

ФИНКИ – ПРЕДМЕТ: НАПРЕДЕН ВЕБ ДИЗАЈН

# TicTacBoom

Стефанија Лазарева

# Содржина

1

Архитектура на апликацијата

2

Компоненти

3

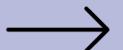
Функционалности и дизајн

4

Податоци и состојби

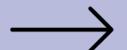
# Вовед

- TicTacBooM е веб-апликација изградена со SolidJS
- Наменета е за управување со содржини преку картички и категории
- Овозможува филтрирање, пребарување и уредување на задачи преку модерен и одзивен интерфејс
- Апликацијата комбинира едноставност со техничка стабилност
- Дизајнирана е како практична продуктивна алатка и едукативен пример за реактивно програмирање, состојбен менаџмент и модуларен дизајн



# Цели и мотивација

- Изградба на едноставна, но функционална платформа за управување со задачи
- Демонстрација на можностите на SolidJS
- Дизајн на апликација со минимален број библиотеки и алатки - едноставност
- Фокус на перформанси, скалабилност и одржливост на кодот
- Примена во: лична употреба, едукација, тимска продуктивност
- Практично искуство со: управување на состојба, динамично прикажување на податоци, интерактивни компоненти и форми



# Технологии и алатки

## SolidJS

- Модерен front-end JavaScript framework
- Користи compile-time reactivity
- Манипулира директно со реалниот DOM
- Обезбедува високи перформанси и интерактивен UI

## Vite

- Модерен build tool
- Обезбедува брзо стартување и Hot Module Replacement (HMR)
- Промени веднаш видливи во прелистувачот
- Зголемена продуктивност при развој

## CSS & Tailwind

- Одзивен (responsive) дизајн
- Tailwind CSS: утврдена utility-first библиотека
- Брзо и доследно стилизирање на компонентите

# Архитектура на апликацијата

- TicTacBoom е едноставна, но моќна веб-апликација за менаџирање на задачи
- Изградена како Single Page Application (SPA) – интеракција без повторно вчитување на страната
- Податоците се складираат локално преку LocalStorage API – без сервер или база
- Користи компонентно-базирана архитектура:
  - Задачи, филтри, модални прозорци → посебни компоненти
- Овозможува повторна употреба, лесно тестирање и одржување на кодот
- Обезбедува флуидно, брзо и интерактивно корисничко искуство



# ГлавНИ КОМПОНЕНТИ

- App – главна компонента – иницијализација на податоци – ги содржи и поврзува сите подкомпоненти
- List – ги прикажува сите задачи – користи For loop за реактивно рендерирање
- Card – поединечна картичка – уредување, бришење, менување статус
- CardModal – модално прозорче за додавање задачи – валидација и createSignal за состојби
- SearchBar – пребарување картички – реактивна логика преку createMemo
- CategoryFilter – филтрирање по категории или статус



# Централна логика на апликацијата

- Главна логичка точка на апликацијата
- Управува со глобалната состојба и корисничките интеракции
- Оркестрира прикажување на сите компоненти
- Главни функции:
  - handleSearch(query) – пребарување
  - handleCategorySelect(category) – избор на категорија
  - handleAddCategory(newCategory) – додавање категорија
  - handleSaveCard(...), handleDeleteCard(...), handleMoveCard(...) – манипулации со задачи
  - filteredLists() – реактивно филтрирање според пребарување и категории



# Картичка со задача

- Визуелна и интерактивна единица за прикажување на една задача
- Самостојна, повторно употреблива и реактивна компонента
- Функционалности:
  - Приказ на детали: слики, категории, етикети, рокови
  - Потсетници: визуелен приказ ако reminder е активен
  - Рок: автоматско означување со боја (overdue, today, tomorrow)
  - Брзо бришење: handleQuickDelete()
  - Споделување: Web Share API или fallback notification
  - Грешка при вчитување слика: прикажување default placeholder
  - Drag & Drop: логика за влечење и пуштање картички



# Модален прозорец за задача

- Централен елемент за креирање/уредување задачи
- Поддржува внес на сите атрибути: наслов, опис, категорија, етикети, слика, рок, потсетник
- Клучни функционалности:
  - Зачувување (handleSave) – гради и враќа финален card објект
  - Бришење (handleDeleteClick) – достапно само за постоечки задачи
  - Откажување (handleClear) – затвора модалот без промени
  - Клучеви (Enter, Escape) – за брзи команди
  - Автоматски фокус на поле за наслов при отворање
  - Додавање нова категорија – динамично поле при избор



# Додавање нова категорија

- Модален прозорец за внес и додавање нова категорија
- Поддршка за организација на задачи преку категории
- Го враќа внесеното име назад преку props.onAdd
- Клучни функционалности:
  - Внес на име → handleInputChange ја ажурира состојбата newCategoryName
  - Потврда (handleAddClick) → Тримирање + props.onAdd(trimmedName) → Затворање на модал
  - Откажување (handleCancelClick) → Повик на props.onClose()
  - Клучеви (handleKeyDown):
    - Enter = додавање
    - Escape = затворање



# Табла со задачи

- Прикажува група на задачи поврзани со категорија
- Копче за додавање нова задача
- Поддршка за drag-and-drop преместување меѓу табли
- Користи Card.jsx за прикажување задачи
- Главни функции:
  - <For each={cardsToDisplay()}> → реактивно прикажување на задачи
  - handleDragOver → дозволува drag drop + визуелна индикација
  - handleDragLeave → отстранува ефект на влечење
  - handleDrop → чита cardId и sourceListId, инициира преместување



# Филтрирање по категории

- Овозможува избор на категорија за филтрирање на задачите
- Дава опција за чистење на филтерот со едно копче
- Функционалности:
  - Клик на копче → повикува onSelect(category)
  - Визуално означување на избраната категорија (classList.selected)
  - Динамично рендерирање на копчиња со <For each={categories}>
  - Условно прикажување на копчето „Clear filter“ со <Show when={selectedCategory}>
  - Клик на „Clear filter“ → ресетира филтер (onSelect("))



# Пребарување на задачи

- Дава интерактивно поле за внесување текст за пребарување задачи
- Овозможува динамично филтрирање според внесените зборови
- Внатрешна логика:
  - Локална состојба: inputValue преку createSignal
  - createEffect – ажурира inputValue ако се промени searchQuery од надвор
  - handleInputChange – при секоја промена:
  - ја ажурира локалната состојба
  - повикува onSearch(query) за филтрирање во родителската компонента



# ФУНКЦИОНАЛНОСТИ

## Добавање задачи

Креирање нова задача преку „+ Add Task“ и пополнување со детали во модален прозорец (CardModal).

## Уредување и бришење

Модификација преку CardModal; бришење со × копче или „Delete“ во модалот.

## Категоризирање и филтрирање

Избор или додавање нови категории (AddCategoryModal) и филтрирање со CategoryFilter.

## Пребарување

Динамично пребарување по наслов, опис, етикети и категории со SearchBar.

## Drag-and-Drop

Преместување задачи помеѓу листи, управувано од Card.jsx, List.jsx и handleMoveCard() во App.jsx.

## Потсетници и рокови

Следење на due date и reminder со визуелна индикација и нотификацији.

## Споделување

Web Share API или алтернативна нотификација при споделување задача.

## Перзистентност

Сите податоци се зачувуваат во LocalStorage за трајно чување по освежување на страната.

# Останати функции

formatReminderTi  
me(isoString)

Форматира ISO датум  
за потсетник во  
разбиралив временски  
приказ.

formatDueDate(iso  
String)

Претвора ISO датум за  
краен рок во описна  
фраза.

getDueDateStatus(  
isoString)

Враќа краток статус  
(overdue, today,  
tomorrow, later) за  
визуелно означување  
на рокови.

isReminderDue(isoS  
tring)

Проверува дали  
потсетникот е веќе  
активен (времето е  
минато).

setupReminders(list,  
onReminderDueCallback)

Закажува автоматски  
потсетници за сите  
задачи со reminder, со  
помош на setTimeout.

showBrowserNotifi  
cation(card)

Прикажува системска  
нотификација преку  
Browser Notification API,  
ако е овозможено.

# Дизајн

- Чист и модерен интерфејс – пријатно и интуитивно корисничко искуство.
- Принципи на дизајн:
  - Јасност и едноставност – лесно препознатливи елементи
  - Фокус на содржината – задачи во центарот на вниманието
  - Конзистентност – повторна употреба на компоненти и стилови
  - Интерактивност – визуелна повратна информација при кориснички дејства
- Техничка имплементација:
  - Tailwind CSS + сопствени CSS класи (style.css, App.module.css, index.css)



# Управување со состојби

- Главен податочен модел:
  - Lists: Колекции со задачи (id, title)
  - Cards: Задачи со атрибути (id, наслов, опис, категорија, етикети, слика, икона, потсетник, рок)
- SolidJS механизми за реактивност:
  - createSignal за основно следење и ажурирање
  - createEffect за извршување логика при промени
  - createMemo за филтрирање и пребарување
- Перзистентност:
  - Податоците се чуваат во LocalStorage, без сервер или база.
- Локални состојби:
  - Секоја компонента управува со свои состојби за подобра енкапсулација и одржливост.



# Заклучок

- Современа и интерактивна апликација за управување со задачи.
- Користени се SolidJS и Tailwind CSS за високи перформанси, реактивност и уреден визуелен дизајн.
- Архитектурата е модуларна, со јасно поделени компоненти и организиран код, што овозможува лесна одржливост и проширување.
- Стабилно имплементирани функционалности(додавање, уредување, бришење, пребарување, филтрирање, потсетници и drag-and-drop)
- Реактивниот модел со createSignal и createEffect обезбедува брза и флуидна интеракција во реално време.



**Ви благодарим на вниманието !**