ДОДАТОК А

Графічний матеріал кваліфікаційної роботи

Харківський національний університет радіоелектроніки
Кафедра ЕОМ

Кваліфікаційна робота
Перший (бакалаврський) рівень

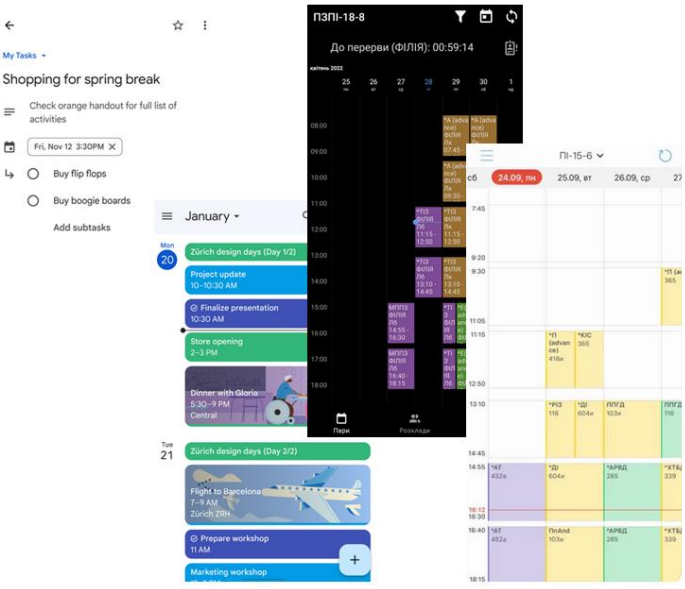
КРОСПЛАТФОРМНИЙ ЗАСТОСУНОК ДЛЯ ОРГАНІЗАЦІЇ НАВЧАЛЬНОГО ПРОЦЕСУ

Автор:
Віктор Волошко,
ст. гр. КІУКІ-21-5

Керівник:
Єгор Корнієнко,
асистент кафедри ЕОМ

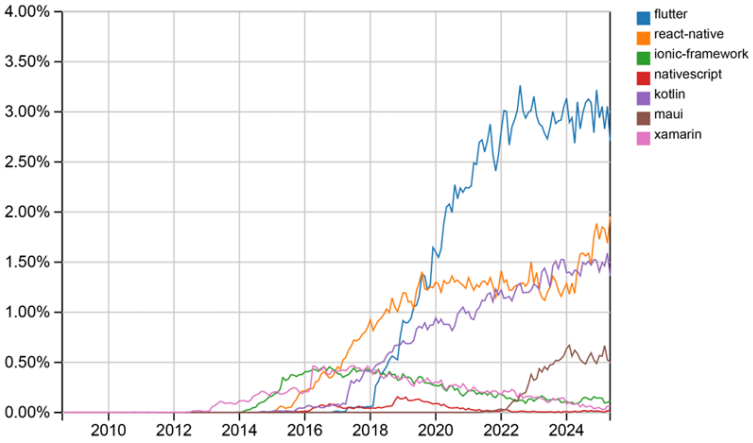
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

- Велика кількість рішень на ринку
- Відсутність єдиного функціонального рішення
- Один застосунок краще за декілька
- Кросплатформність – це потреба



АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

- Різні мови програмування
- Різний підхід до створення інтерфейсу користувача
- Різна продуктивність застосунків
- Популярні рішення майже завжди краще



- Календар та список задач
- Отримання та відображення розкладу
- Можливість додавати, змінювати, редагувати та видаляти події та задачі
- Сучасний фундамент застосунку
- Підтримка популярних мобільних платформ

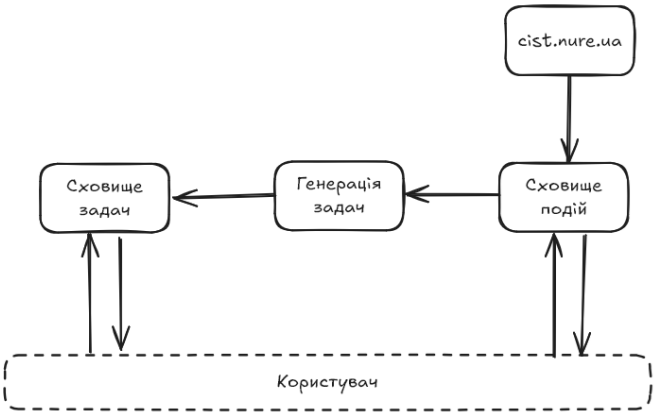
МЕТА РОБОТИ

ПІДХІД ДО ВИРІШЕННЯ

Сервіс cist.nure.ua надає API для отримання даних.

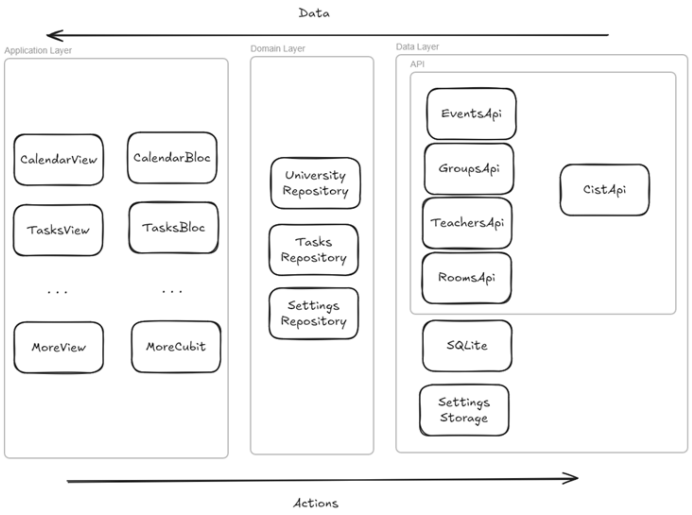
Для розкладу: зберігання та відображення.

Для задач: генерація, збереження та відображення.



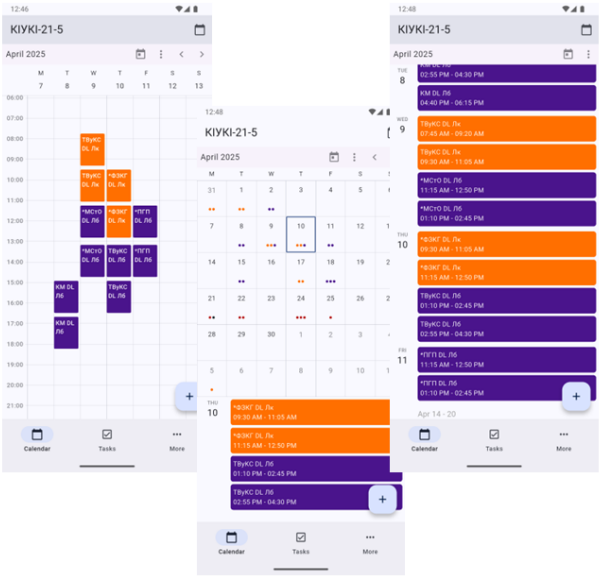
АРХІТЕКТУРА ЗАСТОСУНКУ

- Архітектурний патерн MVVM
- Єдине локальне сховище для задач та подій
- Реактивне програмування



ФУНКЦІОНАЛ: КАЛЕНДАР

- Інтуїтивний
- Зручний незалежно від вподобань
- Швидке сприйняття усієї важливої інформації

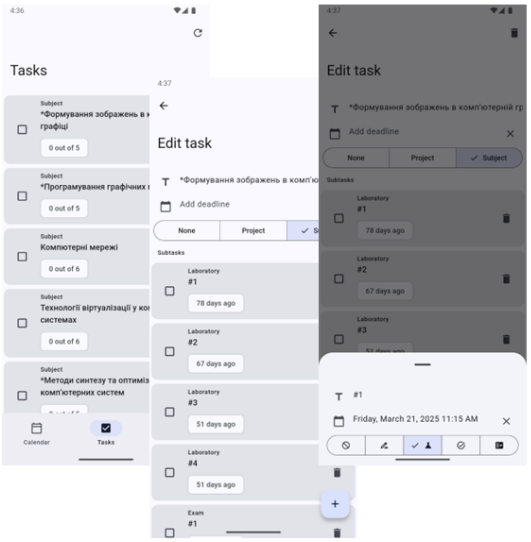


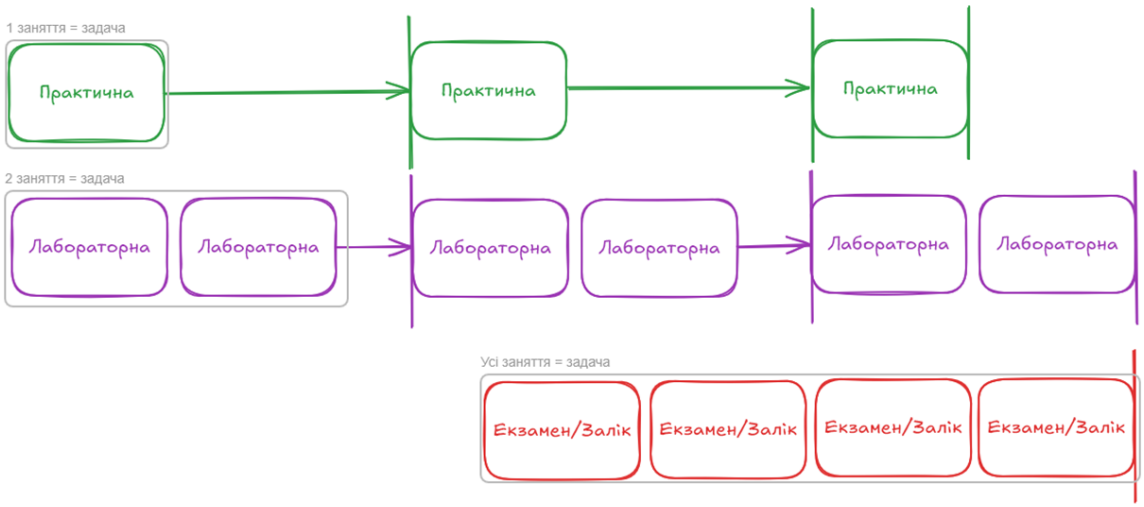
ФУНКЦІОНАЛ: ВЗАЄМОДІЇ З ПОДІЯМИ

- Повна інформація про події
- Створення, редагування та видалення подій
- Вибір предмету, часу та типу події

ФУНКЦІОНАЛ: ЗАДАЧІ

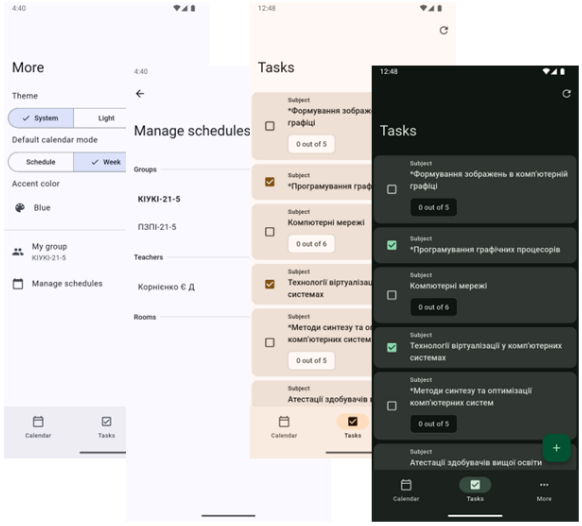
- Компактний перелік задач
- Створення, редагування та видалення задач
- Наявність підзадач для кращої структуризації
- Генерація задач на основі розкладу





ФУНКЦІОНАЛ: НАЛАШТУВАННЯ

- Вибір розкладів для збереження
- Вибір групи користувача
- Вибір теми та кольору акценту
- Вибір режиму календаря за замовчуванням

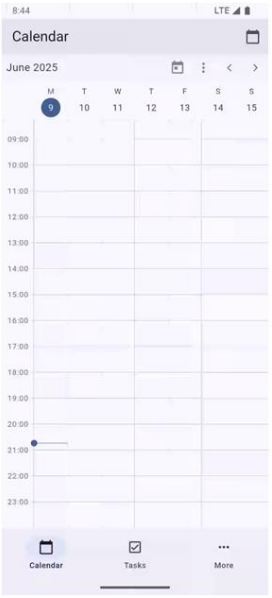


- Застосунок виконує поставлені задачі
- Застосунок зручніший та функціональніший за аналоги
- Кодова база застосунку є легкою для підтримки, відкритою для розширення

- Найбажанішим покращенням є додання функціоналу для синхронізації
- Корисною була б інтеграція з іншими календарями, такими як Google Календар
- Деякі користувачі бажають побачити функціонал тегів та пріоритетів для задач
- Для охоплення більшої аудиторії варто можна збільшити набір підтримуваних платформ

ПІДСУМОК

ДЕМОНСТРАЦІЯ



ДОДАТОК Б

Код проєкту

Лістинг Б.1 – Код TasksRepository

```
class TasksRepository {
  TasksRepository({required db.DriftDB driftDB}) : _driftDB =
driftDB {
    _init();
  }

  final db.DriftDB _driftDB;

  final _tasksStreamController =
BehaviorSubject<List<Supertask>>.seeded(
    const [],
  );

  late final StreamSubscription<List<db.Task>>
_tasksSubscription;

  Stream<List<Supertask>> get tasks =>
      _tasksStreamController.asBroadcastStream();

  Future<int> saveTask(Task task, int supertaskID) =>
      _driftDB.saveTask(task.toDBModel(supertaskID));

  Future<int> saveSupertask(Supertask task) =>
      _driftDB.saveTask(task.toDBModel());

  Future<int> saveSupertaskWithSubtasks(Supertask task) async {
    _driftDB.saveTasks(task.subtasksToDBModels(task.id));
    return _driftDB.saveTask(task.toDBModel());
  }

  Future<void> saveSupertasks(Iterable<Supertask> tasks) async {
    final subtasks = <db.TasksCompanion>[];

    for (final supertask in tasks) {
      final id = await _driftDB.saveTask(supertask.toDBModel());
      subtasks.addAll(supertask.subtasksToDBModels(id));
    }

    return _driftDB.saveTasks(subtasks);
  }

  Future<int> deleteTask(int id) => _driftDB.deleteTask(id);

  Future<int> deleteSupertask(int id) =>
```

```
_driftDB.deleteSupertask(id);

  Future<int> deleteGeneratedTasks() =>
_driftDB.deleteGeneratedTasks();

  Future<dynamic> dispose() async {
    await _tasksSubscription.cancel();
    return _tasksStreamController.close();
  }

  void _init() {
    _tasksSubscription = _driftDB.loadTasks().listen((tasks) {
      final result = <Supertask>[];

      final supertasks = tasks.where((task) => task.supertaskID
== null);
      final subtasks = tasks.where((task) => task.supertaskID !=
null);

      result.addAll(
        supertasks.map(
          (e) => Supertask.fromDBModel(
            e,
            subtasks.where((task) => task.supertaskID == e.id),
          ),
        ),
      );

      _tasksStreamController.add(result);
    });
  }
}
```

Лістинг Б.2 – Код UniversityRepository

```
class SupertaskViewBloc extends Bloc<SupertaskViewEvent,
SupertaskViewState> {
  SupertaskViewBloc({required TasksRepository tasksRepository})
    : _tasksRepository = tasksRepository,
      super(const SupertaskViewInitial()) {

on<SupertaskViewSubscriptionRequested>(_onSubscriptionRequested)
;
    on<SupertaskViewDataChanged>(_onDataChanged);

on<SupertaskViewSubtaskCheckboxToggled>(_onSubtaskCheckboxToggle
d);

on<SupertaskViewSupertaskDeletionRequested>(_onSupertaskDeletion
Requested);

on<SupertaskViewSubtaskDeletionRequested>(_onSubtaskDeletionRequ
ested);
```

```
on<SupertaskViewSubtaskCreationRequested>(_onSubtaskCreationRequ
ested);
  }

  final TasksRepository _tasksRepository;

  Future<void> _onSubscriptionRequested(
    SupertaskViewSubscriptionRequested event,
    Emitter<SupertaskViewState> emit,
  ) async {
    emit(const SupertaskViewLoading());

    await emit.forEach(
      _tasksRepository.tasks,
      onData: (supertasks) {
        final supertask = supertasks.firstWhereOrNull(
          (task) => task.id == event.taskID,
        );

        if (supertask == null) {
          return const SupertaskViewSupertaskDeleted();
        } else {
          return SupertaskViewSuccess(
            task: supertask,
            titleError: _checkTitle(supertask.title),
          );
        }
      },
    );
  }

  void _onDataChanged(
    SupertaskViewDataChanged event,
    Emitter<SupertaskViewState> emit,
  ) {
    if (_checkTitle(event.title) != null) {
      emit(
        SupertaskViewSuccess(
          task: (state as SupertaskViewSuccess).task,
          titleError: _checkTitle(event.title),
        ),
      );
    } else {
      _tasksRepository.saveSupertask(
        (state as SupertaskViewSuccess).task.rewriteWith(
          title: event.title,
          type: event.type,
          deadline: event.deadline,
        ),
      );
    }
  }
```

```
  void _onSubtaskCheckboxToggled(
    SupertaskViewSubtaskCheckboxToggled event,
    Emitter<SupertaskViewState> emit,
  ) => _tasksRepository.saveTask(
    event.task.copyWith(isDone: event.isDone),
    (state as SupertaskViewSuccess).task.id!,
  );

  void _onSupertaskDeletionRequested(
    SupertaskViewSupertaskDeletionRequested event,
    Emitter<SupertaskViewState> emit,
  ) async => _tasksRepository.deleteSupertask(event.id);

  void _onSubtaskDeletionRequested(
    SupertaskViewSubtaskDeletionRequested event,
    Emitter<SupertaskViewState> emit,
  ) => _tasksRepository.deleteTask(event.id);

  Future<void> _onSubtaskCreationRequested(
    SupertaskViewSubtaskCreationRequested event,
    Emitter<SupertaskViewState> emit,
  ) async => emit(
    SupertaskViewSubtaskCreated(
      task: (state as SupertaskViewSuccess).task,
      subtaskID: await _tasksRepository.saveTask(
        Task(
          title: 'New subtask',
          isDone: false,
          isGenerated: true,
          type: null,
          deadline: null,
        ),
        (state as SupertaskViewSuccess).task.id!,
      ),
    ),
  );

  TitleError? _checkTitle(String title) =>
      title.trim().isEmpty ? TitleError.emptyOrWhitespace :
null;
}
```

Лістинг Б.3 – Код генерації задач

```
Future<void> _onGenerationRequested(
    TasksOverviewGenerationRequested event,
    Emitter<TasksOverviewState> emit,
  ) async {
    await _tasksRepository.deleteGeneratedTasks();

    final result = <Supertask>[];
```

```
    final userGroupID = await
_universityRepository.userGroupID.first;
    final groupEvents =
        (await _universityRepository.events.first)
            .where((e) => e.groups.contains(userGroupID))
            .toList();
    final subjectIDs = <int>{};
    subjectIDs.addAll(groupEvents.map((e) => e.subject));
    final subjects =
        (await _universityRepository.subjects.first)
            .where((subject) => subjectIDs.contains(subject.id))
            .toList();

    for (final subject in subjects) {
      final subjectEvents = groupEvents.where(
        (event) => event.subject == subject.id,
      );
      final subjectTasks = <Task>[
        ..._createTasksForType(subjectEvents,
EventBaseType.practice, 1),
        ..._createTasksForType(subjectEvents,
EventBaseType.laboratory, 2),
        ..._createTasksForType(subjectEvents,
EventBaseType.exam),
      ];
      result.add(
        Supertask(
          title: subject.title,
          isDone: false,
          isGenerated: true,
          type: TaskType.subject,
          deadline: null,
          subtasks: subjectTasks,
        ),
      );
    }

    _tasksRepository.saveSupertasks(result);
  }

List<Task> _createTasksForType(
    Iterable<Event> subjectEvents,
    EventBaseType type, [
    int? eventsPerTask,
  ]) {
    if (subjectEvents.isEmpty) return const [];
    if (type == EventBaseType.exam) {
      final examEvents = subjectEvents.where(
        (event) => event.baseType == EventBaseType.exam,
      );

      if (examEvents.isEmpty) return const [];
```

```
      return [
        Task(
          title: '#1',
          isDone: false,
          isGenerated: true,
          type: type.toTaskType(),
          deadline: examEvents.last.endTime,
        ),
      ];
    }

    final result = <Task>[];
    final eventsOfType =
        subjectEvents.where((event) => event.baseType ==
type).toList();
    if (eventsOfType.isEmpty) return const [];

    for (int index = 0; index < eventsOfType.length; index +=
eventsPerTask) {
      result.add(
        Task(
          title: '#${(index + eventsPerTask!) ~/
eventsPerTask}',
          isDone: false,
          isGenerated: true,
          type: type.toTaskType(),
          deadline:
              ((index + eventsPerTask) < eventsOfType.length)
                  ? eventsOfType[index +
eventsPerTask].startTime
                  : eventsOfType[index + eventsPerTask -
1].endTime,
        ),
      );
    }

    return result;
}
```