріїв відбору. У тому числі програма дозволяє побудувати фільтри на основі календаря, відібравши таким чином завдання, які заплановані для певного проміжку часу. У вікні програми можна переключатися між режимами календаря, проектів, категорій, контактів.

До речі, список контактів - одна з незвичайних функцій програми для управління завданнями. Програма дає можливість створити (імпортувати зі смартфона) список контактів і прив'язувати до них завдання. Завдяки цьому можна побачити, які завдання були доручені тієї чи іншої людини - співробітнику або знайомому. В цілому LeaderTask є програмою для зручної оптимізації та систематизації завдань, які виникають перед користувачем. Спробувати програму можна безкоштовно протягом 45 днів.

Плюси: Повністю локалізована, є додатки для мобільних ОС, наявність фільтрів, підтримка проектів

Мінуси: Висока вартість ліцензії

Ще одним аналогом є **ANY.DO**. Цей додаток має версії не тільки для iOS і Android, а й як додаток для браузера Google Chrome. Any.DO має простий і зручний інтерфейс. У мобільних версіях сортувати списки можна простим перетягуванням їх елементів. Одна з переваг програми - завдання

можна промовляти вголос, а мобільна версія сприйме сказане і збереже його у вигляді текстового нагадування. Є підтримка і російської мови.

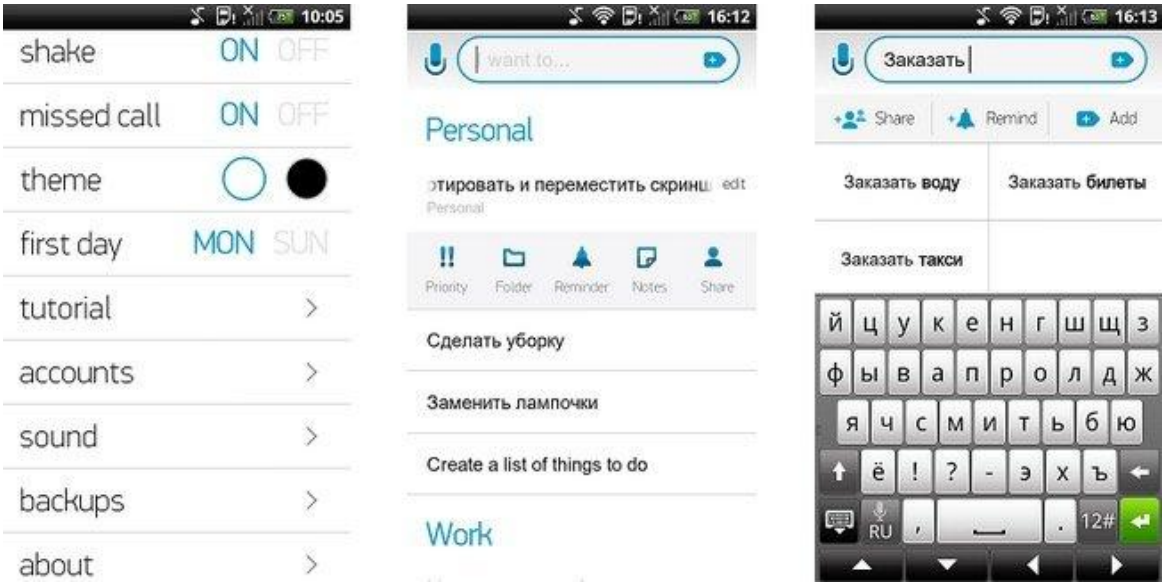

Рисунок 1.5 – Додаток Any.DO

При додаванні завдання Any.DO дозволить переміщати завдання в папки, позначати їх важливість, налаштовувати повторення завдання, додавати до нього розширений опис. До завдання можна прикріпити елемент списку контактів, наприклад, якщо додається завдання - це зустріч з кимось із знайомих або друзів. Одночасно з цим можна налаштувати відправку цієї людини повідомлення про додану задачі. Завдяки цій можливості додаток Any.DO можна використовувати як інструмент планування завдань для співробітників невеликої компанії.

У програмі є можливість установки геолокаційні міток для задач і настройка нагадувань відповідно до місця знаходження користувача. Наприклад, програма зможе нагадати користувачеві купити певні продукти, якщо він виявився в торговому центрі або підкаже йому викупити квитки на прем'єру, якщо користувач виявився біля квиткових кас.

Очевидно, що список продуктів і нагадування потрібно налаштувати заздалегідь. У додатку Any.DO автоматично формуються завдання на основі пропущених або відхилених телефонних дзвінків, вміст завдання -

передзвонити по заданому номеру. Програма оснащена можливостями синхронізації даних між пристроями, а також - зі списком завдань Google Tasks. Також є можливість зберігати резервну копію створених списків.

Плюси: Є голосове введення, геолокаційні мітки, синхронізація і оповіщення інших користувачів

Мінуси: не завжди коректно локалізовані пункти меню

Також достойним варіантом є **2Do**. Програма 2Do: Todo List | Task List представляє собою зручний планувальник завдань для Android-пристрої. Користувачам доступні додавання завдань, використання для них тегів і присвоювання їм геолокаційні мітки, що дозволяє визначити місце реалізації цього завдання (вдома, в офісі, в торговому центрі).

Основні принципи застосування побудовані з використанням відомої системи планування Getting Things Done. Окремі записи в 2Do: Todo List | Task List можна захистити паролем. У програмі немає власного сховища даних, зате в ній можна налаштувати синхронізацію з вашим аккаунтом в Dropbox. А значить, до даних ви отримаєте доступ з будь-якого мобільного пристрою.

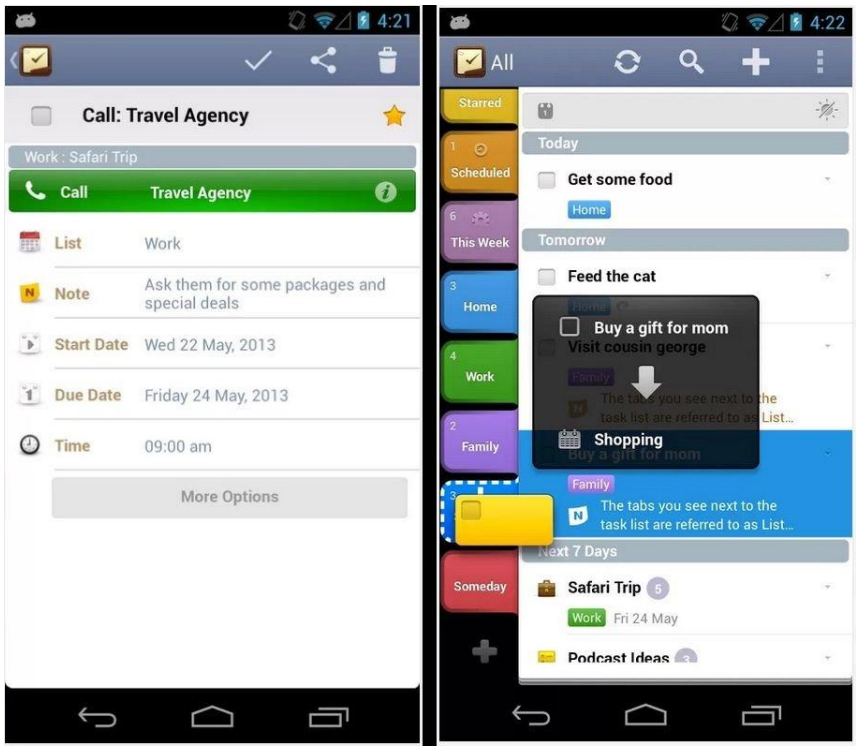

Рисунок 1.6 – Знімок з екрану додатку 2Do

Плюси: Синхронізація списків справ з різними пристроями, в тому числі, з iOS-смартфонами і планшетами.

Мінуси: Немае безкоштовної версії.

Наведемо порівняльну характеристику додатків в таблиці 1.1.

Таблиця 1.1 – Порівняльна характеристика продуктів

Характеристики	LeaderTask 20	Any.DO	2Do:Todo List|Task List
Кросс-браузерність, кросс-платформенність	+	+	+
Можливість пріорітизації задач	+	——	——
Синхронізація з іншими притроями	——	+	+
Наявність безкштовної версії	——	+	——
Інтеграція зі сторонніми сервісами	——	——	——
Зручний інтерфес	——	+	——

Використовуючи досвід попередників, можна зробити ставку на вдосконалення моментів, що були недостатньо продумані в них.

1.4 Аналіз вимог до програмного забеспечення

Планер головною своєю цілю має спрощення та прискорення роботи тестувальника, тобто він повинен бути простим та швидким у використанні. Тому буде реалізовано лише найнеобхідніші функції.

1.4.1 Розроблення функціональних вимог

Планер має наступні функції:

- вхід через Jira;
- свторення задачі;
- зміна статусу задачі;
- видалення задачі;
- редагування задачі;
- додавання зв’язку з тікетом у Jira;
- додавання зв’язку з сторінкою у Confluence.

Сервіс являє собою клієнт-серверний застосунок. Розглянемо варіанти використання клієнтської частини задля деталізації вимог.

Таблиця 1.2 – Варіант використання GE001

Назва	Вхід через Jira
Опис	Користувач має можливість авторизуватись в системі, використовуючи свій акаунт у Jira
Передумови	Неавторизований користувач відкрив додаток
Постумови	Користувач авторизований
Основний сценарій	а) користувач натискає на кнопку “Вхід через Jira”; б) кідкривається додаткова вкладка у браузері; в) у цій вкладці форма логіну у Jira; г) користувач авторизується у Jira; д) користувача повертає у веб-додаток, де він вже авторизований.

Таблиця 1.3 – Варіант використання GE002

Назва	Створити задачу
Опис	Щоб створити задачу користувач натискає на кнопку створення та заповнює всі необхідні поля
Передумови	Користувач авторизувався
Постумови	Задачас створена
Основний сценарій	а) користувач натискає кнопку “Створити задачу”; б) відкиваеться модальне вікно для створення; в) користвувач заповнює всі поля: тема, опис, «Jira», «Confluence»; г) користувач натискає кнопку «Зберегти», зада свторена та ввдображена на дошці.

Таблиця 1.4 – Варіант використання GE003

Назва	Видалити задачу
Опис	Користувач має можливість прибрати с дошки (видалити) будь-яку задачу.
Передумови	Користувач маэ хоча б дну задачу.
Постумови	Задача видалена.
Основний сценарій	а) натиснути кнопку видалення; б) система ставить маркер задачі «видалено», задача зникає.

Таблиця 1.5 – Варіант використання GE004

Назва	Редагувати задачу
Опис	Користувач може редагувати будь-яку інформацію.
Передумови	Користувач має хоча б одну задачу на дошці.
Постумови	Задача відредактована
Основний сценарій	а) натиснути кнопку редагування; б) змінити усю необхідну інформацію; в) натиснути кнопку збереження; г) задачу відредактовано, відображаються нові дані.

Таблиця 1.6 – Варіант використання GE005

Назва	Виконати задачу
Опис	Останньою стадією циклу задачі є її виконання.
Передумови	Користувач має хоча б одну задачу.
Постумови	Задача виконана.
Основний сценарій	а) натиснути на чек-бокс; б) у чек-боксі з’являється галочка, задача вдображаеться, як виконана (зафарбована сірим кольором).

Таблиця 1.7 – Варіант використання GE006

Назва	Змінити приорітет задачі
Опис	Функціонал додатку включає в себе можливість приорітизувати задачі, тобто змінювати їх місце в списку
Передумови	Користувач має хоча б одну задачу
Постумови	Приорітет (розполодення задачі да тощці) змінено
Основний сценарій	а) навести на задачку курсор, натиснути ліву кнопку миші та утримувати; б) перетягнути здачу н потрібне місце і вібустити ліву кнопку миші; в) приорітет та місцерозташування задачі змінено.

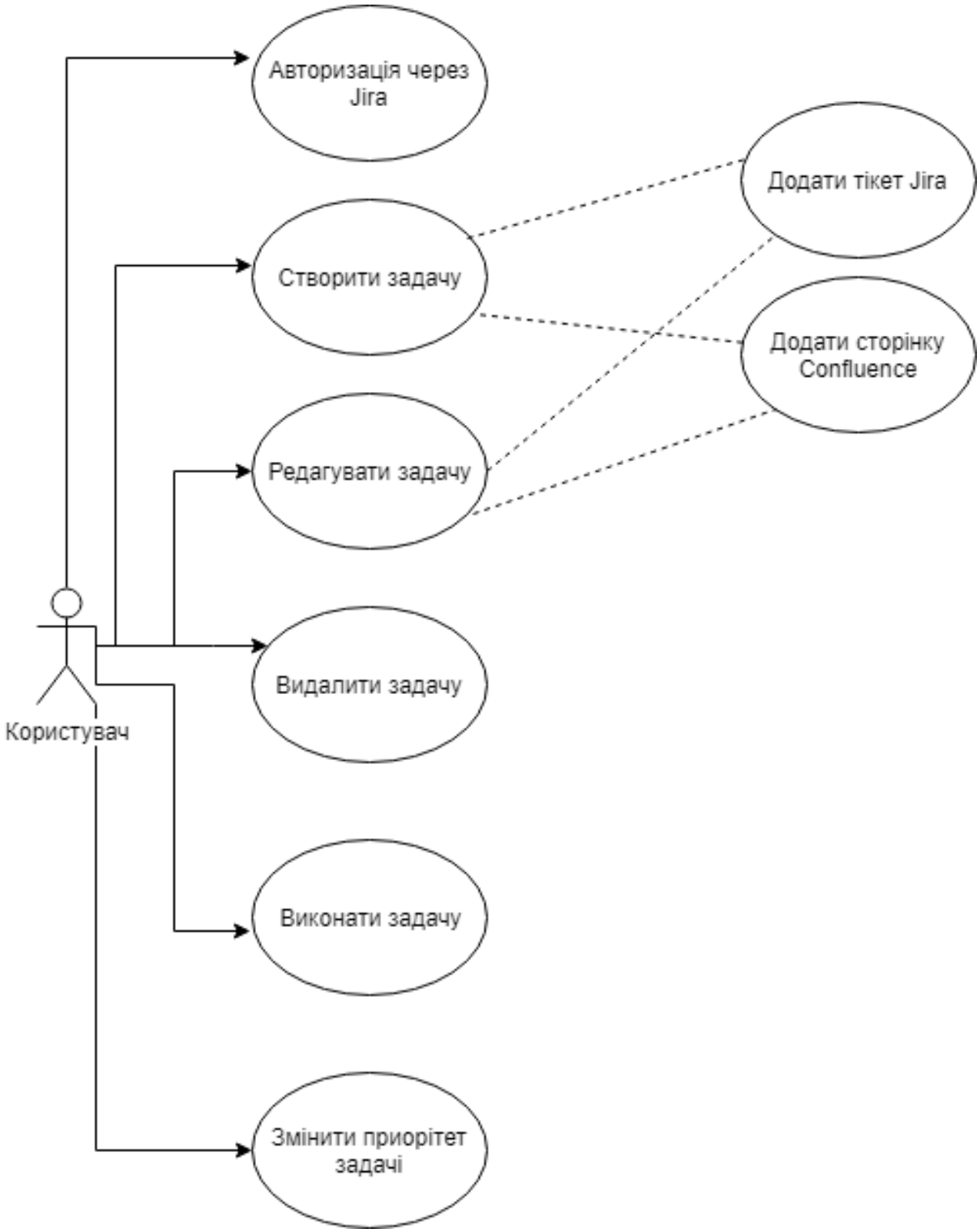

Рисунок 1.7 – Схема структурна варіантів використання

Для того, щоб візуалізувати зв’язок між функціональними вимогами та варіантами використання створимо матрицю трасування.

Таблиця 1.6 – Матриця трасування

	Увійти через JIRA	Додати задачу	Видалити задачу	Редагувати задачу	Виконати задачу	Змынити приорытет задачі
GE001	■					
GE001		■	■	■	■	■
GE002			■			
GE004				■		
GE005					■	
GE006						■

Уся логіка створення об’єктів та роботи з ними а також інтеграції зі стороннім сервісами буде виконана на сервері.

1.4.2 Розроблення нефункціональних вимог

Програмне забезпечення повинне відповідати наступним нефункціональним вимогам:

- локалізація інтерфейсу – можливість відображати всі тексти будь якою мовою;
- підтримка браузерів Chrome, FF, Opera, Safari;
- клієнт-серверна взаємодія – через API;
- передача даних через захищені канали.

1.4.3 Постановка комплексу завдань модулю

Розроблений продукт повинен допомогати тестувальнику оптимально спланувати свій день і поєднати у собі усі необхідні інструменти, бути простим і швидким у використанні.

Простота забезпечується інтуїтивним UX та відсутністю зайвих деталей.

Мета розробки – підвищити продуктивність тестувальника, допомогти оптимізувати процесс та приорітизувати задачі. Для цього потрібно виконати наступні задачі:

- отримання інформації про оновлення статусів задач у Jira;
- зберігання проектної документації;
- зберігання тестової документації;
- планування та приорітизація задач.

1.5 Висновки до розділу

У даному розділі я проаналізувала предметну область продукту, технології, які будуть використані та провела порівнянння з аналогами та конкурентами продукту.

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

Для моделювання програмного забезпечення зобразимо схематично усі сценарії клієнт-серверної взаємодії. Для конструювання використаємо використаємо структурні схеми діяльності. Нам знадобиться послідовний опис кожного сценарія.

Сценарій авторизації в додатку(через Jira):

- користувач натискае кнопку “авторизуваися у джира”;
- сервер пренапраправляє запит на сторінку авторизації в джира, де
- користувач вводить свої дані;
- на сервер надсилаються дані користувача;
- сервер джира перевіряє, чи є такий користувач у базі;
- якщо так, так користувача авторизовано і він може починати роботу з
- планування;
- якщо ні, виводиться повідомлення про помилку.

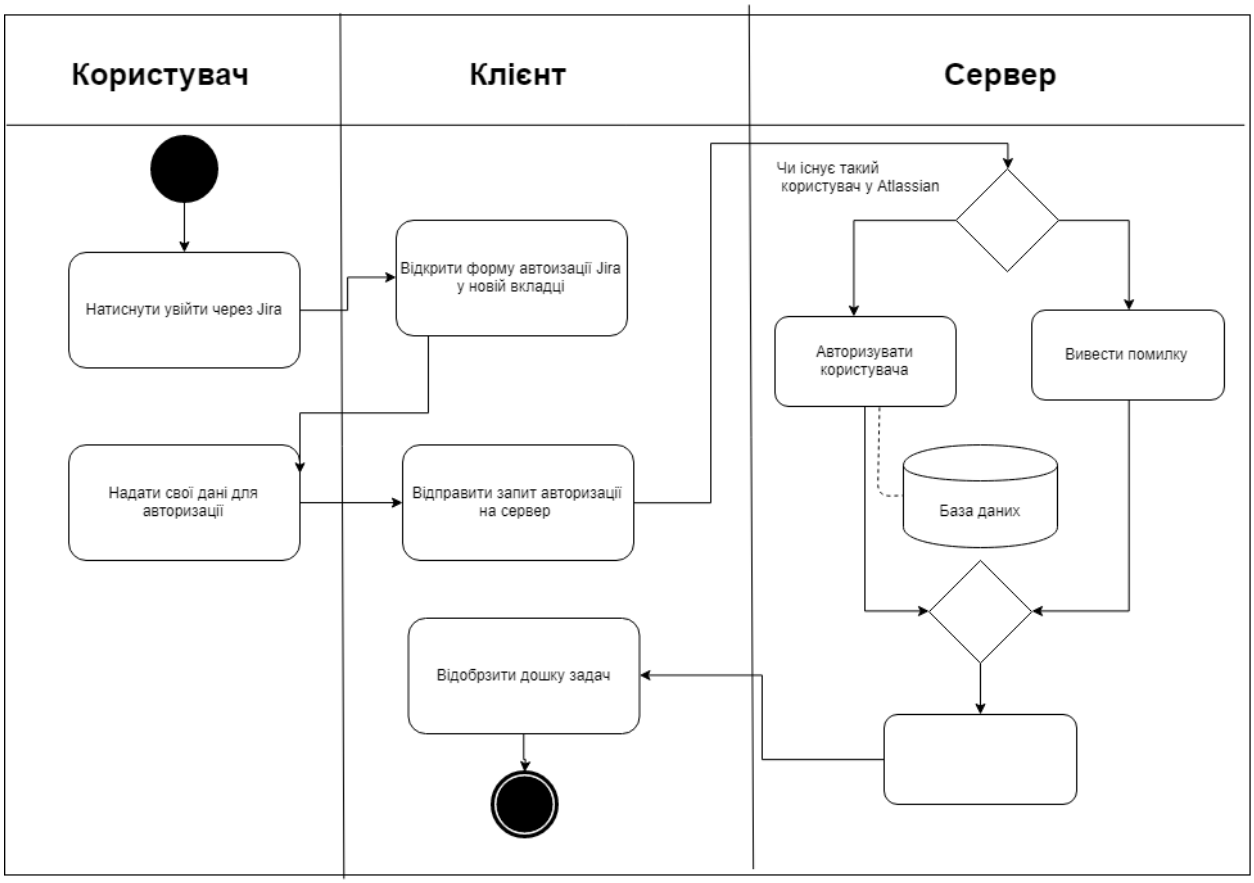

Рисунок 2.1 – Схема бізнес-процесу авторизації користувача через Jira

Сценарій авторизації у конфлюенс (не обов'язково):

- користувач натискае кнопку “авторизуваися у конфлюенс”;
- сервер пренапраправляє запит на сторінку авторизації в
- конфлюенс,де користувач вводить свої дані;
- на сервер надсилаються дані користувача;
- сервер конфлюенса перевіряє, чи є такий користувач у базі;
- якщо так, так користувача авторизовано і він може почати роботу
- пов’язану з інтеграцією;
- якщо ні, виводиться повідомлення про помилку.

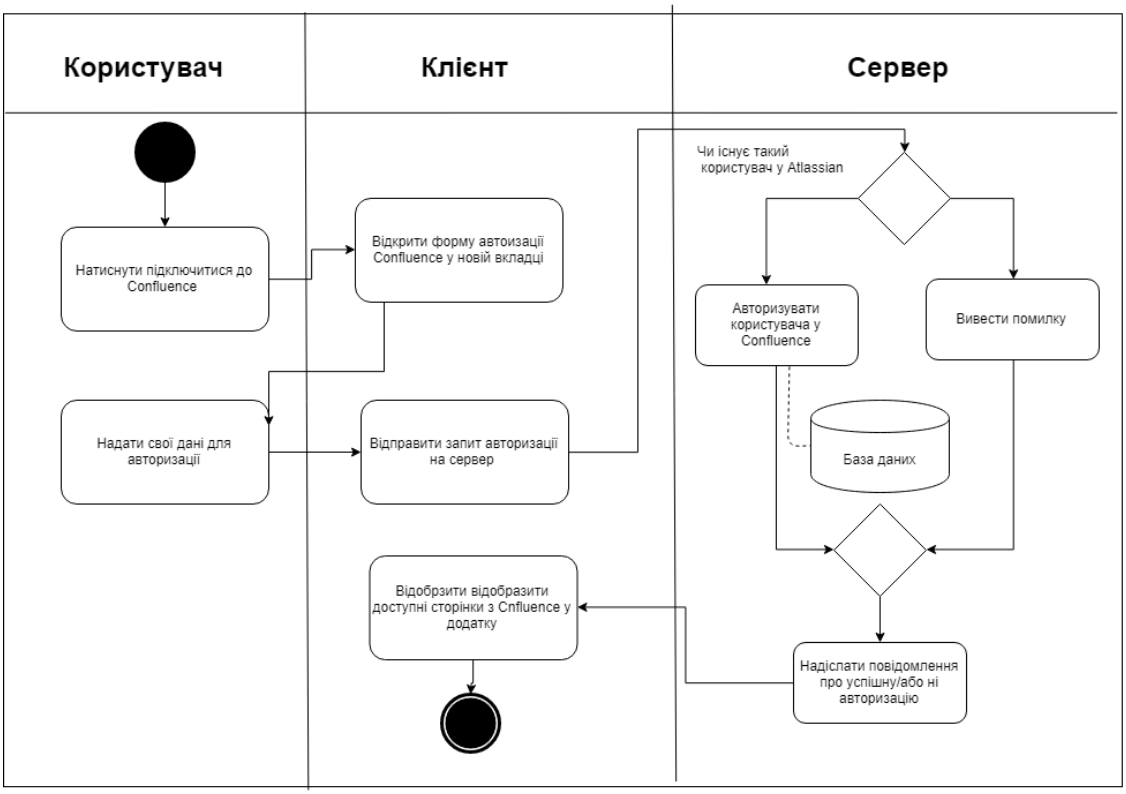

Рисунок 2.2 – Схема бізнес-процесу авторизації користувача у Confluence

Оскільки користувача аторизовано тепер він може повноцінно створювати задачи та керувати їми у додатку.

Сценарій додавання нової задачі:

- користувач натискае кнопку *+*;
- відкриваеться модальне вікно з усіма полями ;
- користувач заповнює всі необхідні поля;
- якщо користувач клікає на поля пов’язані з джирою, або
- конфлюенсом,запит перенаправляється на сервер відповідно джири чи конфлюенса;
- користувач натискає кнопку “SAVE”;
- на сервер надсилаються дані;
- сервер оновлює данні та надсилає клієнту актуальний сптсок
- задач;
- користувач бачить нову добавлену задачу.

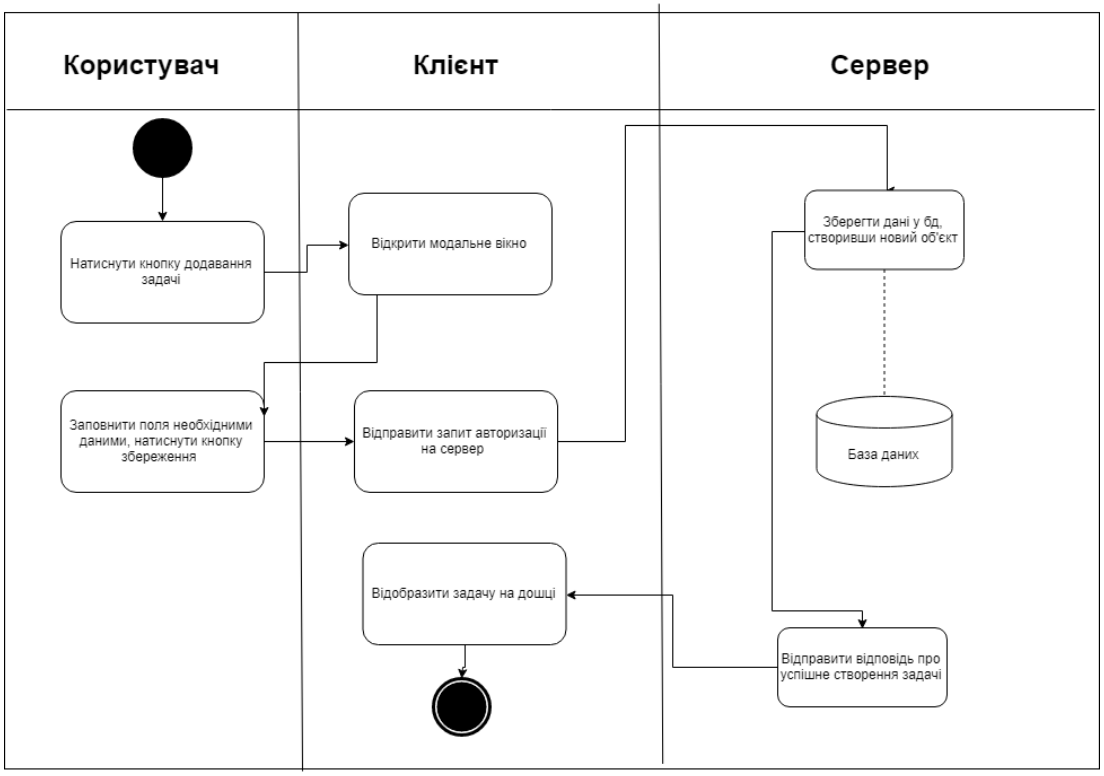

Рисунок 2.2 – Схема бізнес-процесу додавання задачі

Сценарій пріоритизації задач:

- користувач клікає мишкою на задачу, утримує її;
- користувач переносить задачу у інше місце (вище або нижче відносно
- інших задач);
- клієнтська частина змінює значення змінної, що відповідає за номер у
- списку задач;
- запит відправляється на сервер ;
- дані оновлюється;
- пріоритет задачі змінено, користувач бачить актуальну розстановку
- задач.

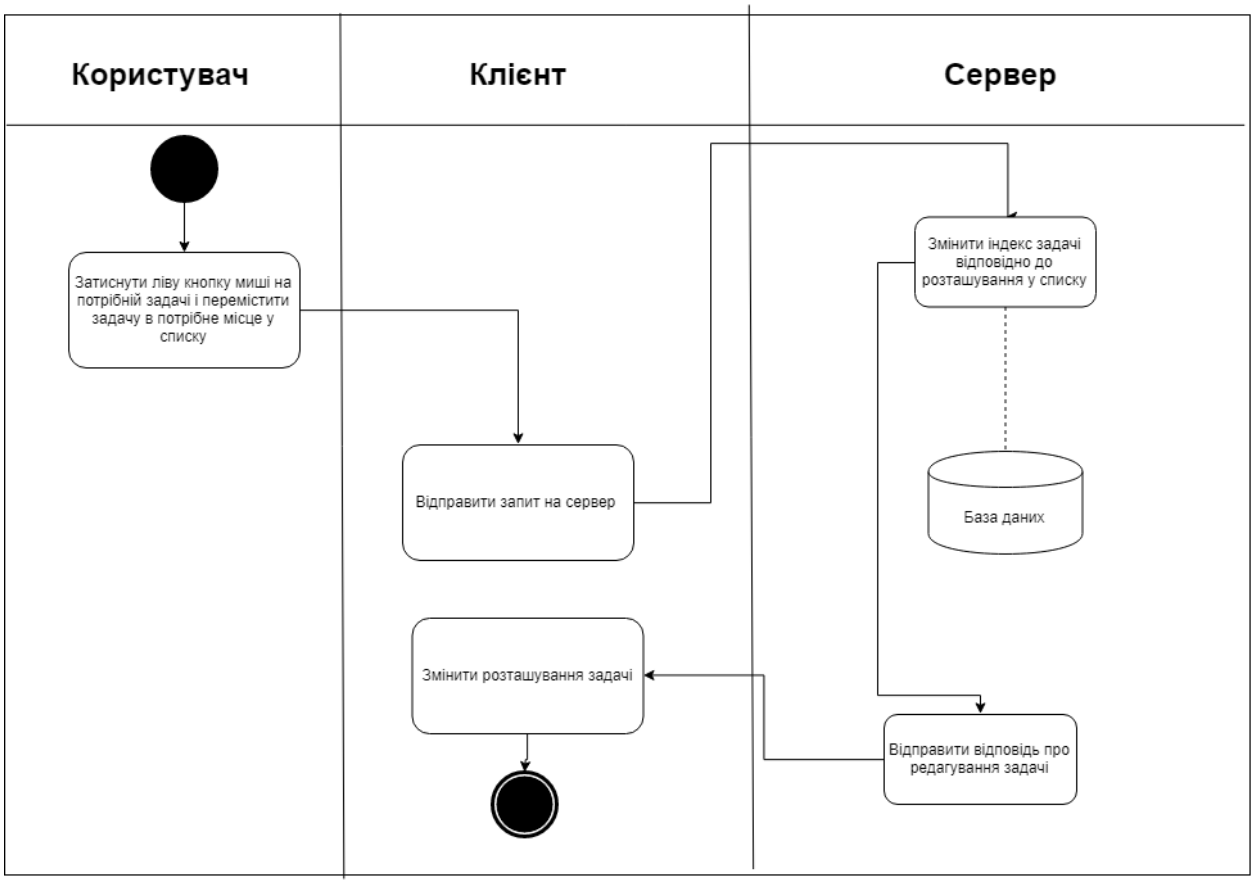

Рисунок 2.4 – Схема бізнес-процесу пріоритизації задач

Сценарій видалення задачі:

- користувач натискає кнопку видалення;
- запит відправляється на сервер;
- сервер обробляє запит та видаляє запис про задачу с бази даних;
- із бази беруться поточні дані;
- клієнт отримує оновлені дані;
- задача зникає з інтерфейсу користувача.

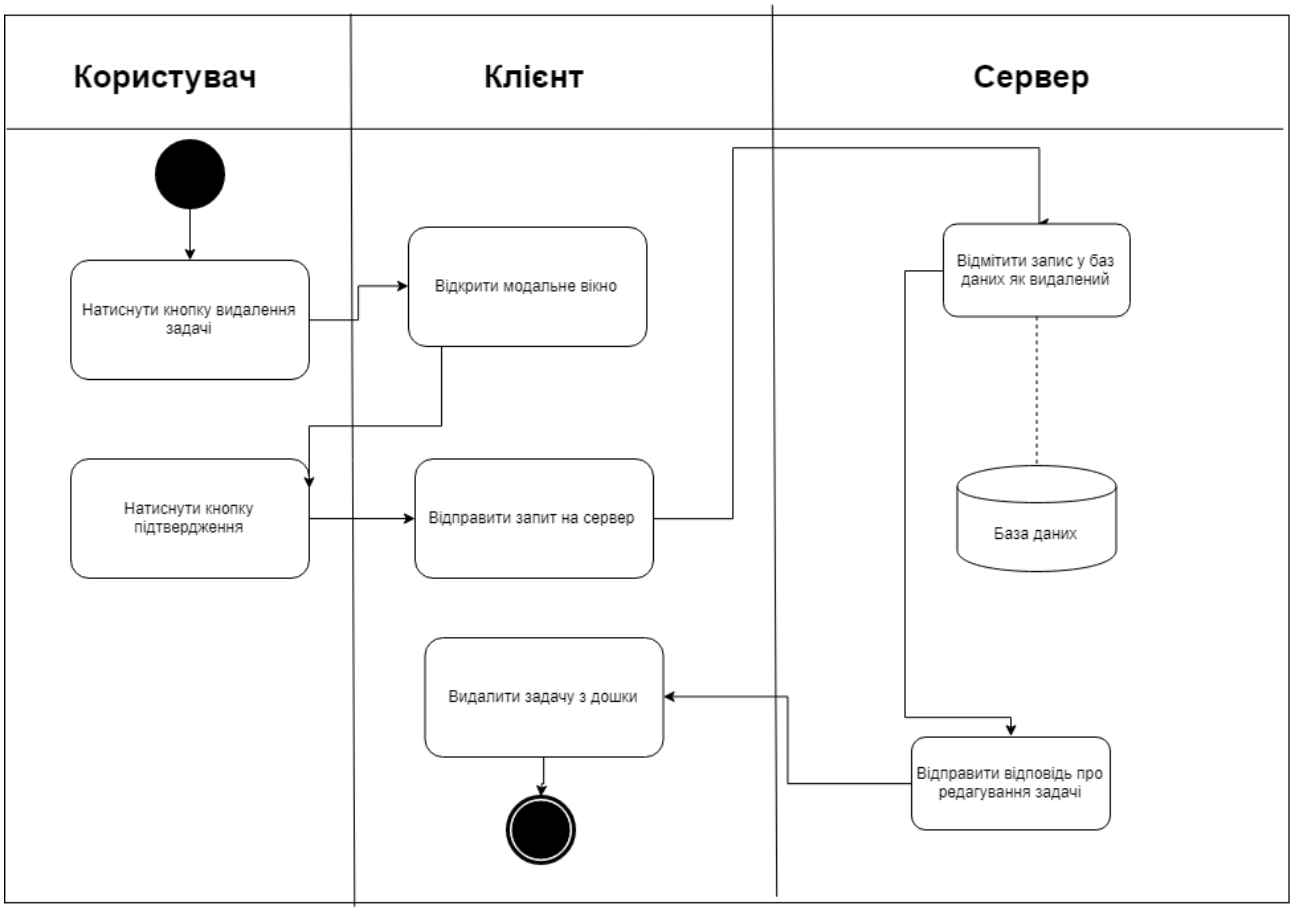

Рисунок 2.5 – Схема бізнес-процесу видалення задач

Сценарій виконання задачі:

- користувач натискає на чек-бокс;
- запит відправляється на сервер;
- сервер обробляє запит та виставляє булеву змінну активності
- задачі у стан false;
- із бази беруться поточні дані;
- клієнт отримує оновлені дані;
- задача зникає з інтерфейсу користувача.

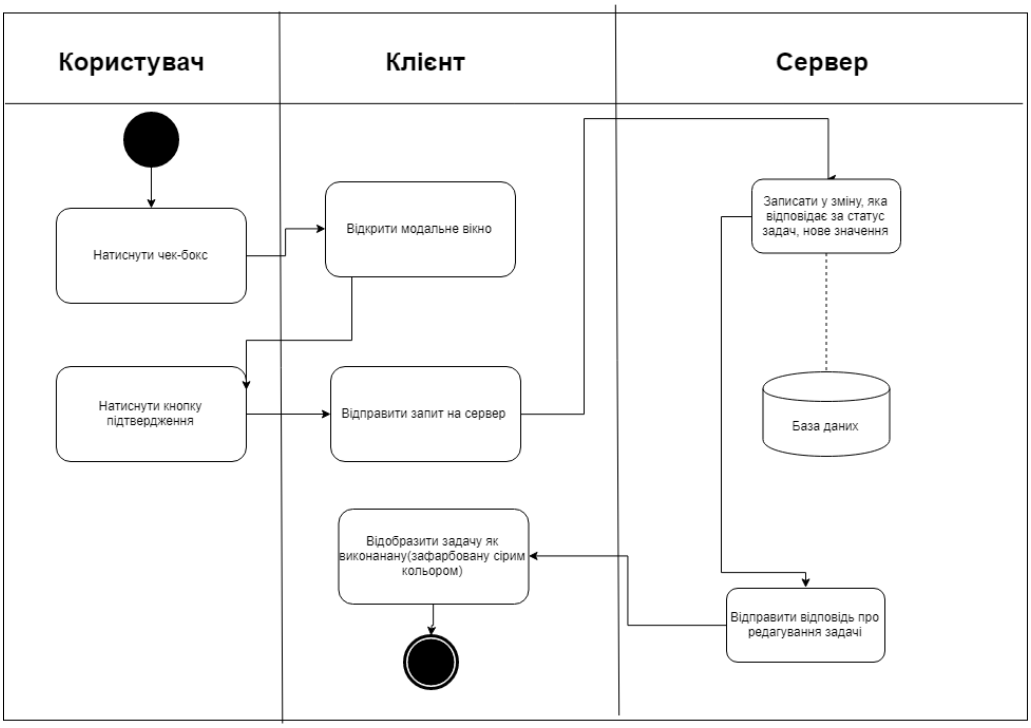

Рисунок 2.6 – Схема бізнес-процесу виконання задач

2.2 Архітектура програмного забезпечення

Для створення цього додатка потрібно сконструювати клієнтську і серверну частини. Серверна частина складається з сервера з бізнес-логікою, бази даних і API для взаємодії з клієнтом. Клієнтський додаток повинен містити графічний інтерфейс для взаємодії з користувачем, бізнес логіку (мало, адже більшість операції буде відбуватися на сервері, а клієнт майже у всіх сценаріях використовується тільки для відображення даних), а також частина взаємодії з мережею через API, є своєрідним прошарком між сервером і клієнтом. Будь-яке звернення до мережі повинно відбуватися через неї.

Для сворення веб-застосунку було використано фреймворк Django мови програмування Python. Для відображення данних на клієнті було використано HTML, CSS, Bootstrap.

Дані передаватимуться через протокол HTTPS, що є розширенням протоколу HTTP з підтримкою криптографічного шифрування. Він використовує криптографічні протоколи SSL та TLS.

2.3 Конструювання програмного забезпечення

База даних

Розробимо базу даних, яка буде містити необхідну інформацію для сервісу.

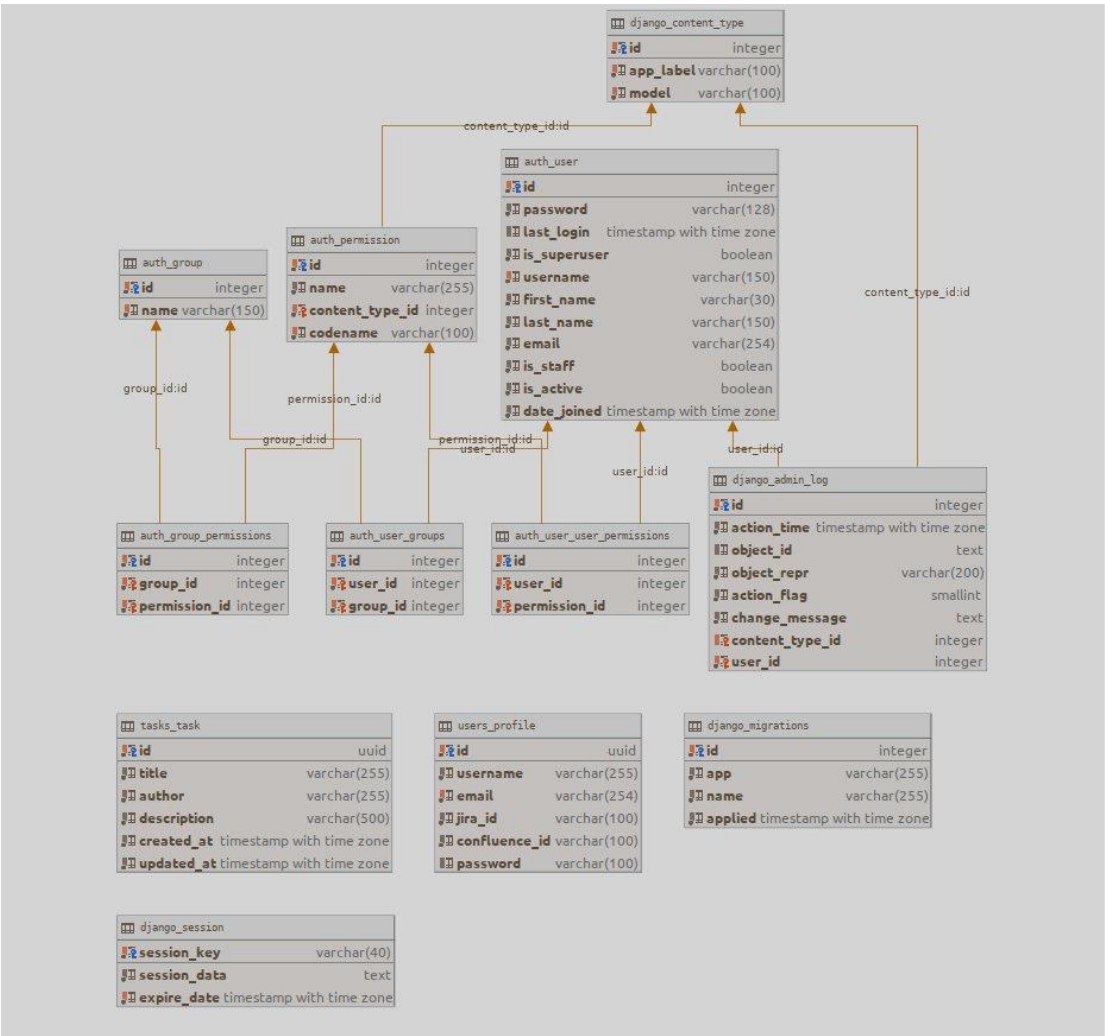

Рисунок 2.7 – Схема бази даних

Основними елементами системи є задача та користувач. Тобто має сенс писати таблиці цих об’єктів у базі даних.

Таблиця 2.1 – Опис таблиці task_task

Поле	Тип	Призначення
Id	Uuid	Унікальний ідентифікатор запису у базі данних.Слугує для ідентифікації задачі у системі.
Title	varchar	Назва задачі. Дає коротке уявлення про те, в чому заключається суть її. Хороша назва задачі має відповідати на питання «Що? Де? Коли?»
Author	varchar	Автор. Дає розуміння про те, хто створив задачу і є її власником.
description	varchar	Опис задачі містить повну інформацію про задачу та деалі її виконання.
created_at	timestamp	Точній час, оли задачу було створенно. Поле є кориним для таймменеджменту.
updated_at	Timestamp	Точний останній час, коли у хадча вносились будь-які зміни.

Таблиця 2.2 – Опис таблиці Users_profole

id	Uuid	Унікальний ідентифікатор запису у базі данних.Слугує для ідентифікації задачі у системі.
Username	Varchar	Ім’я, яке користувач використовує в системі.Слугує ідентифікатором для роботи з користувачем у системі(для зручності при роботі з багатьма користувачами).
Email	Varchar	Електронна андреса користувача.
Jira_id	Varchar	Унікальний ідентифікатор при інтеграціію із сервісом Jira.
Confluence_id	Varchar	Унікальний ідентифікатор при інтеграціію із сервісом Confluence.
password	varchar	Пароль користувача, з якого генеруеться унікальний хеш.

Класи та інтерфейси

Розглянемо основні класи клієнтського застосунку, їх наповнення та інтерфейси.

Фундаментальним типом застосунку є задача– Task. Цей тип має містити всю необхідну інформацію про задачу для– його назву, опис, інтеграцію із сторонніми сервісами,я кщо вона є. Усі ці поля є публічними та немутабельними задля уникнення можливих небажаних змін. Щоб змінити інформацію по задачі буде відправлятися метод PATCH на API додатку.

Логіка додатку базується на представленнях.(рисунок 2.8)

Будь-який об'єкт мови Python, що викликається може бути представленням. Єдина жорстка і необхідна вимога полягає в тому, що викликаємий об'єкт Python повинен приймати об'єкт запиту в якості першого аргументу (зазвичай цей параметр так і називають - request).

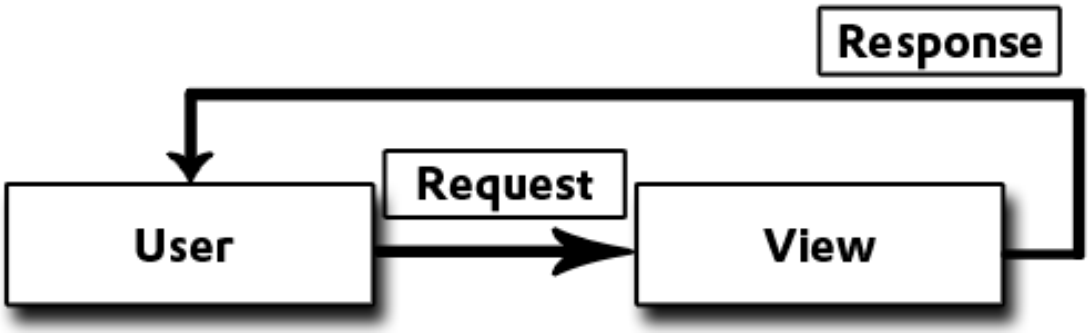

Рисунок 2.7 –Принцип роботи представлень

Педставлення бувають двох видів: Загальні представлення і представлення-класи. Загальні представлення завжди надають якийсь базовий функціонал: візуалізувати шаблон, перенаправити, створити, відредагувати модель і т. д.

Починаючи з версії 1.3, для загальних уявлень в Django з'явилися виставки-класи (CBV); загальні уявлення і CBV надають більш високий рівень абстракції і компонований;

Використані представлення можна переглянути у таблиці 2.2.

Таблиця 2.2 – Представлення-класи використані в проекті

Представлення- класс	Призначення
def get_profile(request)	Повертає профіль користувача за йго унікальим ID із бази даних.
class ProfileCreateView(BSModalCreateView)	Відповідає за будь-які дії із профілем користувача
class CustomLoginView(BSModalLoginView)	Класс що відповідає за функціонал авторизації користувачів додатку. Авторизація у додаток здыйснюеться через сервер Jira.
def task_list(request)	Метод, що выдповыдає за відображення списку задач на дошці. Метод повертає усі наявні задачі для користувача.
class TaskUpdateView(CreateView)	Класс, який відповідає за стоврення нової задачі. Задача містить декілька полів, таких як назва, опис, зв’язок із тікетом в Jira, зв’язок із сторінкою Confluence.
class TaskDeleteView(BSModalDeleteView)	Клас, що відповідає за видалення задачі з дошки. Задача видаляеться та не відображется на дошці користувача.

Продовження таблиці 2.2

def get_docs(request)	Метод, який відображає повний список документів із Confluence , до яких користувач має права доступу(тобто до яки проєктів у Confluence користувач має право доступу)
class JiraTaskChoose(BSModalView)	Клас, що відповідає за інтеграцію з Jira та відображення усіх задач потрібних користувачеві, згідно з відправленим параметрами пошуку.
class ConfluencePageChoose(BSModalView)	Клас, що відповідає за інтеграцію із сервером Confluence.
def check_task(request)	Метод, що відповідає заа дію виконання задачі. Задача виконана, коли користувач натиснув на чек-бокс навпроти задачі.
def archive_task(request)	Метод , що відповідає за архівування задачі.

Для роботи з Jira використовується офіційне Jira API.

API REST API JIRA забезпечують доступ до ресурсів (об'єктів даних) через URI шляхи. Щоб використовувати API REST, ваша програма зробить HTTP-запит і розбере відповідь. API JIRA REST використовує JSON як свій формат зв'язку, а стандартні методи HTTP, такі як GET, PUT, POST і DELETE (див. Описи API нижче, для яких методи доступні для кожного ресурсу). URI для ресурсу API REST API JIRA мають таку структуру:

http: // host: port / context / rest / api-name / api-version / назва-ресурсу

Наразі доступно два назви API, про які йтиметься нижче:

auth - для операцій, пов'язаних з аутентифікацією, і

api - для всього іншого.

Поточна версія API - 2. Однак існує також символьна версія, яка називається останньою, яка вирішується до останньої версії, що підтримується даним екземпляром JIRA. Щоб отримати представлення JSON випуску JRA-9 з публічного відстеження проблем у Atlassian, ви отримаєте доступ до:

https://jira.atlassian.com/rest/api/latest/issue/JRA-9

Існує документ WADL, який містить документацію для кожного ресурсу в JIRA REST API. Він доступний тут.

Розширення в API REST

Щоб мінімізувати мережевий трафік і використання процесора на сервері, API JIRA REST іноді використовує техніку, яка називається розширенням. Коли ресурс REST використовує розширення, то частини цього ресурсу не будуть включені до відповіді JSON, якщо явно не запитано.

Основною одиницею у логіці продукту є задача. Задача описана моделью Task. Повний опис полів моделі наведено в таблиці 2.2

Таблиця 2.3 – Опис моделі Task

Поле	Призначення
id	Унікальний ідентифікатор
title	Назва задчі
author	Користувач, який створив задачу
description	Повний опис задачі запропонований користувачем
created_at	Дата створення задачі

Продовження таблиці 2.3

updated_at	Дата оновлення задачі. Будь-яка зміна інформаціїї у будьякому полі задачі є привідом для оновлення поля updated_at

Діаграму класів наведено в окремому документі.

Розроблені програмні екранні форми наведено на рисунках 2.8-2.13.

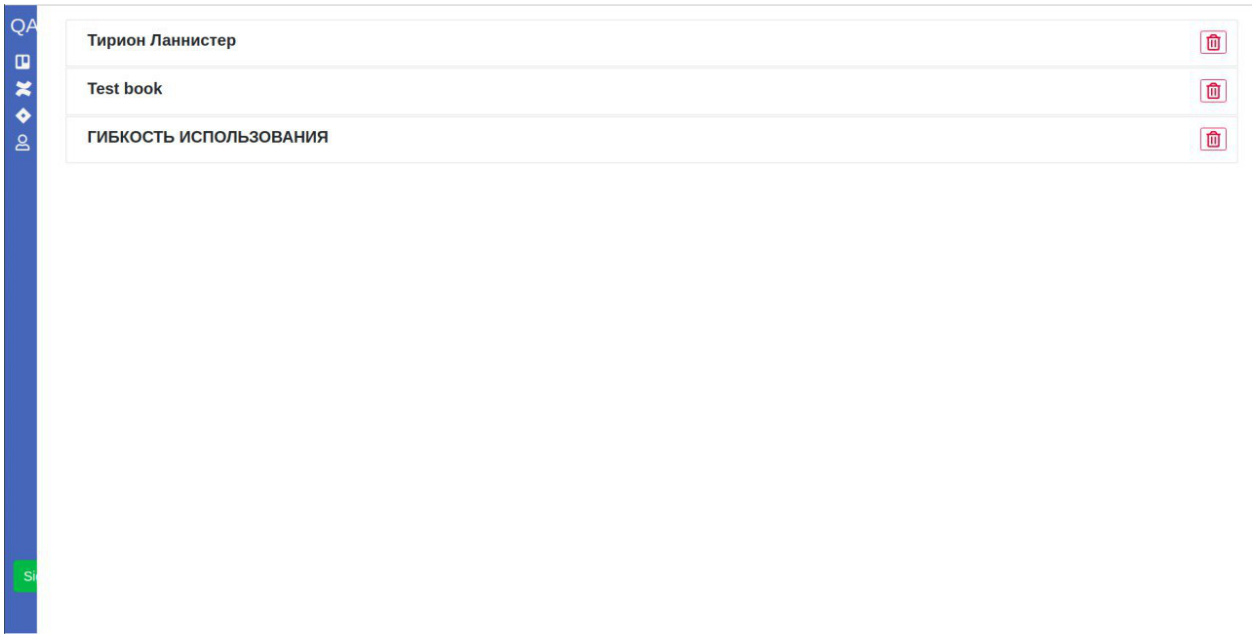

Рисунок 2.8 – Дошка задач

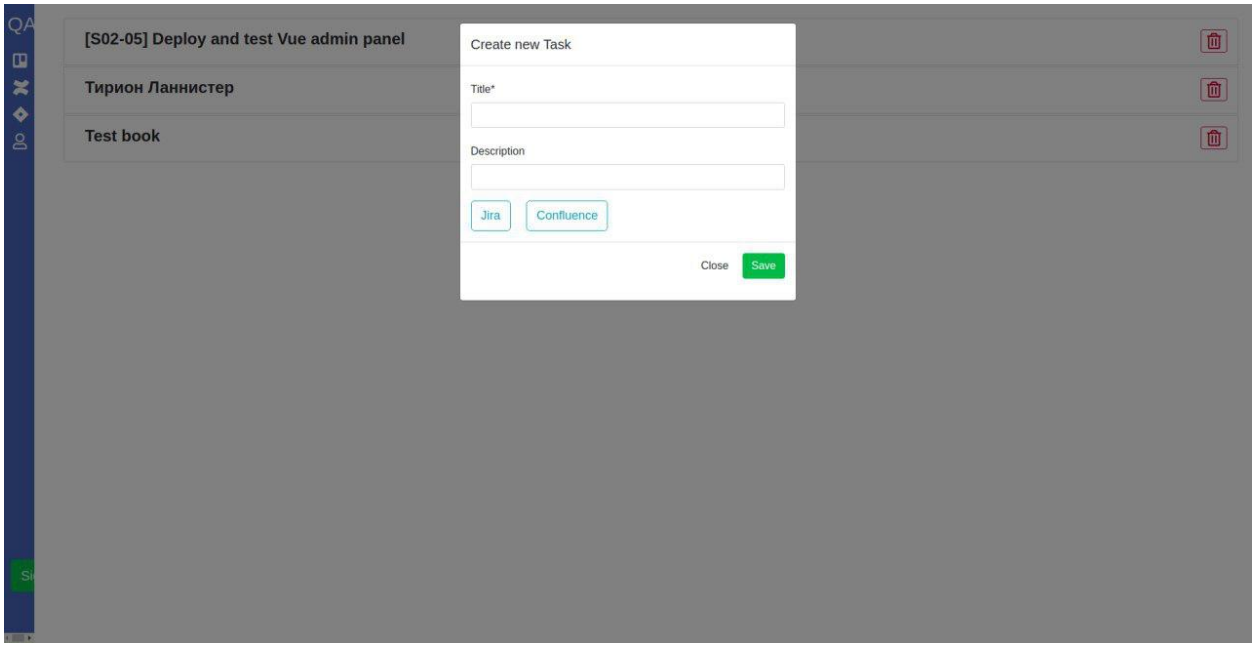

Рисунок 2.9 – Модальне вікно додавання задачі

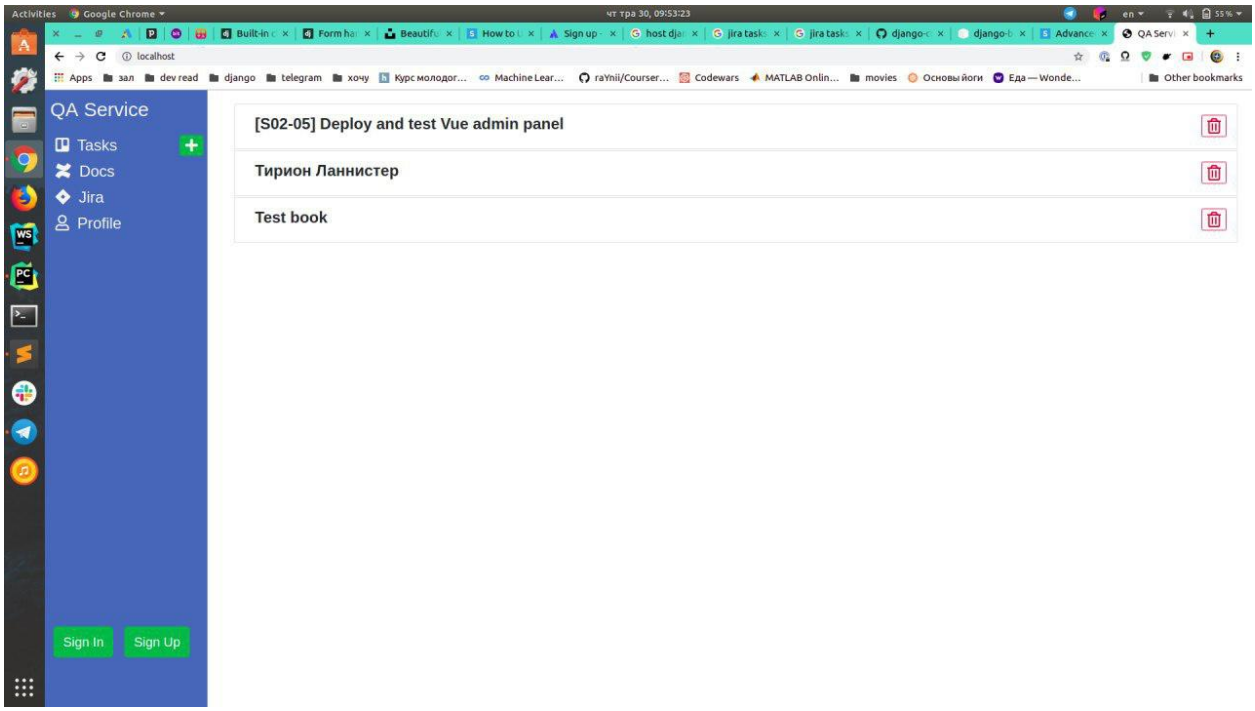

Рисунок 2.10 – Дошка задач з відкритим боковим меню

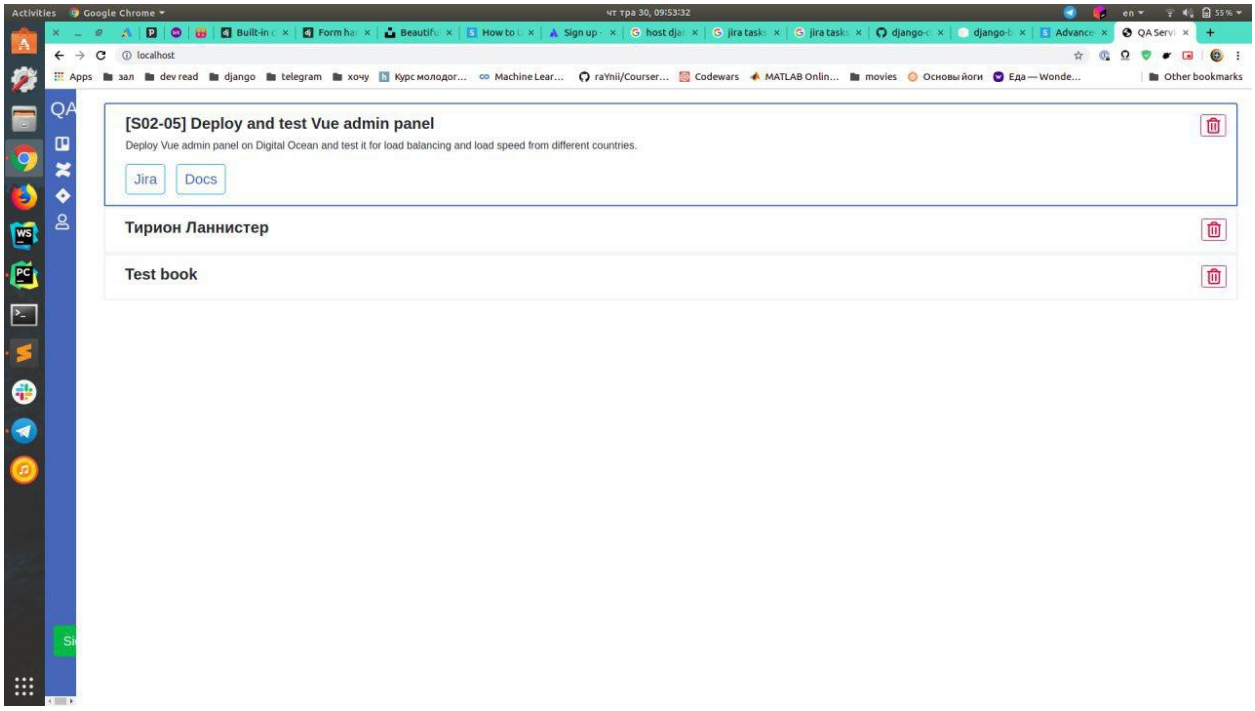

Рисунок 2.11 – Дошка задач з попереднім переглядом задачі

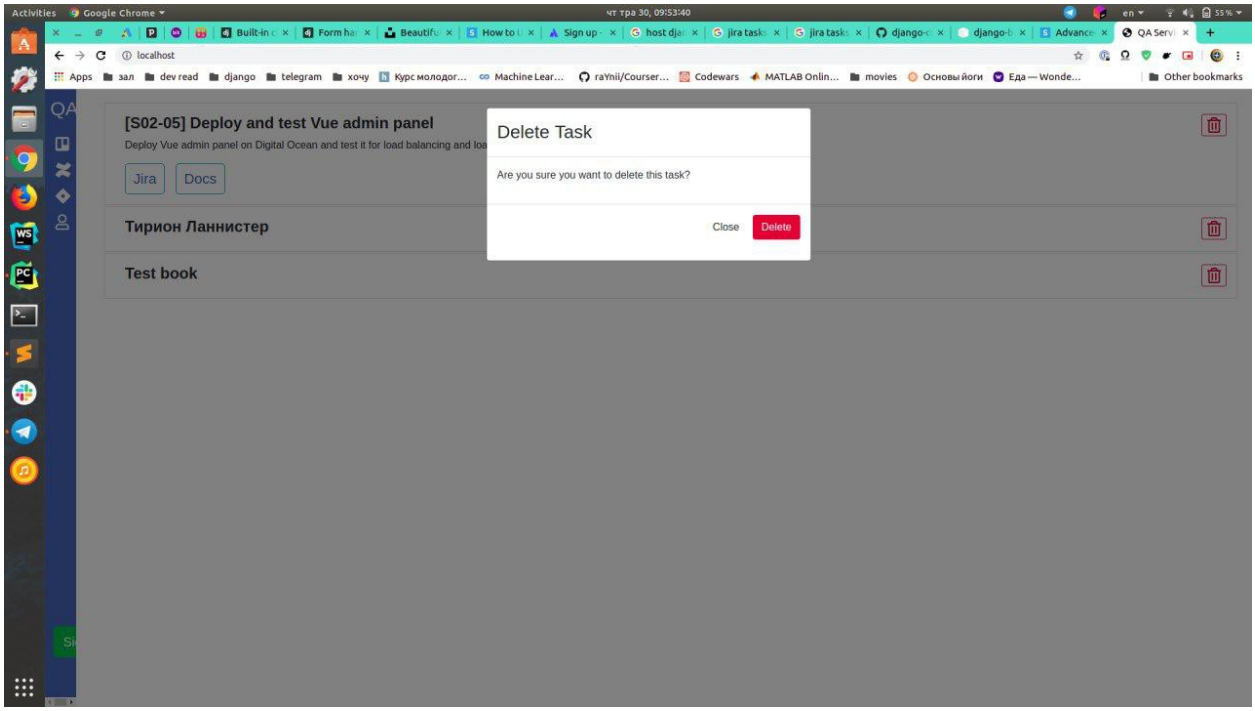

Рисунок 2.12 – Модальне вікно з підтвердження видаленя

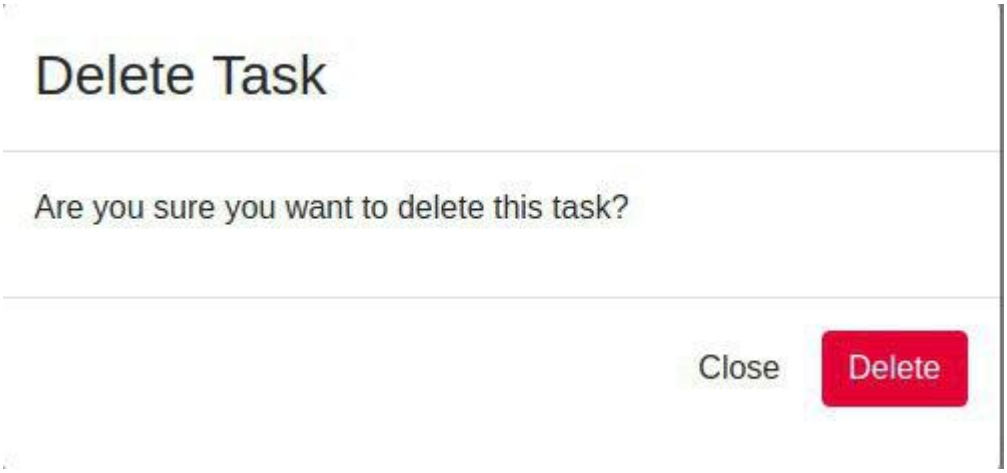

Рисунок 2.13 – Модальне вікно з підтвердження видаленя

2.4 Аналіз безпеки даних

Клієнт-серверна взаємодія здійснюється через протокол HTTPS.

HTTPS - розширення протоколу HTTP для підтримки шифрування з метою підвищення безпеки. Дані в протоколі HTTPS передаються поверх криптографичних протоколів SSL або TLS. На відміну від HTTP з TCP-портом 80, для HTTPS за замовчуванням використовується TCP-порт 443.

Забезпечення безпечної передачі даних необхідно на сайтах, де вводиться і передається конфіденційна інформація (особисті дані користувачів, деталі доступу, реквізити платіжних карт) - на будь-яких сайтах з авторизацією, взаємодією з платіжними системами, поштовими сервісами. Шифрування таких даних дозволить запобігти їх отримання і використання третіми особами.

Для реалізації передачі даних за допомогою HTTPS на веб-сервері, що обробляє запити від клієнтів, повинен бути встановлений спеціальний SSL-сертифікат. Є сертифікати, що захищають тільки один домен. А є сертифікати, які забезпечують захист інформації на всіх піддоменів, і це Wildcard SSL.

Наявність SSL-сертифіката є одним з факторів ранжирування Google, тому перехід на протокол HTTPS підвищує позиції в пошуковій видачі

Google. Хоча на поточний момент цей фактор не має основного значення, його вплив на пошукової результат може збільшуватися в майбутньому.

Вхід у Jira выдбувається на стороні їх серверу, отож додаток не булет нести відповідальності за збереження вхідних даних користувача та його компанії. Буде зберігатися лиш авторизаційний токен у зашифрованому вігляді, токен будуе оновлюватися з певною безпечною періодичністю.

2.5 Висновки до розділу

У цьому розділі були продумані сценарії використання та її реальзація клієнт-серверної взаємодії, розроблені схеми BPMN-діаграм для процессу кожного сценарію та безпосередньо самої архітектури кльент-сервер. Були продумані компоненти програми та їх реалізація.

Також було взято на розглад питання безпеки зберігання данних та їх передачі. Питання було досліджено та були зроблені висновки, щодо оптимального рішення цього питння.ыыыыыыыы

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз якості ПЗ

План тестування ПЗ включає в себе обсяг, підхід, ресурси та план усіх методів тестування. План описує об’єкти та функції що будуть протестовані, тип тестів, ресурси та план необхідний для виконання тестування.

Цілі

У рамках цього плану буде виконано тестування частини продукту, що відповідає за планування процесу тестування та його менеджменту, функціонал календаря.

У даному релізі будуть протестовані наступні функції:

- авторизація користувачів;
- створення та менеджмент тестової документації;
- створення задач;
- редагування задач;
- видалення задач;
- виконання задач;
- зміна пріорітету задачі.

У даному релізі НЕ будуть протестовані:

- інтеграція із багтрекінговую системою;
- інтеграція з системою менеджмента документації.

3.2 Опис процесів тестування

Модульне тестування

В рамках данного плану будуть використані наступні методи тестування:

- функціональне;
- кросс-браузерне.

Функціональне тестування

- перевірка логіну;
- перевірка навігації в лівому меню;
- перевірка пропозицій;
- перевірка кнопок;
- перевірка перетягування та drag-dprop;
- перевірка конфліктів;
- підтвердження спливаючих вікон;
- перевірка валідації;
- перевірка прав доступа.

Кросс-браузерне тестування

Перевірити правильний робочий і користувальницький інтерфейс веб-сайту в різних браузерах.

Браузери:

- Google Chrome 68.0.3440.106;
- Safari (останнє);
- Firefox (останнє);
- IE (останній).

Функціональне тестування

Для функціонального тестування критерієм проходження є успішне виконання кожного пункту тесту. У разі якщо хоча б один пункт не був успішно виконаний – тестування вважається не пройденим. Також витримано період код-фрізу та пройдене регрессійне тестування. Виправлені усі знайдені баги блокери, критичні та мажорні.

Кросс- браузерне тестування

Для кросс-браузерного тестування критерієм проходження є успішне виконання кожного пункту тесту. У разі якщо хоча б один пункт не був успішно виконаний – тестування вважається не пройденим.Також витримано

період код-фрізу та пройдене регрессійне тестування. Виправлені усі знайдені баги блокери, критичні та мажорні.

Данні до тестів

Вхідними даними для компонентного тестування є набори параметрів на яких очікується певний результат, що є вихідними даними даного тесту.

Вхідними даними для функціонального тестування є набори даних та дій які мають привести до певного очікуваного результату.

Вхідними даними для кросс-браузерного тестування є набори браузерів, даних та дій які мають привести до певного очікуваного результату.

Задачі тесту

Кожен тест повинен перевірити як правильність програми у відповідності до умов виконання тесту (test-driven development), так і виявити можливі помилки у роботі.

План виконання

Компонентне тестування повинне виконуватися до інтеграційного, яке, у свою чергу, виконується до тестування швидкодії.

Аппаратна частина

Для успішного виконання тестування потрібна аппаратна платформа що задовільняє наступним умовам:

- процессор x86_64 сумісний або ARM64-сумісний, 2 Ггц тактова
- частота, 4 фізичних або віртуальних ядра;
- 1 Гб оперативної пам’яті;
- Гб постійної пам’яті;
- наявне з’єднання з Інтернет.

За виконання даних умов можливе проведення тестування на локальній машині. Для найбільшої зручності бажано використовувати віртуальний сервер на платформі GCP.

Програмна частина

Для виконання тестування апаратна платформа повинна мати будь-яку операційну систему та браузери: Google Chrome 68.0.3440.106, Safari (остання), Firefox (остання), IE (остання).

Вимоги до безпеки

Для виконання тестування бажано створити в джирі та конфлюенсі тестові проекти, що мають важливої інформації. Інтеграцію для тестів проводити з цими проектами.

Інструменти

Для виконання тестування використовувати наступні програмні інструменти:

- Postman- тестування API;
- Jira - відстеження помилок;
- Confluence - управління документацією.

3.3 Опис контрольного прикладу

Варіанти використання, перевірені в рамках тестування:

- вхід через Jira (таблиця 3.1);
- створити задачу (таблиця 3.2);
- змінити пріорітет задачі (таблиця 3.3);
- виконати задачу (таблиця 3.4).

Таблиця 3.1 – Перевірка можливості входу через Jira

Мета тесту	Перевірка можливості входу через Jira
Початковий стан	Відкрито додаток.
Вхідні дані	Відсутні.
Схема проведення тесту	Натиснути кнопку «Увійти через Jira», у діалоговому вікні натиснути «ОК».
Очікуваний результат	Відкрито нову вкладку з сторінкою авторизації у Jira.
Стан програмного продукту після проведення випробувань	Користувача авторизовано в додатку.

Таблиця 3.2 – Перевірка можливості створення задачі

Мета тесту	Перевірка можливості створення задачі
Початковий стан	Користувача авторизовано.
Вхідні дані	Дані задачі: назва, опис, тікет з Jira(не обов’язково), сторінка з Confluence(не обов’язково).
Схема проведення тесту	Натиснути кнопку створення задачі;у відкритомому модалному вікні заповнити дані; натиснути кнопку збереження.
Очікуваний результат	Задачу відображено на дошці.
Стан програмного продукту після проведення випробувань	У базі даних створено запис. Тепер до задачі можна застосувати усі можливі дії

Таблиця 3.3 – Перевірка можливості зміни пріорітету задачі

Мета тесту	Перевірка можливості зміни пріорітету задачі.
Початковий стан	Користувач авторизований та має дві і більше задачі.
Вхідні дані	Дві і більше задач на дошці.
Схема проведення тесту	Навести курсор миші на потрібну задачу; затиснутиліву кнопку миші;перетягнути задачу у потрібне місце та відпустити ліву кнопку миші.
Очікуваний результат	Розташування у списку на дошцы задач змінено.
Стан програмного продукту після проведення випробувань	Індекс задачі у базі даних змінено. Інтерфейсі порядок задач змінено.

Таблиця 3.4 – Перевірка можливості виконання задачі

Мета тесту	Перевірка можливості виконання задачі
Початковий стан	Користвача авториовано і н дошці є хочаб одна задача.
Вхідні дані	Задача на дошці.
Схема проведення тесту	Натиснути на чек-бокс навпроти задачі.
Очікуваний результат	Чек-бокс відмічений;задача зафарбована сірим кольором.
Стан програмного продукту після проведення випробувань	Параметр, відповідаючий статусу задачі у бд змінено.

3.4 Висновки по розділу

У цьому розділі я побудувала план тестування. Були описані процеси тестування та необхідні вимоги. Також були визначені види тестування та визначені задачі тестування. Основною частиною плану тестування було описати, які компонети підлягають тестуванню, а які ні. Продукт було протестовано за цим планом.

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

Для розгортання сервера використовується Docker.

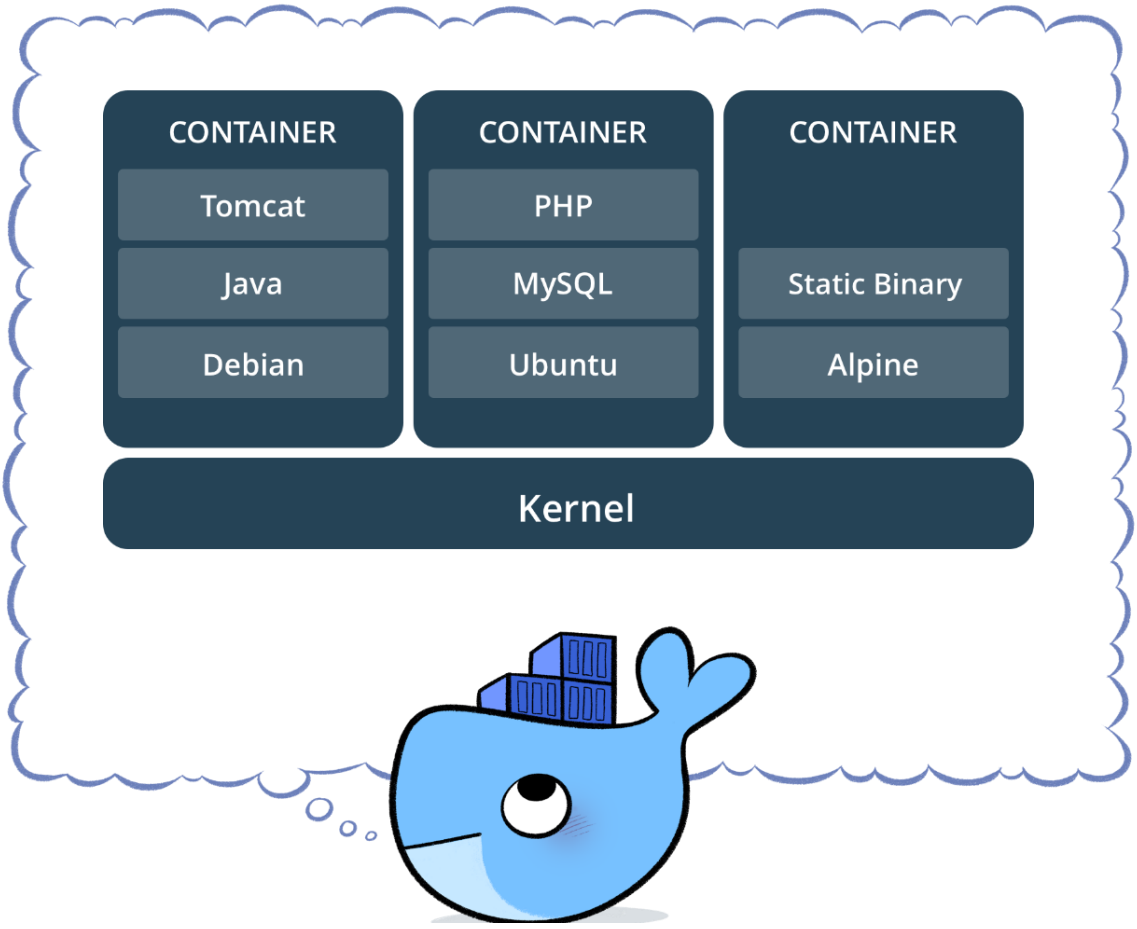

Рисунок 4.1 – Схематичне зображення Докер контейнерів

Docker — інструментарій для управління ізольованими Linux-контейнерами. Docker доповнює інструментарій LXC більш високорівневим API, що дозволяє керувати контейнерами на рівні ізоляції окремих процесів. Зокрема, Docker дозволяє не переймаючись вмістом контейнера запускати довільні процеси в режимі ізоляції і потім переносити і клонувати сформовані для даних процесів контейнери на інші сервери, беручи на себе всю роботу зі створення, обслуговування і підтримки контейнерів. [9].

Для розгортання використовується два докер контейнера: один для бази даних, інший для веб-інтерфейсу.

Контейнер Docker - це запущений образ. Контейнери Docker можна запускати, спиняти, переміщувати і видаляти. Також можна зробити docker commit контейнера, що створить образ з поточного стану контейнера[10].

Докер запускається командою «docker-compose up», в цей момент локально запускаться сервер на 8000 порту (localhost:8000)

Також для авторизації необхідний функціонуючий Facebook-сервер.

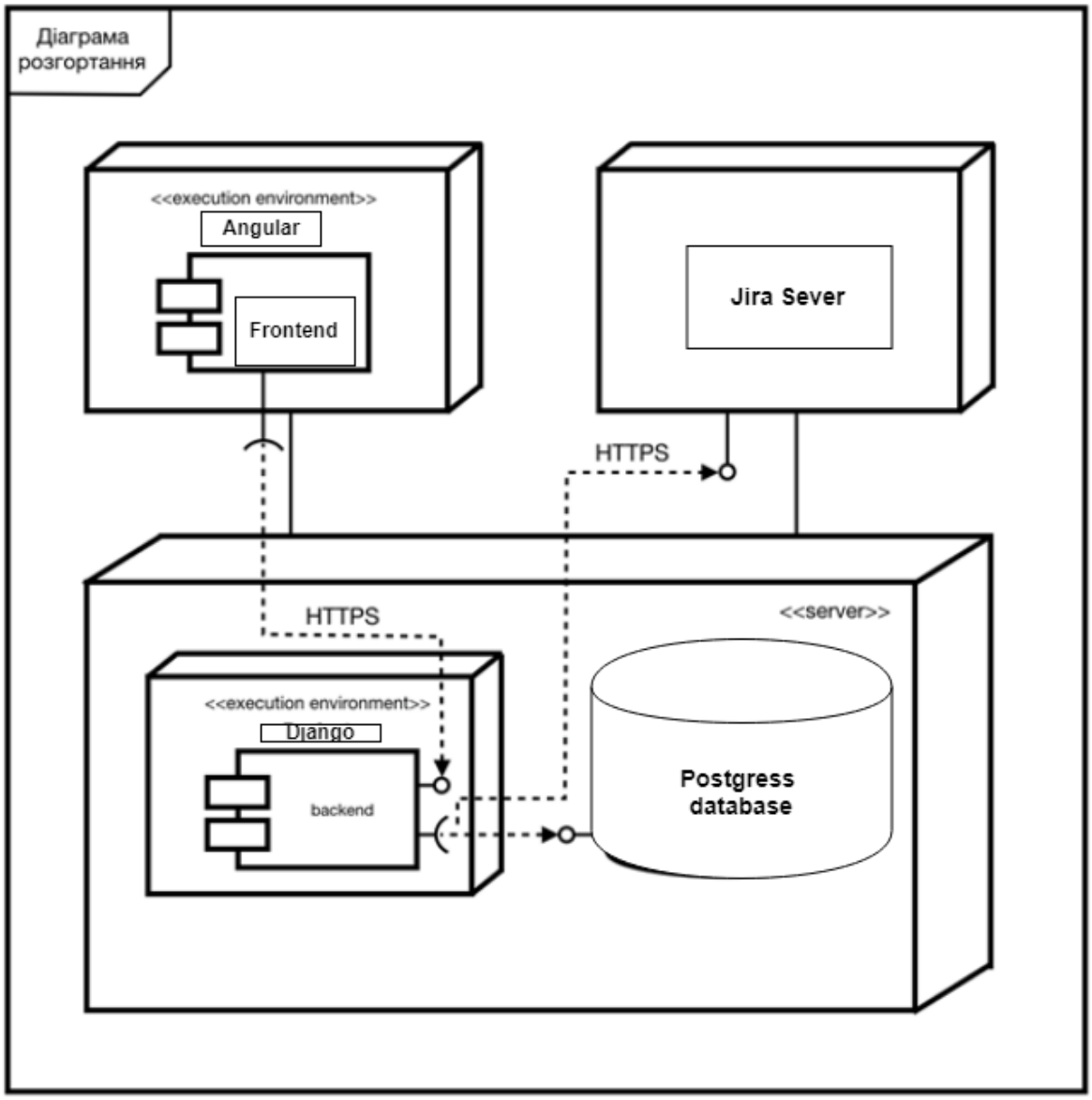

Рисунок 4.2 – Схема структурна розгортання

4.2 Робота з програмним забезпеченням

Детальний покроковий опис дій щодо користування додатком наведено в керівництві користувача. Наведемо опис основних етапів роботи.

Авторизація (передумова – користувач відкрив додаток):

- натиснути «Вхід через Jira»;
- у новій відкритій вкладці авторизуватися у Jira.

Створити задачу (передумова – користувача авторизовано):

- натиснути кнопку додавання задачі;
- заповнити усі необхідні поля у відкритому модальному вікні;
- натиснути кнопку «Зберегти».

Редагування задачу (передумова – на дошці є хоча б одна задача):

- натиснути на кнопку редагування;
- у відкритому модльному вікні змінити потрібні дані;
- натиснути кнопку «Зберегти».

У (передумова – на дошці є дві і більше задач):

- натиснути на вибрану задачу лівою кнопкою миші та утримувати;
- перемістити курсор миші разом із задачею на потрібне місце;
- відпустити ліву кнопку миші, після цього розташування та приорітет
- задачі буде змінено.

Виконати задачу (передумова – на дошці є оча б одна задача):

- натиснути на чек-бокс біля відповідної задачі.

4.3 Супровід програмного забезпечення

Для аналізу помилок використовується сервіс Sentry. Цей сервіс збирає статистику помилок програмного забезпечення разом з усією інформацією про час, частоту повторення, деталі, на якому ендпоінті була помилка та іншу статистику.

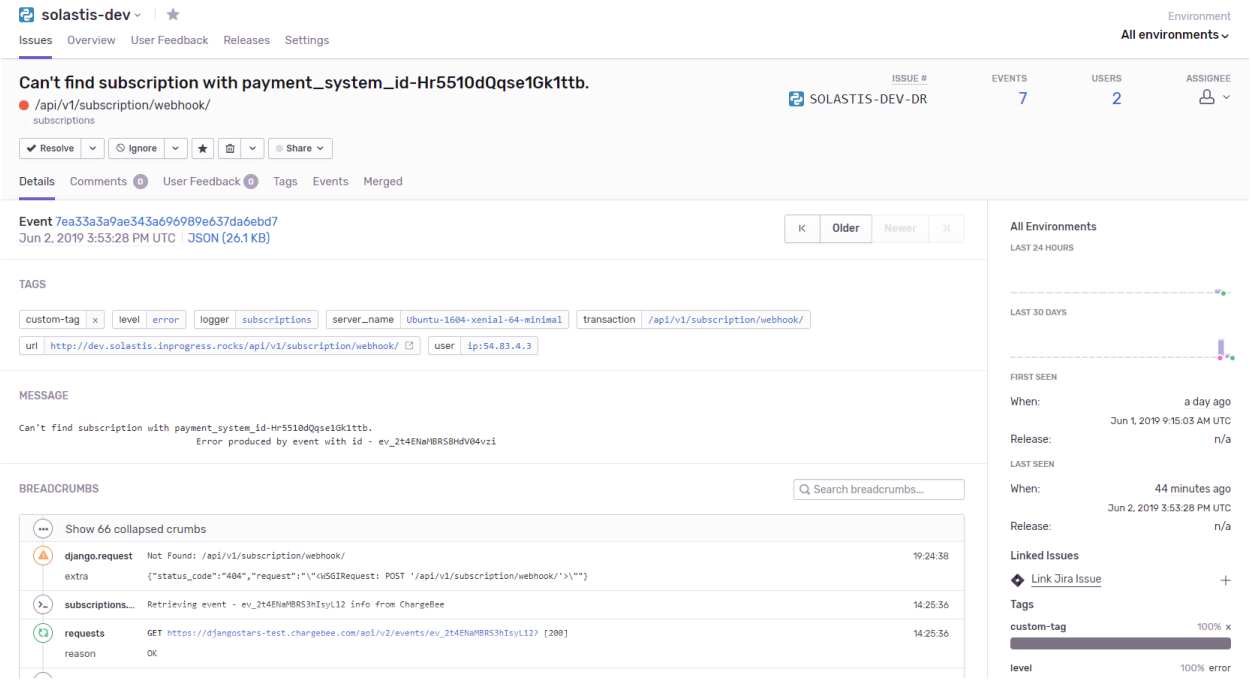

Рисунок 4.3 – Аналітика в Sentry

4.4 Висновки по розділу

У цьому розділі були сформовані вимоги для розгортання, підтримки та користування створеним програмним забезпеченням.

Визначено необхідний інструмент для запуску та розгортання серверної частини.

Створено наочну діаграму розгортання, що відображає зв’язки між елементами.

Створено інструкцію користувача та наведено опис основних етапів роботи із додатком.

Визначено програмне забезпечення для детекції та аналізу помилок, та збору аналітики.

Таким чином, система на даному етапі готова до запуску та повноцінного функціонування.

5 ВИСНОВКИ

У ході дипломного проекту було проведено аналіз проблем у роботі тестувальників.

Було спроектовано і розроблено кліент-сервену архітектуру та створено додаток. Розроблена система додає ряд переваг, які не було реалізовано в проаналізованих існуючих аналогах, а саме:

- інтеграція з Jira;
- інтеграція Confluence;
- більш зручний та не перевантажений інтерфейс.

Був розроблений план та стратегія тестування, на основі яких здійснено тестування, усі недольки були виправлені та систему було покращено.

Було розроблено схеми процесів робти додатку, варіантів використань, а також керівництво для користувачів.

ПЕРЕЛІК ПОСИЛАНЬ

1) 7 best survey tools [Електронний ресурс]: (Стаття) / Wordstream. – Електрон. дан. (1 файл). – 2018. – Режим доступу: https://www.wordstream.com/blog/ws/2014/11/10/best-online-survey-tools. – Назва з екрана.

2) Bcrypt [Електронний ресурс]: (Стаття) / Wikipedia. – Електрон. дан. (1 файл). – 2018. – Режим доступу: https://uk.wikipedia.org/wiki/Bcrypt. – Назва з екрана.

3) Password storage cheat sheet [Електронний ресурс]: (Стаття) / Owasp. – Електрон. дан. (1 файл). – 2018. – Режим доступу: https://www.owasp.org/index.php/Password_Storage_Cheat_Sheet#Use_a_cryptographically_strong_credential-specific_salt. – Назва з екрана.

4) TCP [Електронний ресурс]: (Стаття) / Wikipedia. – Електрон. дан. (1 файл). – 2018. – Режим доступу: https://uk.wikipedia.org/wiki/Transmission_Control_Protocol. – Назва з екрана.

5) TLS [Електронний ресурс]: (Стаття) / Wikipedia. – Електрон. дан. (1 файл). – 2018. – Режим доступу: https://uk.wikipedia.org/wiki/Transport_Layer_Security. – Назва з екрана.

6) Docker [Електронний ресурс]: (Стаття) / Вікіпедія. – Електрон. дан. (1 файл). – 2017. – Режим доступу: https://uk.wikipedia.org/wiki/Docker

7) Docker контейнер [Електронний ресурс]: (Стаття) / Вікіпедія. – Електрон. дан. (1 файл). – 2017. – Режим доступу: https://uk.wikipedia.org/wiki/Docker

8) API [Електронний ресурс]: (Стаття) /. Quality-Assurance– Електрон. дан. (1 файл). – 2017. – Режим доступу:https://www.quality-assurance-group.com/korotko-pro-api-jogo-testuvannya/

9) Django for APIs: Build web APIs with Python and Django [Електронний ресурс]: (Книга) / Django for APIs. – Електрон. дан. (1 файл). – 2018. – Режим доступу: https://www.amazon.com/REST-APIs-Django-powerful-Python-ebook/dp/B07DR9XS6L/ref=sr_1_1?ie=UTF8&qid=1541969595&sr=8-1&keywords=django+python. – Назва з екрана.

10) Django. Розробка веб-додатків на Python — Джефф Форсье, Пол Биссекс, Уэсли Дж. Чан [Електронний ресурс]: (Книга) / Django. Розробка веб-додатків на Python — Джефф Форсье, Пол Биссекс, Уэсли Дж. Чан – Електрон. дан. (1 файл). – 2018. – Режим доступу: https://www.ozon.ru/context/detail/id/141734375/– Назва з екрана.

Факультет інформатики та обчислювальної техніки
Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”
В.о. завідувача кафедри
_____ О.А. Павлов
“___” _____ 2019 р.

Програмне забезпечення підтримки діяльності тестувальника

Опис програми

КПІ.ІП-5207.045440.02.13

“ПОГОДЖЕНО”
Керівник проекту:
_____ О.А. Халус

Нормоконтроль:
_____ К.І. Ліщук

Виконавець:
_____М.Д. Ільїна

Київ – 2019 року

Тексти програмного коду

Програмне забезпечення підтримки діяльності тестувальника

(Найменування програми (документа))

CD-R

(Вид носія даних)

13 арк, 232 Кб

(Обсяг програми (документа) , арк., Кб)

Київ - 2019

```
class Task(models.Model):
    id = models.UUIDField(primary_key=True, default=uuid.uuid4, editable=False)
    title = models.CharField(max_length=255)
    author = models.CharField(max_length=255)
    description = models.CharField(max_length=500, blank=True)
    created_at = models.DateTimeField(auto_now_add=True, editable=False)
    updated_at = models.DateTimeField(auto_now=True)

    def __str__(self):
        return self.title

    # def save(self, *args, **kwargs):
    #     """ On save, update timestamps """
    #     if not self.id:
    #         self.created_at = timezone.now()
    #     self.updated_at = timezone.now()
    #     return super(Task, self).save(*args, **kwargs)
```

```
from django.urls import path
from . import views

urlpatterns = [
    path('', views.task_list, name='task_list'),
    path('create/', views.TaskUpdateView.as_view(), name='create_task'),
    path('delete/<uuid:pk>', views.TaskDeleteView.as_view(), name='delete_task'),
    # path('task/<uuid:pk>/', views.task_details, name='task_details'),
    # path('task/new', views.task_edit, name='task_edit'),
    # path('task/<uuid:pk>/edit/', views.task_edit, name='task_edit'),
]
```

```
from django.shortcuts import render
from django.urls import reverse_lazy
from django.views.generic import CreateView

from bootstrap_modal_forms.generic import (BSModalCreateView, BSModalDeleteView)

from .forms import TaskForm
from .models import Task
```

```
def task_list(request):
    tasks = Task.objects.all().order_by('-created_at')

    return render(request, 'tasks/task_list.html', {'tasks': tasks})

# class TaskCreateView(BSModalCreateView):
#     template_name = 'tasks/create_task.html'
#     form_class = TaskForm
#     success_url = reverse_lazy(task_list)
#

class TaskUpdateView(CreateView):
    model = Task
    fields = ('title', 'description')
    template_name = 'tasks/update_task.html'
    success_url = reverse_lazy(task_list)


class TaskDeleteView(BSModalDeleteView):
    model = Task
    template_name = 'tasks/delete_task.html'
    success_message = 'Success: task was deleted'
    success_url = reverse_lazy(task_list)
from .models import Task
from bootstrap_modal_forms.forms import BSModalForm


class TaskForm(BSModalForm):
    class Meta:
        model = Task
        fields = ['title',
                  'author',
                  'description'
                  ]


# from django import forms
#
# from crispy_forms.helper import FormHelper
# from crispy_forms.layout import Layout, Div, Submit, HTML, Button, Row, Field
# from crispy_forms.bootstrap import AppendedText, PrependedText, FormActions
#
#
```

```
# class MessageForm(forms.Form):
#     text_input = forms.CharField()
#
#     textarea = forms.CharField(
#         widget = forms.Textarea(),
#     )
#
#     radio_buttons = forms.ChoiceField(
#         choices = (
#             ('option_one', "Option one is this and that be sure to include why it's great"),
#             ('option_two', "Option two can is something else and selecting it will deselect option one")
#         ),
#         widget = forms.RadioSelect,
#         initial = 'option_two',
#     )
#
#     checkboxes = forms.MultipleChoiceField(
#         choices = (
#             ('option_one', "Option one is this and that be sure to include why it's great"),
#             ('option_two', 'Option two can also be checked and included in form results'),
#             ('option_three', 'Option three can yes, you guessed it also be checked and included in form results')
#         ),
#         initial = 'option_one',
#         widget = forms.CheckboxSelectMultiple,
#           help_text = "<strong>Note:</strong> Labels surround all the options for much larger click areas and a more usable
form.",
#     )
#
#     appended_text = forms.CharField(
#         help_text = "Here's more help text"
#     )
#
#     prepended_text = forms.CharField()
#
#     prepended_text_two = forms.CharField()
#
#     multicolon_select = forms.MultipleChoiceField(
#         choices = (('1', '1'), ('2', '2'), ('3', '3'), ('4', '4'), ('5', '5')),
#     )
#
#     # Uni-form
#     helper = FormHelper()
#     helper.form_class = 'form-horizontal'
```

```
#     helper.layout = Layout(
#         Field('text_input', css_class='input-xlarge'),
#         Field('textarea', rows="3", css_class='input-xlarge'),
#         'radio_buttons',
#         Field('checkboxes', style="background: #FAFAFA; padding: 10px;"),
#         AppendedText('appended_text', '.00'),
#                 PrependedText('prepended_text', '<input type="checkbox" checked="checked" value="" id="" name="">',
active=True),
#         PrependedText('prepended_text_two', '@'),
#         'multicolon_select',
#         FormActions(
#             Submit('save_changes', 'Save changes', css_class="btn-primary"),
#             Submit('cancel', 'Cancel'),
#         )
#     )# Generated by Django 2.2 on 2019-04-14 15:17

from django.db import migrations, models
import uuid


class Migration(migrations.Migration):

    initial = True

    dependencies = [
    ]

    operations = [
        migrations.CreateModel(
            name='Task',
            fields=[
                ('id', models.UUIDField(default=uuid.uuid4, editable=False, primary_key=True, serialize=False)),
                ('title', models.CharField(max_length=255)),
                ('author', models.CharField(max_length=255)),
                ('description', models.CharField(max_length=500)),
                ('created_at', models.DateTimeField(editable=False)),
                ('updated_at', models.DateTimeField()),
            ],
        ),
    ]


import uuid
```

```
from django.db import models


class Profile(models.Model):
    id = models.UUIDField(primary_key=True, default=uuid.uuid4, editable=False, unique=True)
    username = models.CharField(max_length=255)
    password = models.CharField(max_length=100, blank=True, null=True)
    email = models.EmailField(max_length=254, unique=True)
    jira_id = models.CharField(max_length=100, blank=True)
    confluence_id = models.CharField(max_length=100, blank=True)
    USERNAME_FIELD = 'username'

    def __str__(self):
        return self.username

from django.shortcuts import render
from django.urls import reverse_lazy
from .forms import CustomCreationForm, CustomAuthenticationForm
from bootstrap_modal_forms.generic import BSModalCreateView, BSModalLoginView

from .models import Profile


def get_profile(request):
    # profile = Profile.objects.get(id=request.POST['id'])
    profile = Profile.objects.all().first()

    return render(request, 'users/profile.html', {'profile': profile})


class ProfileCreateView(BSModalCreateView):
    template_name = 'users/create_user.html'
    form_class = CustomCreationForm
    success_url = reverse_lazy('get_profile')


class CustomLoginView(BSModalLoginView):
    authentication_form = CustomAuthenticationForm
    template_name = 'users/auth.html'
    extra_context = dict(success_url=reverse_lazy('get_profile'))
from django.contrib.auth.forms import UserCreationForm
from bootstrap_modal_forms.mixins import PopRequestMixin, CreateUpdateAjaxMixin
from .models import Profile
```

```
from django.contrib.auth.forms import AuthenticationForm
from django.contrib.auth.models import User

class CustomCreationForm(PopRequestMixin, CreateUpdateAjaxMixin, UserCreationForm):
    class Meta:
        model = User
        fields = [
                'email',
                'username',
                'password1',
                'password2'
                ]


class CustomAuthenticationForm(AuthenticationForm):
    class Meta:
        model = User
        fields = ['username',
                'password'
                ]

# Generated by Django 2.2 on 2019-05-18 19:10

from django.db import migrations, models
import uuid


class Migration(migrations.Migration):

    initial = True

    dependencies = [
    ]

    operations = [
        migrations.CreateModel(
            name='Profile',
            fields=[
                ('id', models.UUIDField(default=uuid.uuid4, editable=False, primary_key=True, serialize=False, unique=True)),
                ('username', models.CharField(max_length=255)),
                ('email', models.EmailField(max_length=254, unique=True)),
                ('jira_id', models.CharField(blank=True, max_length=100)),
                ('confluence_id', models.CharField(blank=True, max_length=100)),
```

```
            ],
        ),
    ]

# Generated by Django 2.2.1 on 2019-05-20 19:04

from django.db import migrations, models


class Migration(migrations.Migration):

    dependencies = [
        ('users', '0001_initial'),
    ]

    operations = [
        migrations.AddField(
            model_name='profile',
            name='password1',
            field=models.CharField(blank=True, max_length=100, null=True),
        ),
        migrations.AddField(
            model_name='profile',
            name='password2',
            field=models.CharField(blank=True, max_length=100, null=True),
        ),
    ]
"""
Django settings for diploma project.

Generated by 'django-admin startproject' using Django 2.0.7.

For more information on this file, see
https://docs.djangoproject.com/en/2.0/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/2.0/ref/settings/
"""

import os

# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
```

```
# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/2.0/howto/deployment/checklist/

# Application definition
# APPS
DJANGO_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',

]
THIRD_PARTY_APPS = [
    'crispy_forms',
    'bootstrap_modal_forms',
    'widget_tweaks',
]
LOCAL_APPS = [
    'tasks',
    'documentation',
    'users',
]

INSTALLED_APPS = DJANGO_APPS + THIRD_PARTY_APPS + LOCAL_APPS

CRISPY_TEMPLATE_PACK = 'bootstrap4'

# MIDDLEWARE
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'diploma.urls'
```

```
# TEMPLATES
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'diploma.wsgi.application'



# Database
# https://docs.djangoproject.com/en/2.0/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'HOST': 'db',
        'NAME': 'dev',
        'USER': 'docker',
        'PASSWORD': 'docker'
    }
}



# Password validation
# https://docs.djangoproject.com/en/2.0/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
```

```
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]


# Internationalization
# https://docs.djangoproject.com/en/2.0/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True

USE_TZ = True


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/2.0/howto/static-files/

STATIC_URL = '/static/'
STATIC_ROOT = os.path.join(BASE_DIR, 'static')


# Pull base image
FROM python:3

# Set environment varibles
ENV PYTHONDONTWRITEBYTECODE 1
ENV PYTHONUNBUFFERED 1

RUN apt-get update && apt-get upgrade -y && apt-get autoremove \
    && apt-get install -y --no-install-recommends \
        postgresql-client \
    && rm -rf /var/lib/apt/lists/*
```

```
# Set work directory
RUN mkdir -p /diploma
WORKDIR /diploma

# Install dependencies
RUN pip install --upgrade pip
COPY diploma/requirements/base.txt .
RUN pip install -r base.txt

# Copy project
COPY . /diploma/

# Server
#EXPOSE 8000
STOPSIGNAL SIGINT
```

```
version: "3"

services:
  web:
    build: .
    command: python manage.py runserver 0.0.0.0:8000
    volumes:
      - .:/diploma
    depends_on:
      - db
    ports:
      - "80:8000"
    restart: on-failure
  db:
    image: postgres
    environment:
      POSTGRES_DB: dev
      POSTGRES_PASSWORD: docker
      POSTGRES_USER: docker
    ports:
      - "5432:5432"
    volumes:
      - postgres_data:/var/lib/postgresql/data/
volumes:
  postgres_data:
```

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ О.А. Павлов

“___” _____ 2019 р.

Програмне забезпечення підтримки діяльності тестувальника

Керівництво користувача

КПІ.ІП-5207.045440.05.34

“ПОГОДЖЕНО”

Керівник проекту:

_____ О.А. Халус

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____М.Д. Ільїна

Київ – 2019 року

ЗМІСТ

1 ІНСТРУКЦІЯ КОРИСТУВАЧА

1.1 Авторизація

Для того, щоб користуватися додатком необхідно пройти етап авторизації. Першим кроком буде – натиснути на кнопку «Увійти через Jira», буде відкрита нова вкладка у браузері із формою логіна Jira(Рисунок 1.1).

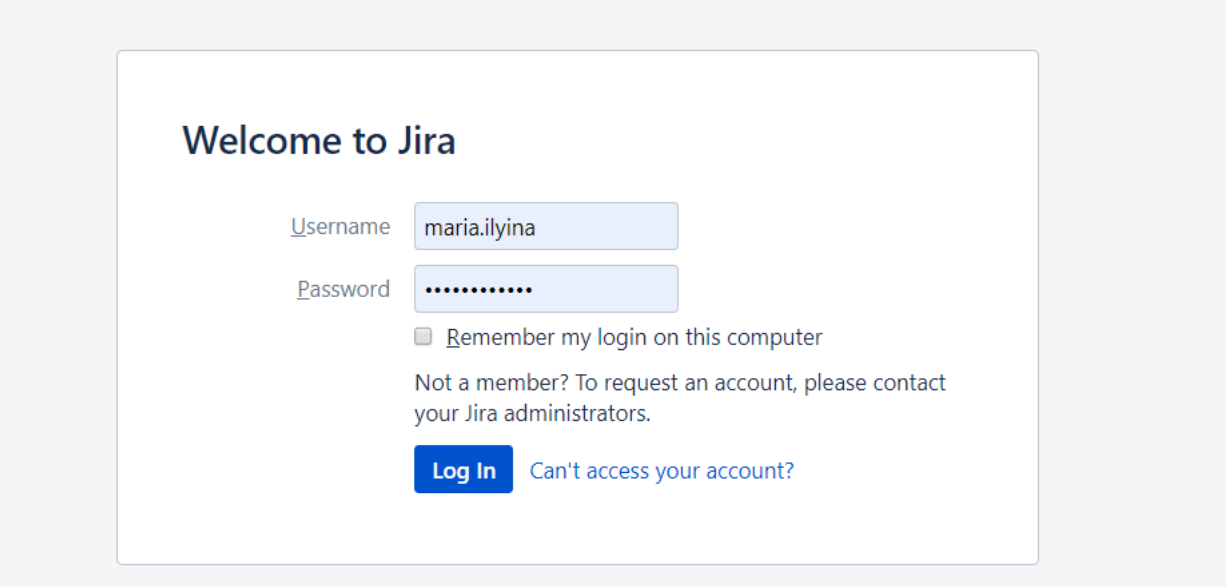

Рисунок 1.1 – Сторінка авторізації

1.2 Додавання нової задачі

Найменшою одиницею додатку є задача. Саме задачі є цеглинками з яки складається ідеально спланований день. Отож для планування дня необхідно створити принаймні одну задачу. Для її створення необхідно натиснути кнопку додавання задачі та заповнити усі необідні поля, такі як назва, опис. Також, якщо це необхідно, можна додати зв’язок із тікетом в Jira, або сторінкою у Cnfluence. Після чого потрібно натиснути кнопку «ОК».

(Рисунок 1.2).

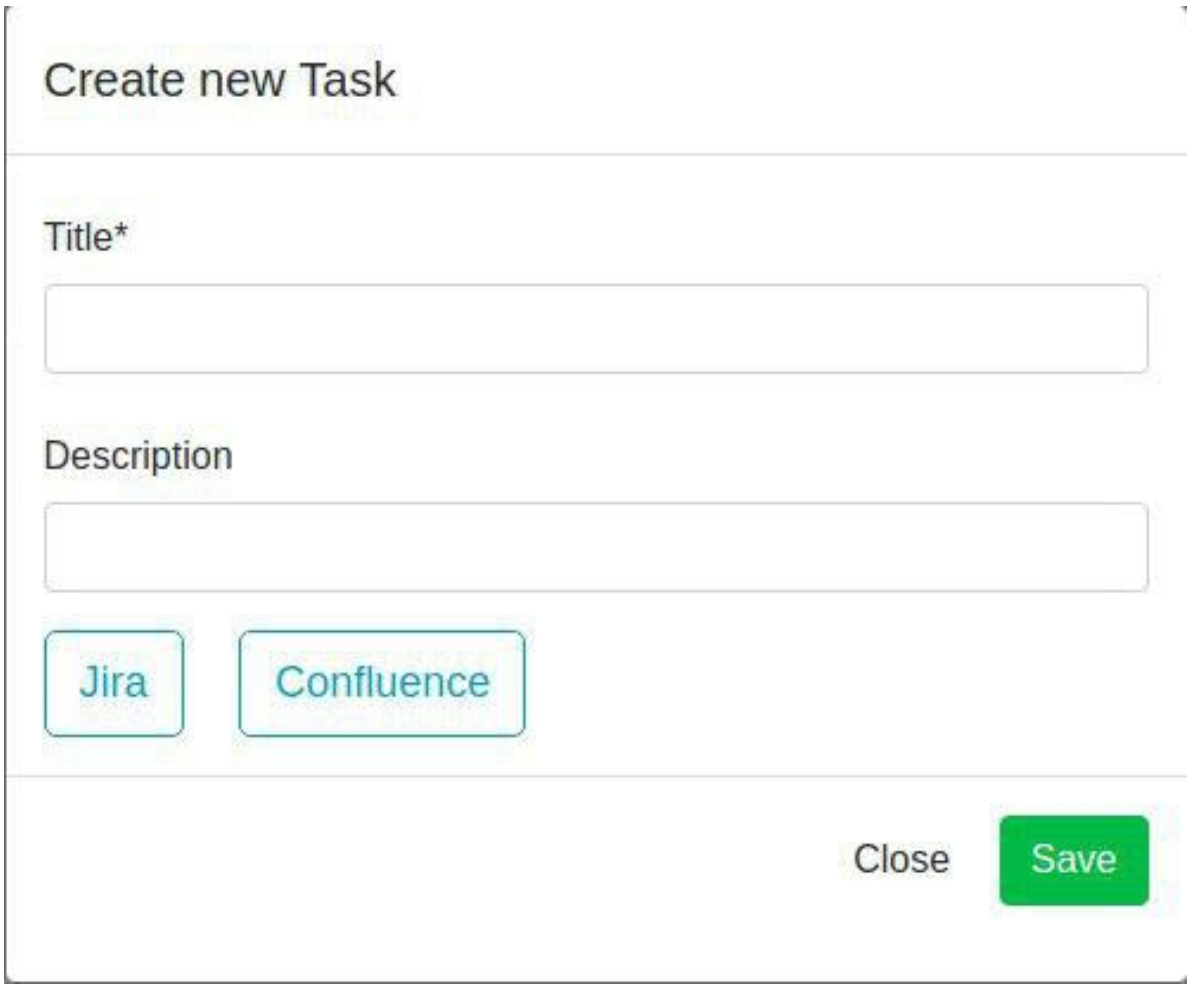

Рисунок 1.2 – Модальне вікно створення задачі

1.3 Редагування задачі

Будь-яке поле задачі можно змінити та відрегувати. Для цього потрібно натиснути на кнопку редагування. У відкритому модальному вікні змінити усі необхідні дані та натиснути кнопку збереження.

(Рисунок 1.3)

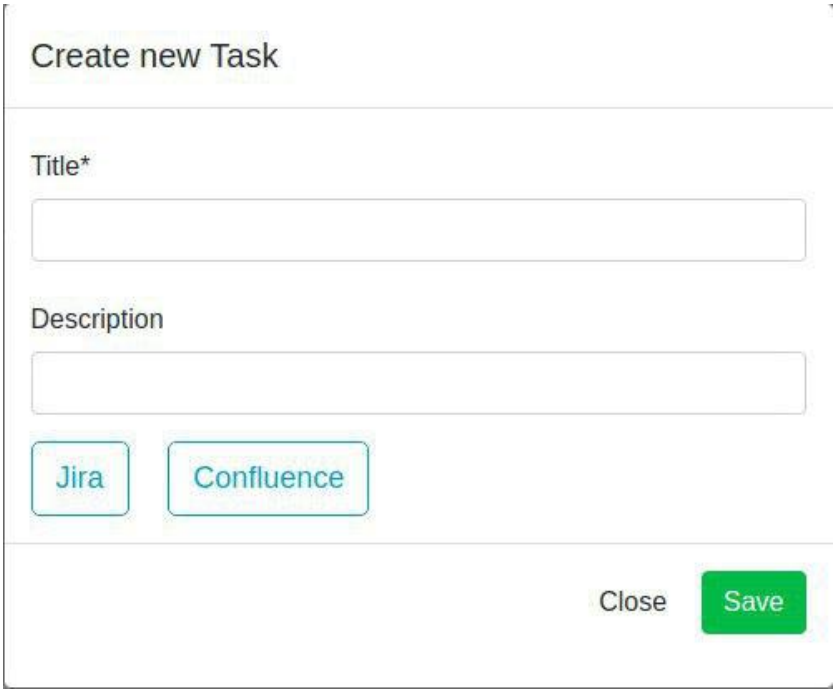

Рисунок 1.3 – Модальне вікно для редагування

Після редагування оновлена задача з’явиться на дошці (Рисунок 1.4).

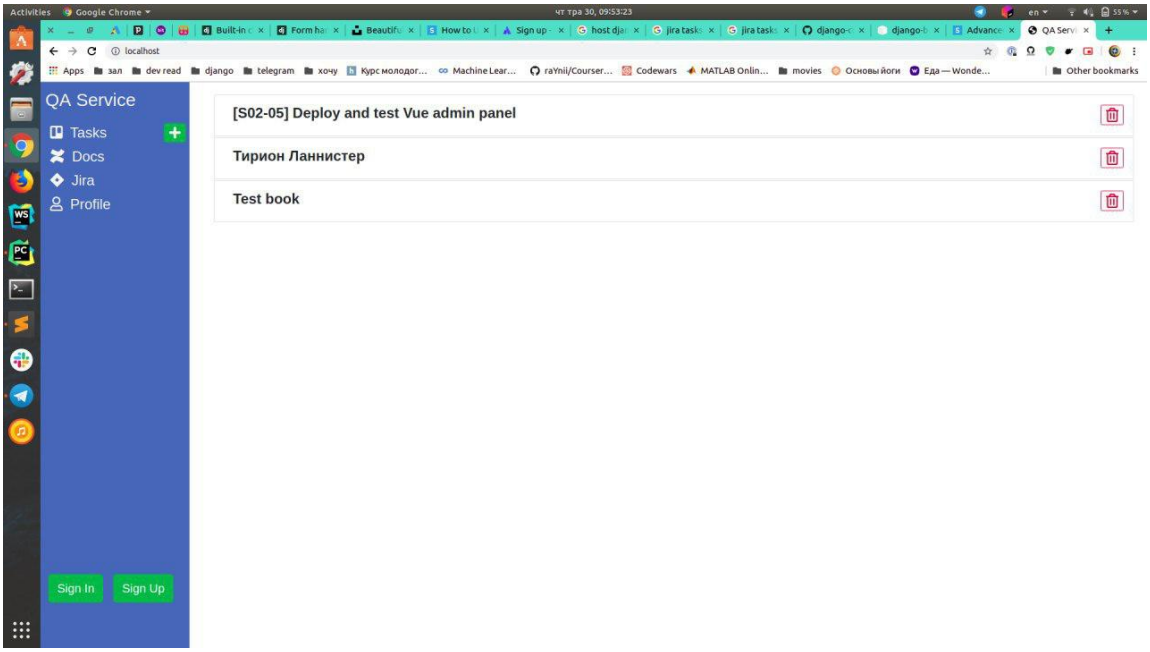

Рисунок 1.4 – Загальний вигляд дошки задач

1.4 Зміна пріоритету задачу

Додаток створенний для того, щоб планувати задачі, тобто зміна пріоритету задачі є дуже необхідним. Для того, щоб змінити пріоритет задачі потрібно виконати наступні дії: вибрати задачу; затиснути ліву кнопку миші; перетягнути задачу у потрібне місце; відпустит ліву кнопку миші.