Универзитет „Св. Кирил и Методиј“ во Скопје
ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

ПРОЕКТНА ЗАДАЧА ПО ПРЕДМЕТОТ „НАПРЕДЕН ВЕБ ДИЗАЈН“

ТЕМА: **TicTacBoom - Website with SolidJS**

Изработила:
Стефанија Лазарева 223154

Јуни, 2025

СОДРЖИНА:

Вовед

Проектот „TicTacBoom" претставува интерактивна веб-апликација изградена со користење на современи front-end технологии, главно базирана на SolidJS – реактивен JavaScript framework познат по своите перформанси и едноставност. Целта на оваа апликација е да обезбеди интуитивен начин за управување со содржини преку картички и категории, кои корисникот може да ги филтрира, пребарува и уредува. Преку модерен и одзивен дизајн, проектот нуди лесна навигација и ефикасна интеракција.

Во основа, TicTacBoom претставува мала но моќна продуктивна алатка, инспирирана од потребите на корисниците за подобра организација и менаџирање со задачи. Преку систем на категории, модали за додавање и уредување содржини и едноставен механизам за пребарување, корисниците можат брзо и лесно да управуваат со сопствените податоци.

Со оглед на експоненцијалниот раст на веб-технологиите и зголемената побарувачка за брзи и лесни апликации, TicTacBoom е одговор на потребата од едноставна, но технички стабилна и проширлива апликација. Покрај тоа што служи како практична алатка, оваа апликација е и едукативен ресурс – демонстрирајќи ги принципите на реактивно програмирање, состојбен менаџмент, модуларен дизајн и современа UX пракса.

Апликацијата е замислена како основа што може лесно да се проширува, надградува и прилагодува, со потенцијал да се поврзе со реални бекенд услуги, бази на податоци, па дури и мобилни платформи преку хибридни технологии. Во суштина, TicTacBoom е проект кој го комбинира практичното со теоретското, и претставува мост помеѓу учењето и реалната имплементација.

Цели и мотивација

Примарната цел на TicTacBoom е да се изгради едноставна, но функционална платформа за управување со содржини (картички) при што се следат најдобрите практики во развојот на front-end апликации. Апликацијата е дизајнирана за да ги демонстрира способностите на SolidJS, но и да послужи како основа за понатамошен развој или проширување, вклучително и интеграција со back-end системи, бази на податоци или REST API интерфејси.

Проектот исто така има за цел да прикаже како може да се имплементира реактивен пристап во развој на кориснички интерфејси, без преголема сложеност. Преку TicTacBoom се илустрира како може да се креира апликација со одлични перформанси, скалабилност и одржливост на кодот, користејќи минимален број на библиотеки и алатки. Дополнителна цел беше да се создаде основа за апликација која може да се користи и на поголеми проекти преку модуларна архитектура.

Мотивацијата произлегува од потребата за едноставни алатки кои го олеснуваат организирањето на информации и задачи – било да е тоа за лична употреба, едукација или тимска продуктивност. Во процесот на учење на нови технологии, особено frameworks како SolidJS, ваков тип на проект овозможува практично искуство со управување на состојба, динамичко прикажување на податоци, имплементација на форми и интерактивни компоненти. TicTacBoom е замислен и како демонстративен проект кој може да биде презентиран како портфолио пример за идни можности во индустријата. од потребата за едноставни алатки кои го олеснуваат организирањето информации и задачи – било да е тоа за лична употреба, едукација или тимска продуктивност.

Технологии и алатки

Во развојот на апликацијата TicTacBoom беа употребени неколку внимателно избрани технологии и алатки со цел да се постигнат високи перформанси, интуитивен интерфејс и чист код. Секоја алатка одигра значајна улога во различни фази од развојниот процес. Оваа комбинација на технологии овозможи брз, стабилен и модуларен развој на апликацијата.

SolidJS

SolidJS е front-end JavaScript framework кој користи реактивен пристап за манипулација со состојба и DOM. Главната предност на SolidJS е тоа што користи compile-time reactivity, што резултира со многу брз и ефикасен извршен код. За разлика од традиционалните frameworks кои користат виртуелен DOM, SolidJS директно манипулира со реалниот DOM, што го прави исклучително брз. Оваа карактеристика беше клучна за развој на лесна и интерактивна апликација.

Vite

Vite беше користен како build tool. Тоа е модерна алатка која овозможува исклучително брзо стартување на девелопмент сервер и hot module replacement (HMR), што значително го забрза развојниот процес. Благодарение на Vite, промените во кодот веднаш беа видливи во прелистувачот, што ја зголеми продуктивноста.

CSS и Tailwind

Во стилизирањето на апликацијата беше користен CSS со фокус на одзивен дизајн.

WebStorm

Како главна развојна околина (IDE) беше користен WebStorm, поради неговата поддршка за JavaScript, SolidJS екстензии и моќни алатки за дебагирање.

Node.js и npm

Node.js беше користен за извршување на девелопмент серверот, додека npm (Node Package Manager) служеше за инсталација и менаџирање на зависности. Преку npm беа додадени сите потребни библиотеки и алатки за развој.

Архитектура на апликацијата

Апликацијата TicTacBoom е едноставна, но моќна веб-апликација за менаџирање на задачи, дизајнирана да им овозможи на корисниците лесен начин за создавање, уредување, филтрирање и следење на нивните обврски.

TicTacBoom следи Single Page Application (SPA) архитектура, каде целата интеракција се изведува на клиентската страна. Овој пристап обезбедува флуидно и брзо корисничко искуство, бидејќи сите промени се прикажуваат без повторно вчитување на страната. Апликацијата не користи сервер или база на податоци; наместо тоа, податоците се складираат локално во прелистувачот преку LocalStorage API.

Апликацијата е изградена со компонентно-базирана архитектура, каде секоја логичка целина (форма за задачи, филтрирање, модално прозорче и сл.) е посебна компонента. Овој пристап ја олеснува повторната употреба, тестирањето и одржувањето на кодот.

Главни компоненти

TicTacBoom е модуларно изградена апликација со неколку клучни компоненти кои меѓусебно соработуваат преку реактивни сигнали и прописи. Главните компоненти се:

- App - главната компонента каде се иницијализираат податоците и се вметнуваат сите подкомпоненти. Служи како корен на апликацијата.
- List - одговорна за прикажување на сите картички, односно задачи. Се користи For loop од SolidJS за реактивно рендерирање на листата.
- Card - претставува поединечна картичка, со можности за уредување, бришење и менување статус.
- CardModal - модално прозорче за додавање нова картичка, со валидација и употреба на createSignal.

- SearchBar - компонента која овозможува пребарување низ картичките користејќи createMemo.
- CategoryFilter - компонента за филтрирање на картичките по категории или статус.

App.jsx

Компонентата App.jsx претставува централна логичка точка на апликацијата. Во неа се дефинира и управува глобалната состојба, се оркестрираат сите визуелни и интерактивни компоненти, и се реализираат клучните функционалности како додавање, уредување, бришење и пребарување задачи.

Во рамките на App.jsx се користат бројни createSignal состојби, со цел да се управува со динамиката на апликацијата и да се реагира на промени во состојбата. Се овозможува генерирање на категории, поставување потсетници, а главните состојби вклучуваат:

- lists – главната колекција од табли со задачи
- nextCardId, nextListId – се користат за автоматско генерирање на ID вредности
- addingCardToListId, editingCard, currentModalListId – го контролираат приказот на модалите за додавање или уредување задачи
- searchQuery, selectedCategory, categories – овозможуваат пребарување и филтрирање на содржините.
- notifications – управува со прикажувањето на кратки известувања за задачи.

Компонентата дефинира повеќе функции кои се поврзани со кориснички интеракции:

- **handleSearch (query)** – пребарување според текст.
- **handleCategorySelect (category)** – селекција или поништување на категорија.
- **handleAddCategory (newCategory)** – додавање на нова категорија во листата.
- **handleSaveCard (...)** – зачувување нова или уредена задача.
- **handleDeleteCard (...)** – бришење на постоечка картичка.
- **handleMoveCard (...)** – преместување картичка меѓу табли.

Преку функцијата filteredLists() се применуваат филтри базирани на текстуален пребарувачки збор (searchQuery), селектирана категорија (selectedCategory). Филтрирањето се врши по content, description, labels и category на картичките, со цел да се прикажат само релевантни резултати.

Компонентата App ги вклучува следните визуелни елементи:

- **Header** – горен дел со наслов.
- **SearchBar** – поле за пребарување.
- **CategoryFilter** – визуелен избор на категории.
- **List** – прикажува табла со картички.
- **CardModal и AddCategoryModal** – модални форми за додавање и уредување.
- **Notification** – кратки известувања за задачи.

Овие компоненти се рендерираат реактивно, со користење на SolidJS директивите For и Show, што овозможува брз и ефикасен DOM рендеринг.

Header.jsx

Компонентата Header е едноставна, минималистичка, но функционална визуелна компонента, која го претставува насловот на апликацијата и поставува тон за останатиот дизајн. Нејзината независна имплементација придонесува за чистота на кодот и лесна повторна употреба низ различни делови од апликацијата.

Card.jsx

Компонентата Card.jsx претставува визуелна и интерактивна единица за прикажување на една единствена задача (картичка) во апликацијата. Таа е дизајнирана за да биде самостојна, повторно употреблива и целосно реактивна, со вградени функционалности за приказ, уредување, бришење, споделување и нотификации.

Основни карактеристики се: прикажување на задача со категории, слики, етикети, рокови и потсетници, интерактивност, вградена логика за стилско означување на рокови и потсетници, споделување, обработка на грешки при вчитување на слики преку default placeholder.

Компонентата прима неколку props:

- card: објект со податоци за задачата
- listId: ID на таблата каде се наоѓа задачата
- onEdit(card, listId): callback при клик за уредување
- onDelete(cardId, listId): callback за брзо бришење.

Функционалности и логика:

- Споделување задача (handleShare): Ако уредот поддржува Web Share API → отвара системски дијалог. Ако не → покажува внатрешна нотификација со showBrowserNotification.
- Брзо бришење (handleQuickDelete): Повик до props.onDelete(...) без потреба од влегување во уредување.
- Приказ на рок (Due Date): Се користи getDueDateStatus за да се определи дали рокот е поминат (overdue), денес (today), утре (tomorrow), или подоцна (без класа). Се додава соодветна класа (classList) за визуелно означување со боја.
- Потсетник: Се проверува дали reminder е поминат со isReminderDue. Се прикажува икона и форматот на времето со formatReminderTime.
- Drag and Drop: При старт на влечење (onDragStart), се поставуваат dataTransfer вредности: cardId, sourceListId. Оваа логика се користи подоцна во onDrop во таблите (List компонента).
- Грешка со слика: Ако не се вчита profileIcon или card.image, се прикажува placeholder слика.

За визуелен изглед, компонентата содржи CSS класи стилизирани така што секоја картичка визуелно да се издвојува и овозможува интуитивна интеракција.

CardModal.jsx

Компонентата CardModal е модален прозорец за креирање, уредување и бришење задачи (картички) во апликацијата. Претставува централен елемент за внес на податоци и поддржува целосна интеракција со сите атрибути на задачата: наслов, опис, категорија, етикети, слики, рокови и потсетници.

Основна улога е да креира и уредува задачи, го прифаќа објектот card како влезен пропс, кој може да биде празен (нова задача) или со податоци (постоечка задача).

Компонентата користи createEffect() за да ги пополни формите со вредностите на постоечката задача (props.card), кога компонентата се вчитува или се отвора за уредување. Ова обезбедува автоматско вчитување на податоците.

Функционалности:

- Зачувување на задача (handleSave) - Креира финален објект со сите податоци. Ја повикува функцијата props.onSave(updatedCard, listId, isNew). Автоматски затвора модалот.

- Бришење задача (handleDeleteClick()) - Проверува дали постојат card.id и listId. Повикува props.onDelete(...) ако е достапна. Се користи само кога се уредува постоечка задача.
- Откажување (handleClear) - го затвора модалот без промени преку props.onClose().
- Клучни команди (handleKeyDown) - Enter: Зачувува задача (освен во textarea). Escape: Го затвора модалот.
- Автоматски фокус (createEffect(() => {
  if (contentInputRef) contentInputRef.focus();
  }); - При отворање на модалот, фокусот автоматски се поставува на полето за наслов.

Компонентата овозможува избор на постоечка категорија или додавање нова. Кога корисникот избира „Add new category...“, се прикажува дополнително поле за внес на нова категорија.

Визуелно, прозорецот е поделен на заглавие (наслов и иконка за профил), главна форма (наслов, опис, категорија, етикети, слики, потсетник, рок) и копчиња (додавање, уредување, бришење) за постоечка задача.

AddCategoryModal.jsx

Компонентата AddCategoryModal е едноставен, но важен модален интерфејс кој им овозможува на корисниците да внесат и додаваат нови категории во апликацијата. Таа игра поддржувачка улога во организирањето на задачите според нивната категорија, обезбедувајќи едноставен и интуитивен начин за проширување на достапните филтри.

Основна улога е да обезбеди UI форма за внес на ново име на категорија, по потврдување треба да ја проследи новата вредност назад до родителската компонента преку props.onAdd и да обезбеди можност за откажување и затворање на модалот преку props.onClose.

Компонентата користи една реактивна состојба која го чува внесеното име за новата категорија.

Функционалности на оваа компонента се додавање на категорија (внес преку onInput={handleInputChange} каде секоја промена за текст се зачувува во полето newCategoryName), потврдување (handleAddClick - го отстранува празниот простор од името, ако името не е празно повикува props.onAdd(trimmedName) и потоа се

затвора модалниот прозорец), откажување (handleCancelClick - се повикува props.onClose() без да се додаде категорија). Оваа компонента исто така нуди поддршка за тастатура handleKeyDown (enter - додавање на категорија и escape - затворање на модалниот прозорец).

List.jsx

Компонентата List претставува еден визуелен и логички колона-елемент, кој содржи задачи (картички) поврзани со конкретна категорија или цел. Улога на оваа компонента е да прикажува група на задачи (картички) во рамките на една табла, да обезбедува UI копче за додавање нова задача, да поддржува drag-and-drop логика за преместување задачи меѓу различни табли и да ја користи компонента Card.jsx за визуелно прикажување на секоја задача.

Компонентата користи <For each={cardsToDisplay()}> за реактивно прикажување на сите задачи од конкретната табла. Секоја задача се прикажува со компонента Card, на која ѝ се проследуваат соодветните функции и ID вредности.

Компонентата имплементира три главни обработувачи:

- handleDragOver - Дозволува drop со e.preventDefault() и визуелен ефект.
- handleDragLeave - Го отстранува стилот за „drag-over“.
- handleDrop - Ги чита cardId и sourceListId од dataTransfer и иницира преместување.

Визуелна структура

```
<div class="list">
   <div class="list-header">{list.title}</div>
   <button class="add-task-btn">+ Add Task</button>
   <div class="list-cards">...</div>
</div>
```

list-header - Прикажува наслов на таблата.
add-task-btn - Копче за додавање нова задача.
list-cards - Содржи низа од Card компоненти и ги прифаќа drop-настаните.

CategoryFilter.jsx

Компонентата CategoryFilter е интерактивна лента за филтрирање, која овозможува избор на категории за прикажување задачи. Таа им овозможува на корисниците да го ограничат прикажувањето на задачи според избрана категорија, како и да го ресетираат филтерот со едно копче.

Влезни props: categories (достапни категории што треба да се прикажат.), selectedCategory (моментално селектираната категорија.), onSelect(category) (callback функција која се повикува при избор или поништување на категорија).

Функционалности:

- Селекција на категорија - Го повикува props.onSelect(category) со избраната вредност. Користи classList за да го означи избраното копче со класа selected.

```
onClick={() => handleCategoryClick(category)}
```

- Откажување на филтер - Прикажува дополнително копче „Clear filter“ само ако е активна некоја категорија. Повикот на handleClearFilter() ја ресетира категоријата (onSelect('')).

```
<Show when={props.selectedCategory}>
  <button ... onClick={handleClearFilter}>Clear filter</button>
</Show>
```

- For - за динамично рендерирање на копчиња за секоја категорија.

```
<For each={props.categories}>
  ...
</For>
```

- Show - условно го прикажува копчето за чистење на филтер.

```
<Show when={props.selectedCategory}>
  ...
</Show>
```

SearchBar.jsx

Компонентата SearchBar претставува едноставен, но функционален input-елемент за пребарување на задачи. Таа им овозможува на корисниците динамично да ги филтрираат задачите според внесен текст, како дел од напредната функционалност за навигација низ содржини.

Влезни параметри: searchQuery (почетна вредност за пребарување, која доаѓа од родителот), onSearch(query) (callback функција што се повикува секогаш кога се менува текстот).

Компонентата содржи една состојба:

```
const [inputValue, setInputValue] = createSignal(props.searchQuery || '');
```

Се користи createEffect() за автоматско ажурирање на влезната вредност ако се смени props.searchQuery (на пример, кога се ресетира пребарувањето од надворешен извор).

Промената на вредност на влезот се прави преку:

```
const handleInputChange = (e) => {
   const query = e.target.value;
   setInputValue(query);
   props.onSearch(query);
};
```

Секоја промена веднаш го ажурира локалниот state. Паралелно се повикува onSearch функцијата, која го филтрира списокот на задачи во родителската компонента (App.jsx).

Notification.jsx

Компонентата Notification претставува едноставен визуелен елемент за приказ на кратки известувања (нотификации) во апликацијата TicTacBoom. Таа се користи за информирање на корисникот за настани како потсетници за задачи или известувања за споделување, и се прикажува динамично на екранот, без да го прекине текот на користење.

Влезни параметри: notification (објект со две полиња title и content, кои се прикажуваат во компонентата), index (целобројна вредност која ја одредува вертикалната позиција на нотификацијата на екранот).

Index.jsx

Улогата на компонентата index.jsx е да го стартува цалиот циклус односно да направи почетна иницијализација, да ја поврзи апликацијата со реалниот DOM, да го интегрира HTML темплејтот и да овозможи стабилна контрола. Иако е едноставен кодот, тој е клучен за правилно функционирање на целиот проект.

Функционалности

Апликацијата TicTacBoom претставува интерактивен систем за менаџирање на задачи преку картички, организирани по листи и категории. Апликацијата е богата со функционалности кои го олеснуваат организирањето, пребарувањето и следењето на активности. Клучните функции кои ги поддржува системот се:

- Додавање на нова задача
    - Корисникот може да креира нова задача со клик на копче „+ Add Task“ во рамките на секоја табла. При клик, се прикажува модален прозорец (CardModal) каде може да се внесе наслов, опис, категорија (постоечка или нова), labels, слика и профил икона (URL), потсетник (reminder), рок (due date).
- Уредување на постоечка задача
    - При клик на постоечка картичка, се отвара CardModal компонента пополнета со тековните податоци. Корисникот може да направи измени и да кликне „Save“, по што промените се запишуваат во листата.
- Бришење на задача
    - Картичката може да се избрише на два начини: со клик на ✕ копчето (quick delete) од самата картичка, или со клик на „Delete“ копчето во модалот за уредување.
- Категоризирање
    - Секоја задача може да има своја категорија. Може да се избере веќе постоечка категорија, може да се додаде нова категорија преку AddCategoryModal. Исто така може да се филтрира по категорија преку компонентата CategoryFilter.
- Пребарување на задачи
    - Преку компонентата SearchBar, корисникот може во реално време да пребарува задачи според наслов, опис, лабели или категории.
- Преместување задачи
    - Секоја картичка може да се премести од една табла во друга преку drag-and-drop: Card.jsx ја иницира акцијата со onDragStart. List.jsx го обработува onDrop и го преместува картичката во целната табла. Логиката за ажурирање се изведува преку handleMoveCard() во App.jsx.
- Потсетници и рокови
    - Секоја задача може да има reminder и due date. Овие информации се обработуваат преку utility функции од utils.js.
- Notifications

    - При активирање на потсетник, апликацијата автоматски креира визуелна нотификација преку компонентата Notification.jsx. Секоја нотификација се прикажува на различна висина (top) според индекс. Нотификациите автоматски исчезнуваат по неколку секунди.
- Споделување задача
    - Преку Card.jsx, секоја задача може да се сподели преку Web Share API. Доколку API-то не е поддржано, се прикажува нотификација со информации за задачата.
- Зачувување на податоци
    - Сите податоци се чуваат локално во прелистувачот (LocalStorage), овозможувајќи перзистентност дури и по рефреш на страната. Ова е имплементирано со createEffect во App.jsx.

Помошни функции

Фајловите utils.js и formatters.js содржат сет на помошни (utility) функции кои се користат во апликацијата за пресметка и форматирање на рокови и потсетници. Овие функции се повикуваат од повеќе компоненти и овозможуваат информативен приказ на временски информации.

- formatReminderTime(isoString) - Форматира ISO датум за потсетник (reminder) во лесно разбирлива форма.
- formatDueDate(isoString) - Форматира ISO датум за краен рок (dueDate) во опишана фраза.
- getDueDateStatus(isoString) - Враќа статус за рокот во кратка форма, кој се користи за стилизирање и визуелно означување на картичките.
- isReminderDue(isoString) - Проверува дали времето е во минатото.
- setupReminders(lists, onReminderDueCallback) - Автоматски закажува потсетници за сите задачи што имаат reminder датум, користејќи setTimeout.
- showBrowserNotification(card) - Прикажува системска (browser) нотификација ако е дозволено преку Notification API.

Дизајн

Апликацијата TicTacBoom е дизајнирана со фокус на чист, модерен и одзивен интерфејс кој обезбедува пријатно корисничко искуство. Дизајнот не само што го

следи визуелниот идентитет на една продуктивна алатка, туку и активно го поддржува функционалниот аспект – правејќи го управувањето со задачи интуитивно и ефикасно.

При дизајнирањето на интерфејсот се водевме според следните принципи:

- Јасност и едноставност – секој елемент има своја функција и е лесно препознатлив.
- Фокус на содржината – задачите (картичките) се во центарот на вниманието.
- Конзистентност – преку повторна употреба на компоненти и стилови.
- Интерактивност – визуелна повратна информација при сите кориснички дејства.

Дизајнот е имплементиран преку Tailwind CSS и преку сопствени CSS класи (style.css, App.module.css, index.css).

Податоци и состојби

Апликацијата е базирана на реактивен модел на податоци, каде управувањето со состојбата (state) е клучен дел од логиката. Со користење на функционалностите од SolidJS, се постигнува флуидно, оптимизирано и лесно одржливо следење на промените во податоците и нивно реактивно прикажување на интерфејсот.

Главниот податочен модел во апликацијата се состои од:

- Lists - Колекции од задачи, секоја со уникатен id и title.
- Cards - Елементи што претставуваат поединечни задачи, со уникатен id, content (наслов), description, category, labels, image, profileIcon, reminder, dueDate.

Овие податоци иницијално се вчитуваат од датотеката initialData.js и се чуваат во состојба преку createSignal.

Во апликацијата се користи createSignal како основен механизам за следење и ажурирање на состојбата. Покрај createSignal, се користат и createEffect и createMemo. CreateEffect овозможува извршување на логика секојпат кога одредена состојба се менува. CreateMemo се користи за филтрирање и пребарување во листите.

Апликацијата не користи сервер или база на податоци. Наместо тоа, податоците се зачувуваат во LocalStorage, што овозможува перзистентност дури и по затворање на прелистувачот.

Исто така, секоја поголема компонента има свои локални состојби. А важно е да се напомене дека таквиот пристап обезбедува добра енкапсулација и лесна одржливост на кодот.

Придобивки од изборот на SolidJS за апликацијата

Во развојот на апликацијата TicTacBoom, беше избран framework-от SolidJS како главна алатка за изградба на корисничкиот интерфејс. Овој избор се базираше на високите перформанси, реактивниот систем, и едноставност во дизајнот.

Изборот на SolidJS за изградба на TicTacBoom се покажа како одлична технолошка одлука. Таа понуди перформанси, едноставност, реактивност и скалабилност, што овозможи реализација на една современа, лесна и функционално богата апликација. SolidJS е одличен избор за модерни front-end апликации со висок степен на интерактивност и динамички UI.

Предизвици и решенија

Некои од главните предизвици беа:

- Управување со повеќе модали - решено со централизирана состојба.
- Скалабилност на категориите и картичките - решено со генерички податоци и функции.
- Дизајн за мобилни уреди - решено со media queries и флексибилен layout
- Валидација при уредување - имплементирани проверки во модалите.

Заклучок

Апликацијата TicTacBoom е успешно реализирана како современо, интерактивно и кориснички ориентирано решение за управување со задачи. Преку внимателно избраните технологии, пред сè SolidJS и Tailwind CSS, беа постигнати високи перформанси, реактивност и визуелна уредност. Архитектурата на апликацијата е модуларна, со јасно поделени компоненти и добро организиран код, што овозможува лесна одржливост и понатамошно проширување.

Главните функционалности, како додавање, уредување, бришење, пребарување и филтрирање задачи, како и автоматски потсетници и drag-and-drop, беа успешно имплементирани со стабилна логика и одличен кориснички интерфејс. Реактивниот модел на податоци преку createSignal и createEffect ја направи апликацијата многу флуидна и брза во реално време.

Со оваа апликација се демонстрира способност за користење на современи технологии, организација на код, и креирање решение кое е практично, лесно за користење и визуелно привлечно.