ПРИКЛАДНІ ПРОБЛЕМИ КОМП'ЮТЕРНИХ НАУК,
БЕЗПЕКИ ТА МАТЕМАТИКИ, 2023, № 2

УДК 004.9

РОЗРОБКА ХМАРНОГО СЕРВІСУ ДЛЯ ОРГАНІЗАЦІЇ БУФЕРУ ОБМІНУ

Пелех Г.П.,
Булатецька Л.В. (ORCID: 0000-0002-7202-826X),
Волинський національний університет імені Лесі Українки, Луцьк, Україна

DEVELOPMENT OF A CLOUD SERVICE FOR ORGANIZING THE CLIPBOARD

Pelekh H.P., Bulatetska L.V.
Lesya Ukrainka Volyn National University, Lutsk, Ukraine

Abstract. In this paper, a range of applications, functionally similar to a cloud clipboard, was analyzed, including Crococlip, Copy 2 Online, and Online Clipboard. Based on this analysis, requirements for the software were identified, with the primary purpose of facilitating file and data exchange. A prototype of a cloud clipboard was implemented using interfaces that interact with the file system and the computer's clipboard from within a web browser. Programming aspects of the software components, data storage methods, and web server configuration were discussed. The final product underwent testing, with identified areas affecting user experience requiring refinement. The cloud clipboard was tested on a real hosting environment for student projects within the Department of Computer Science and Cybersecurity. The developed product enables data and file exchange between devices with internet access. The development API can be used independently. In comparison with counterparts, the product allows exchange without authentication or additional software installation, supports uploading files, and utilizes drag-and-drop functionality. The complexity of the code cell, and consequently the security level, is determined by the user.

Keywords: cloud service, web application, interface, clipboard, drag and drop, information technology, C#, ASP.NET, HTML, JavaScript, Angular, MongoDB, Nginx, Docker.

Вступ. Актуальність розробки хмарного буферу обміну надзвичайно важлива в контексті сучасних тенденцій в області обміну інформацією. Зі зростанням використання Інтернету та зменшенням ролі фізичних носіїв, зокрема флеш-накопичувачів, користувачі стикаються з потребою ефективного, безпечного та зручного обміну файлами та даними.

Завдяки поширенню роботи з великими обсягами інформації та важливості швидкості обміну, виникає необхідність в інноваційних рішеннях. Хмарний буфер обміну вирішує цю проблему, надаючи зручний інструмент для передачі файлів через інтернет, без надмірних процедур та обмежень.

Мета дослідження полягає у створенні робочого прототипу зручного у використанні хмарного буферу обміну.

Серед інших програмних продуктів такого типу найбільш схожими на хмарний буфер обміну є: Crococlip, Copy 2 Online та Online Clipboard.

Crococlip – вебзастосунок, розроблений за допомогою бібліотеки React. Цей інструмент дозволяє тимчасово зберігати та поширювати текст, надаючи можливість простого форматування та вибору терміну дії посилання. Crococlip дозволяє швидко обмінюватися даними між пристроями, використовуючи згенерований унікальний код, за яким можна отримати доступ до даних, ввівши цей код на веб-сайті Crococlip. Однак слід відзначити обмеження, пов'язані з можливістю працювати лише з текстовим вмістом [1].

Copy 2 Online – desktop-застосунок, розроблений Arketip Software and Consultancy Ltd. Доступність для встановлення у Windows, Linux та MacOS пропонує більшу інтегрованість у робочий процес користувача. Також можливе використання вебверсії застосунку. Історія копіювань дозволяє працювати із кількома текстами одночасно [2].

Окрім архітектури, основною відмінністю від попереднього застосунку є необхідність авторизації. Це дозволяє синхронізувати роботу застосунку на декількох пристроях та використовувати програму без постійного обміну кодами доступу, що безперечно є перевагами. Проте головний недолік полягає у обмеженості безкоштовної версії застосунку, що зберігає лише 10 останніх копіювань, дозволяє копіювання лише 100 символів за раз та одночасне використання одного аккаунту лише з двох пристроїв.

Online Clipboard – ще один вебдодаток, що дозволяє простий обмін даними через Інтернет. Він приймає на вхід текст, введений у текстове поле, або дані з буферу обміну. На відміну від першого застосунку, він не підтримує форматування тексту, проте пропонує функцію вставлення даних напряму з буферу обміну. Цей застосунок теж генерує код доступу до завантажених даних. На сторінці відсутня будь-яка інформація про термін зберігання даних. Код доступу складається із чотирьох цифр та генерується непослідовно, тому невідомо, що відбувається після генерації десяти тисяч кодів: перезапис старих даних, чи помилка програми. Вихідний код додатку був доступний для аналізу через інструменти розробника. З нього видно, що дані з буферу дістаються за допомогою Clipboard API (рис. 1). [3]

```
async function getClipboardContentsAndSend() {
    const items = await navigator.clipboard.read();
    console.log(items)
    const textBlob = await items[0].getType(items[0].types[0]);
```

Рис. 1. Фрагмент функції взаємодії з буфером обміну

Головною відмінністю цього застосунку від аналогів є можливість вставлення із буферу обміну зображень. При цьому використовується елемент розмітки з посиланням на графічний файл, тому використання цієї функції обмежено вебкартинками.

В табл. 1 подано порівняння розглянутих існуючих хмарних буферів обміну.

Таблиця 1. Порівняльний аналіз існуючих хмарних буферів обміну

	Робота з				Синхронізація (авторизація)
	текстом	картинками	буфером обміну	іншими файлами	
Crococliip	+	-	-	-	-
Copy 2 Online	+	-	+	-	+
Online Clipboard	+	+	+	-	-

Результати дослідження та їхнє обговорення.

Розробка призначена для тимчасового зберігання та обміну файлами та даними буферу обміну через мережу Інтернет.

Хмарний буфер повинен виконувати такі операції:

- зберігати файли та дані буферу обміну у визначених користувачем комірках;
- видаляти дані через певний проміжок часу;
- видаляти файли за запитом користувача;
- надавати користувачу доступ до комірки за її кодом.

Хмарний буфер обміну повинен бути доступний користувачам у вигляді вебзастосунку. Користувач повинен мати можливість:

- використовувати застосунок без будь-якої авторизації;
- самостійно визначати код комірки;

- вивантажувати та завантажувати файли та дані у застосунок через перетягування або копіювання.

Результатом розробки повинен бути застосунок у вигляді набору файлів, який можливо завантажити на сервер та запустити.

Програмна розробка складається із трьох основних елементів: фронтенд, бекенд та база даних. Для організації взаємодії між ними кожен компонент поміщено у окремий контейнер (рис. 2).

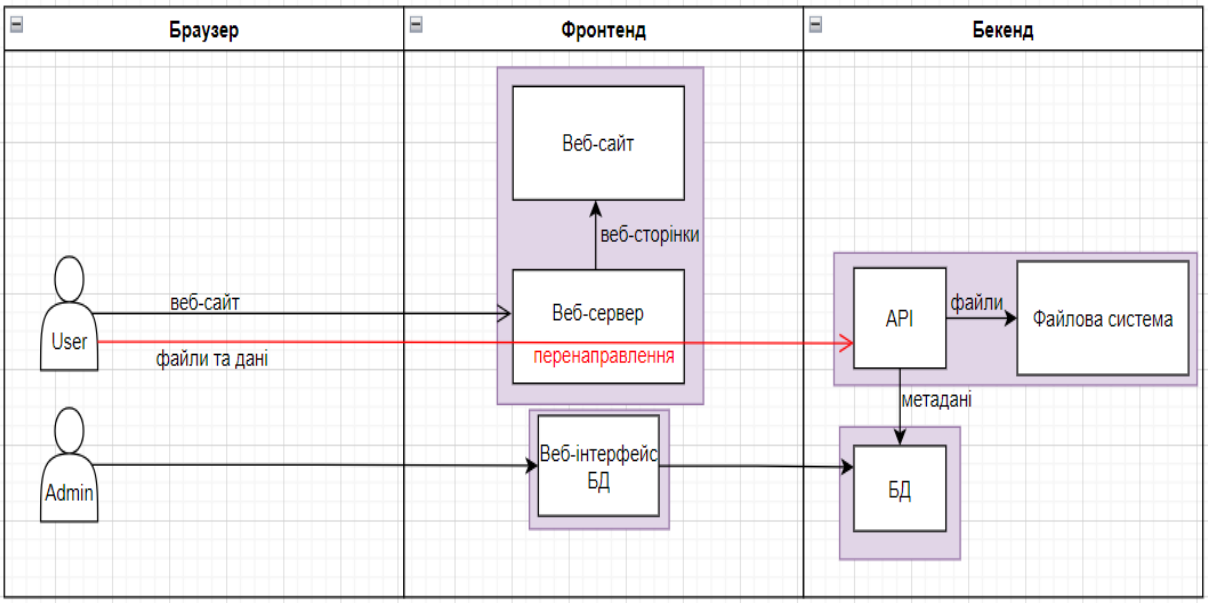

Рис. 2. Загальна структура взаємодії елементів у проекті

У першому контейнері знаходяться усі файли, необхідні для роботи фронтенду, та запущено вебсервер, що обробляє запити користувачів.

Інтерфейс фронтенд-частини дозволяє користувачу ввести код комірки та працювати з її вмістом: завантажувати, копіювати та видаляти дані (рис. 3). Для зручності можливе перетягування елементів між файловою системою комп'ютера та вікном комірки.

Вебсервер отримує запити, відправлені користувачами та відображає сторінку згідно умов маршрутизації. Запити для роботи з файлами та даними перенаправляються у другий контейнер.

Другий контейнер містить бекенд-частину. Вона обробляє запити, перенаправлені вебсервером та відповідає за:

- оновлення вмісту бази даних;
- роботу з файловою системою;
- аудит та видалення застарілих файлів.

Файли зберігаються у файловій системі під випадковими іменами. Файли видаляються через певний проміжок часу після спроби доступу до них або спеціального запиту на їх аудит.

У третьому контейнері знаходиться база даних. Використовується нереляційна база даних, яка складається із двох таблиць. У одній зберігається вміст використовуваних комірок, друга зберігає метадані усіх збережених файлів (рис. 4).

Четвертий контейнер не є обов'язковим та містить вебінтерфейс для бази даних. Це дозволяє наочний огляд її вмісту та полегшує віддалене адміністрування.

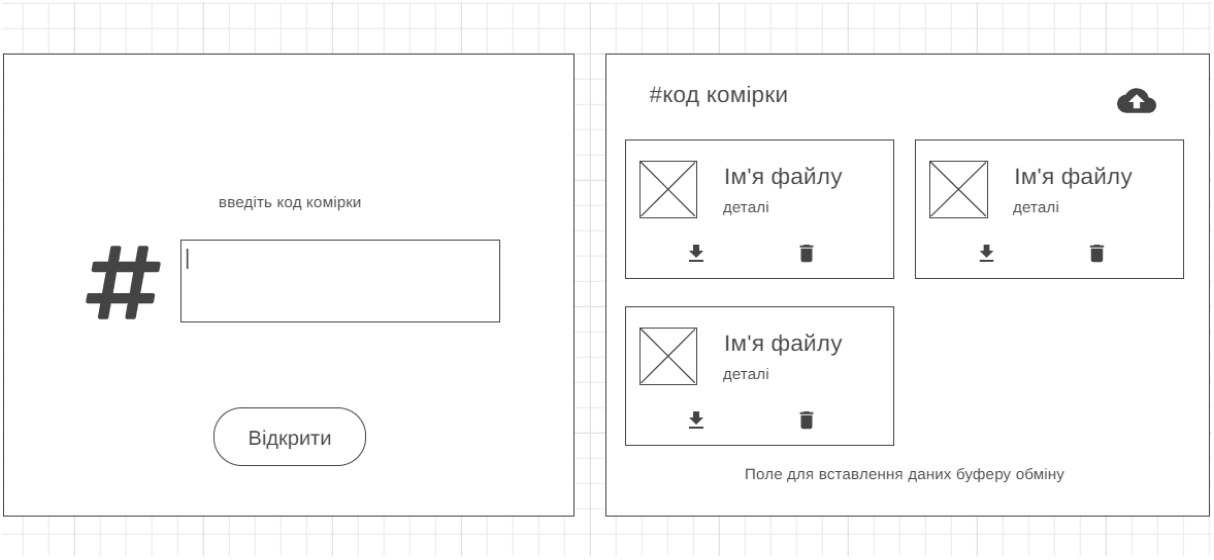

Рис. 3. Каркас інтерфейсу головної сторінки та подання комірки

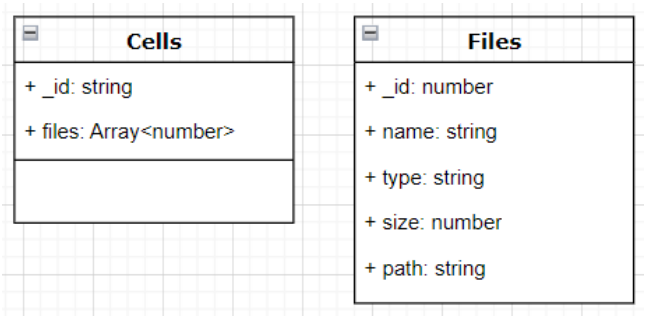

Рис. 4. Схема бази даних

Головними критеріями відбору засобів розробки були простота застосування та схожість робочих процесів. Зазвичай кожен фрагмент застосунку потребує окремого підходу до вибору інструментів, проте через індивідуальний характер проекту вибір фронтед та бекенд фреймворків із схожою архітектурою дозволяє значно спростити роботу з ними.

Для створення фронтенду застосунку було використано фреймворк Angular. Angular – вебфреймворк, що використовується для розробки динамічних односторінкових додатків (SPA). Він базується на мові програмування TypeScript і дозволяє розробникам створювати масштабовані програми з високою продуктивністю. Використовує паттерн розробки Model-View-Controller (MVC), надає розширені можливості з маршрутизації, формування запитів на сервер, валідації даних тощо. Використовує декларативний підхід до створення компонентів, що забезпечує повторне використання коду та легку керованість [4].

Альтернативами у цьому виборі були фреймворк Vue та бібліотека React. У порівнянні із ними Angular має більш визначену модель MVC, що спрощує масштабування та пропонує готовий каркас фрагментів додатку.

Для розробки бекенд частини було використано платформу ASP.NET та мову програмування C#. ASP.NET Web API (ASP.NET вебінтерфейс програмування додатків) – це фреймворк розробки вебсервісів, що дозволяє створювати та розгортати сучасні

застосунки з використанням архітектурного підходу REST (передача стану представлення). Ключовим принципом REST є безстанність – клієнт і сервер взаємодіють без збереження стану між запитами [5, 6].

Вибір нереляційної бази даних залежав від простоти інтеграції та взаємодії із нею зі сторони бекенду. Найлегшим та водночас найоптимальнішим інструментом виявилась база даних MongoDB. Пакет для підключення зі сторони ASP.NET дозволяв уникнути зайвих налаштувань та зосередитись на самій структурі бази даних. Зберігання даних у вигляді документів повністю покривало потреби застосунку. [7]

Для контейнеризації та налаштування взаємодії було використано Docker. Його публічна бібліотека готових образів дозволила підібрати потрібну версію бази даних, інструкції якої підтримувалися процесором сервера. Для налаштування одночасного запуску та роботи компонентів було використано Docker Compose. [8]

Хмарний буфер обміну дозволяє завантажувати на сервер файли та дані локального буферу обміну. Завантаження можливе декількома способами: через натискання відповідної кнопки, шляхом перетягування даних у комірку та через вставлення даних із буферу обміну. Зворотне завантаження можливе тими ж способами.

Для спрощення реалізації застосунок обробляє дані буферу обміну як об'єкт файлу, ім'ям якого є вміст буферу обміну, типом – типи даних у буфері обміну а розміром – кількість цих типів.

Існує три способи завантаження даних із сервера.

Перший спосіб – натискання відповідної кнопки. У компоненті file-item створено дві кнопки: для завантаження та видалення файлу.

Кнопка завантаження файлу використовує посилання, згенероване сервісом взаємодії з API. У випадку завантаження даних буферу обміну вони записується у текстовий файл.

Другий спосіб завантаження – перетягування потрібного компоненту у цільове вікно. Для цього додано панелі дій компоненту file-item властивість draggable.

Було створено функцію, що буде реагувати на подію перетягування елементу та записувати у неї дані, необхідні для його завантаження. У разі перетягування файлу записується посилання на нього, якщо ж перетягнуто елемент із даними буферу обміну – відповідні дані записуються у подію перетягування.

Для реалізації можливості перетягування між комірками застосунку записується ідентифікатор файлу. Щоб уникнути випадкового копіювання елементу у вихідну комірку, записується також ідентифікатор комірки походження (рис. 5).

Третій спосіб завантаження із застосунку специфічний для даних буферу обміну – копіювання даних у локальний буфер обміну. Для цього потрібно створити функцію, що реагуватиме на подію копіювання та змінюватиме дані, що копіюються у буфер обміну. Для вибору елементу, що буде джерелом даних, потрібно спочатку реалізувати батьківський компонент cell-view.

Компонент cell-view відображатиме файли, що містяться у відображуваній комірці, тобто компонент file-item для кожного файлу у комірці. Аргументами у компонент передаються ідентифікатор поточної комірки для перетягування між ними, дані самого файлу, реєструються запити на видалення файлу та здійснюється обмін даними для індикації даних буферу обміну, вибраних для копіювання (рис. 6).

Завантаження файлів на сервер також здійснюється трьома способами. У компоненті cell-view реалізована кнопка завантаження файлів у комірку.

При натисканні на кнопку файли, що задовольняють умови застосунку будуть завантажені через API.

```
dragDownload(event: DragEvent)
{
  if(!event.dataTransfer||!event.target) return
  if(this.file.type=='clipboardData')
  {
    for(let i=0;i<this.clipboardData.length;i++)
    {
      event.dataTransfer.setData(this.clipboardData[i].type,
      this.clipboardData[i].data);
    }
  }
  else{
    event.dataTransfer.setData('DownloadURL', `${this.file.type}
    :${this.file.name}:${this.url}`);
  }
  event.dataTransfer.setData('fileID', this.file.id);
  event.dataTransfer.setData('originCell', this.currentCell)
}
```

Рис. 5. Реакція на подію перетягування комірки

```
<div  class="list">
    <div *ngFor="let file of files;index as i;trackBy:trackByFunc"
    class="list-item">
        <div *ngIf="file|async as file" class="list-item-wrapper">
            <app-file-item [currentCell]="this._cellID" [file]="file"
            (delete)="delete($event)" [autoLoad]="_autoLoad"
            [clipboardSelected]="selectedClipboard==i"
            (clipboardSelected)="selectedClipboard=i"></app-file-item>
        </div>
    </div>
    <div *ngIf="files.length==0" class="empty">
        Cell is empty!
    </div>
</div>
```

Рис. 6. Відображення списку файлів у комірці

Для завантаження файлів через перетягування у комірку необхідно використати директиву. Директива – це механізм, що дозволяє розширювати функціональність елементів в додатку. Директиви дозволяють вставляти нові атрибути, стилі та поведінку до існуючих елементів або навіть створювати власні елементи з певними функціональними можливостями.

При натисканні на іконку елементу із даними буферу обміну, обраний елемент підсвічується та здійснюється виділення частини його вмісту у компоненті file-item (рис. 7). Після натискання сполучення клавіш для копіювання, подія копіювання реєструється лише у елементі із виділенням, що дозволяє записати дані у буфер обміну лише із нього (рис. 8).

```
makeSelection(e:Event)
{
  if(this.file.type!='clipboardData'||!e||!e.target) return
  let range = new Range();
  range.setStart(e.target as Node, 0);
  range.setEnd(e.target as Node, 0);
  console.log(range);
  document.getSelection()?.removeAllRanges();
  document.getSelection()?.addRange(range);
  this.clipboardSelectedEmitter.emit()
}
```

Рис. 7. Виділення вмісту елементу та його маркування для копіювання

```
onCopy(e:ClipboardEvent)
{
  e.preventDefault()
  e.stopPropagation()
  if(!e||!e.clipboardData) return;
  for(let i=0;i<this.clipboardData.length;i++)
  {
    e.clipboardData.setData(this.clipboardData[i].type, this.clipboardData[i].
    data);
  }
  for(let i=0;i<e.clipboardData.types.length;i++)
  {
    console.log(e.clipboardData.types[i])
    console.log(e.clipboardData.getData(e.clipboardData.types[i]))
  }
}
```

Рис. 8. Записування даних у буфер обміну при копіюванні

Створимо директиву, що реагує на перетягування даних на елемент із нею. Реалізуємо зміну зовнішнього вигляду елементу для візуального відображення можливості завантаження (рис. 9).

```
@HostListener('dragleave', ['$event'])
onDragLeave(event: DragEvent) {
  this.active = false;
  this.sizeError = false;
}

@HostListener('dragover', ['$event'])
onDragOver(event: DragEvent) {
  event.stopPropagation();
  event.preventDefault();
    this.active = true;
}
```

Рис.9. Зміна класів елементу, коли перетягувані дані входять та залишають область елемента

Якщо у перетягуваному елементі містяться дані про комірку походження, файл добавляється у цільову комірку без завантаження, використовуючи існуючі дані (рис. 10).

```
if(dataTransfer.getData('fileID'))
{
  let q = {fileID: dataTransfer.getData("fileID"),
  originCell:dataTransfer.getData("originCell")}
  this.fileMove.emit(q);
```

Рис.10. Перетягування файлу з існуючої комірки

Якщо перетягуваний елемент містить файли, їх дані записуються та випускається подія завантаження файлів (рис. 11).

```
} else if (dataTransfer.items) {
  const files = [];
  for (let i = 0; i < dataTransfer.items.length; i++) {
    if (dataTransfer.items[i].kind === 'file') {
      files.push(dataTransfer.items[i].getAsFile());
    }
  }
  dataTransfer.items.clear();
  let q = {target:{files:files}}
  this.fileDrop.emit(q);
```

Рис. 11. Виклик функції завантаження перетягуваних файлів

Останнім способом завантаження файлів на сервер є їх вставлення із буферу обміну. Для цього створено незмінне текстове поле, що дозволить користувачу вставляти вміст буферу обміну. Використаємо ту саму директиву, розширивши її можливості та додавши підтримку вставлення даних (рис. 12). Передбачимо можливість вставлення не лише текстових даних, але й файлів (рис. 13). У цьому випадку дані надсилаємо аналогічно до функції перетягування.

```
else if(event.clipboardData.items[i].kind === 'string'){
  let type = event.clipboardData.items[i].type;
  console.log(event.clipboardData.items[i])
  event.clipboardData.items[i].getAsString((s:string)=>{
    console.log(s)
    data.push({type:type??"null", data:s});
    if(i==0)
    {
      let w = {items:data}
      this.dataPaste.emit(w);
    }
  });
}
```

Рис. 12. Обробка даних, вставлених у поле

```
if (event.clipboardData.items[i].kind === 'file') {
  files.push(event.clipboardData.items[i].getAsFile());
}
```

Рис. 13. Обробка файлів, вставлених у поле

Отримані дані надсилаємо у функцію завантаження даних, реалізовану у компоненті cell-view (рис. 14).

У результаті створено робочу фронтенд частину хмарного буферу обміну. Для її запуску необхідно налаштувати вебсервер, у даному випадку – Nginx. Для поточного проекту достатньо мінімальних налаштувань, які можна описати у файлі nginx.conf.

На рис. 14 подано стартову сторінку вебзастосунку.

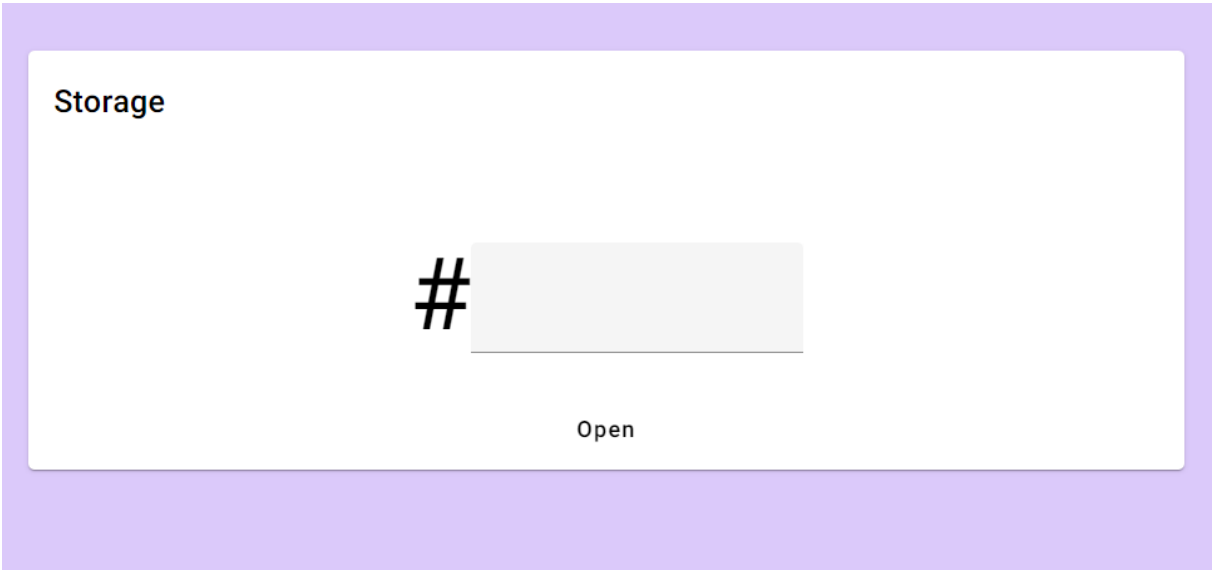

Рис.14. Стартова сторінка вебзастосунку

Після введення коду комірки та натиснення кнопки бачимо вміст обраної комірки (рис. 15). Після наповнення комірки даними комірка набуває наступного вигляду (рис. 16).

Тестування програми здійснювалося за допомогою Angular DevTools. Це розширення для Chrome DevTools, що дозволяє працювати напряму із компонентами.

У першій версії застосунку завантаження перетягуванням було можливо лише за попереднього завантаження файлу у вигляді BLOB (Binary Large Object). Проблемою було різне джерело файлу: замість відносних посилань використовувалися абсолютні, із доступом до бекенду через окремий порт, що заборонялось політиками безпеки. Після налаштування переадресації на вебсервері, усі зовнішні запити здійснювалися з використанням одного порту, що дозволяло завантаження перетягуванням напряму, без підготовки файлу заздалегідь.

Наявним недоліком поточної версії застосунку є відсутність оновлення вмісту комірки при вставленні елементу з буферу обміну. Можливим варіантом вирішення є примусове оновлення через фіксований період часу.

Також у застосунку відсутня можливість збільшити термін зберігання елементу. Для вирішення наразі можливо лише перенести файл у іншу комірку. Впровадження керованого терміну зберігання потребуватиме оновлення бази даних та усього механізму аудиту файлів.

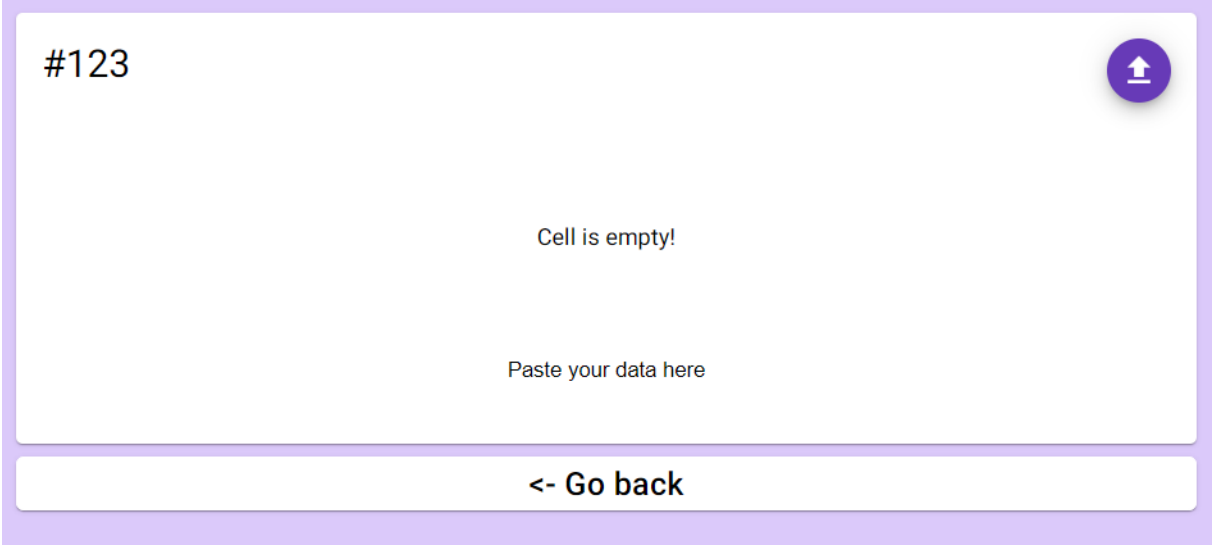

Рис. 15. Відображення вмісту комірки з ідентифікатором "123"

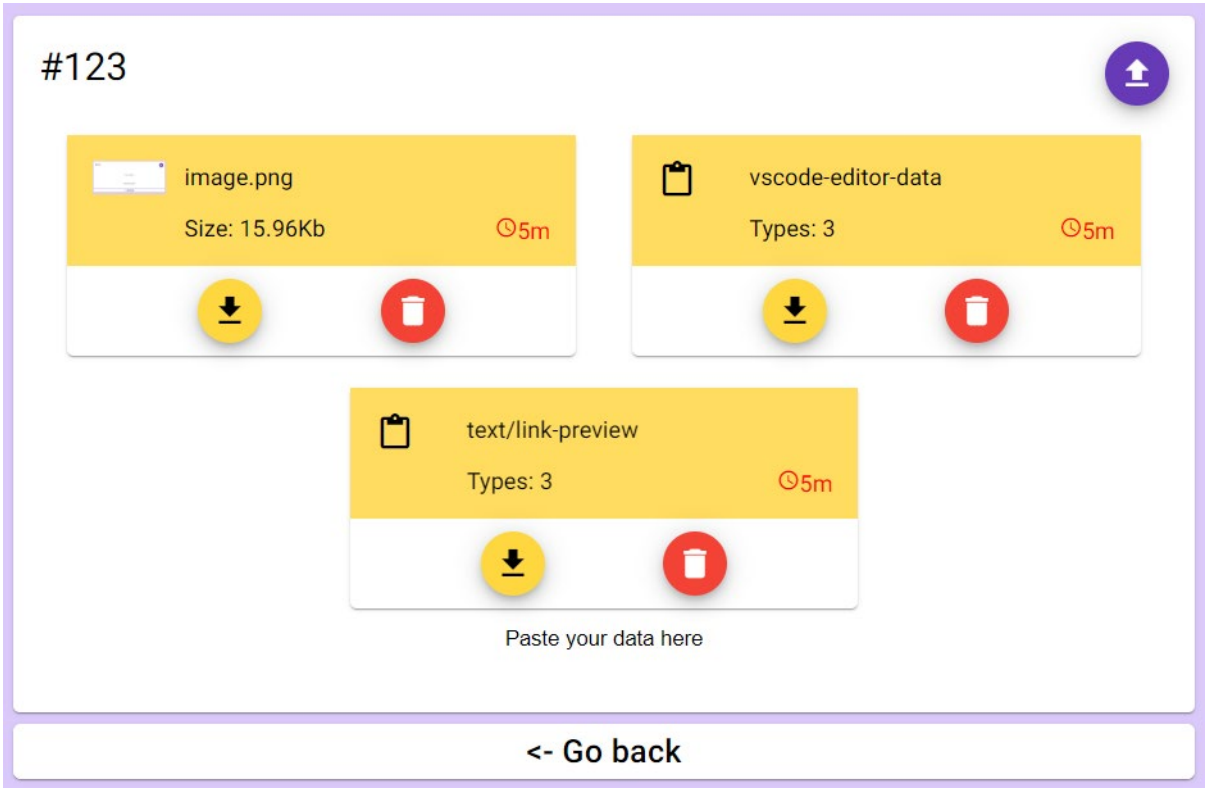

Рис. 16. Комірка із різними типами даних

Висновки. У результаті виконаної роботи реалізовано прототип хмарного буферу обміну із використанням інтерфейсів взаємодії файлової системи та буферу обміну комп'ютера із браузером. Головним призначенням продукту є обмін файлами та даними буферу обміну. У порівнянні із аналогами, продукт дозволяє обмін без авторизації та встановлення додаткового програмного забезпечення, надає можливість завантаження не лише тексту, а і файлів та дозволяє використовувати перетягування і копіювання. Складність коду комірки, а отже і рівень захищеності визначається користувачем.

Хмарний буфер обміну було розгорнуто та протестовано на реальному хостингу для студентських розробок кафедри комп'ютерних наук та кібербезпеки. Параметри хостингу: Ubuntu Server 22.04.2 LTS (GNU/Linux 5.15.0-73-generic x86_64), Docker 24.02.

Можливість використання вебзастосунку перевірена на персональному комп'ютері із ОС Windows 10 22H2 та браузером Microsoft Edge v114. Для коректної роботи необхідна підтримка браузером Drag and Drop API.

Бібліографія

1. crococlip | Online Clipboard. *crococlip | Online Clipboard*. URL: https://www.crococlip.com/.
2. Copy 2 Online. *Slashdot*. URL: https://slashdot.org/software/p/Copy-2-Online/.
3. Online Clipboard. *GGAMES.MOBI*. URL: https://online-clipboard.online/online-clipboard/.
4. Angular. *Angular*. URL: https://angular.io/docs.
5. Базові поняття .Net: Конспект лекцій. / укладачі: В. В. Булатецький, Л. В. Булатецька; ВНУ ім. Лесі Українки. Луцьк : ВНУ ім. Лесі Українки, 2022. – 37 с. URI : https://evnuir.vnu.edu.ua/handle/123456789/21666
6. What is ASP.NET? | .NET. *Microsoft*. URL: https://dotnet.microsoft.com/en-us/learn/aspnet/what-is-aspnet.
7. Team M. D. MongoDB Documentation. *MongoDB: The Developer Data Platform | MongoDB*. URL: https://www.mongodb.com/docs/ (date of access: 13.12.2023).
8. Docker Compose overview. *Docker Documentation*. URL: https://docs.docker.com/compose/.

References

1. crococlip | Online Clipboard. *crococlip | Online Clipboard*. URL: https://www.crococlip.com/.
2. Copy 2 Online. *Slashdot*. URL: https://slashdot.org/software/p/Copy-2-Online/.
3. Online Clipboard. *GGAMES.MOBI*. URL: https://online-clipboard.online/online-clipboard/.
4. Angular. *Angular*. URL: https://angular.io/docs.
5. Bazovi ponyattya .Net: Konspekt lektsiy. / ukladachi: V. V. Bulatets′kyy, L. V. Bulatets′ka; VNU im. Lesi Ukrayinky. Luts′k : VNU im. Lesi Ukrayinky, 2022. – 37 p. URI : https://evnuir.vnu.edu.ua/handle/123456789/21666
6. What is ASP.NET? | .NET. *Microsoft*. URL: https://dotnet.microsoft.com/en-us/learn/aspnet/what-is-aspnet.
7. Team M. D. MongoDB Documentation. *MongoDB: The Developer Data Platform | MongoDB*. URL: https://www.mongodb.com/docs/ (date of access: 13.12.2023).
8. Docker Compose overview. *Docker Documentation*. URL: https://docs.docker.com/compose/.