Documentación Técnica Avances proyecto final

Backend API + Frontend (React)

Diego Rafael Mairena López

1. Resumen

Unitask es una aplicación para la gestión de materias y tareas con recordatorios. El frontend está desarrollado en React y consume una API REST. La API ya cuenta con autenticación mediante Google (endpoint /auth/google).

2. Autenticación

Método de autenticación: Google Sign-In con intercambio de token en el backend.

Endpoint: POST /auth/google

Flujo:

1) El frontend obtiene el credential (ID token) de Google.

2) Envía el credential al backend en POST /auth/google.

3) El backend valida el token con Google, registra o actualiza al usuario y responde con un JWT propio.

4) El frontend almacena el JWT en localStorage y lo envía como Authorization: Bearer <token> en las siguientes solicitudes.

Ejemplo de solicitud (JSON):

```
{
  "credential": "<google_id_token>"
}
```

Ejemplo de respuesta (JSON):

```
{
  "token": "<jwt>",
  "user": { "id": 1, "nombre": "Diego", "email": "diego@example.com" }
}
```

3. Endpoints

3.1 Auth

POST /auth/google

3.2 Etiquetas

GET /etiquetas

POST /etiquetas

3.3 Materias
GET /materias

POST /materias

GET /materias/mias

3.4 RecordatorioOps
PATCH /recordatorios/{id}

DELETE /recordatorios/{id}

3.5 Recordatorios
GET /tareas/{tareaId}/recordatorios

POST /tareas/{tareaId}/recordatorios

3.6 Tareas
GET /tareas

POST /tareas

PATCH /tareas/{id}

DELETE /tareas/{id}

POST /tareas/{id}/etiquetas/{etiquetaId}

4. Modelos de datos
Usuario: id, nombre, email, foto_url (opcional), creado_en, actualizado_en

Materia: id, nombre, usuario_id

Etiqueta: id, nombre, color_hex (opcional), usuario_id

Tarea: id, materia_id, titulo, descripcion, vence_en, prioridad(Alta/Media/Baja), completada, silenciada, eliminada

Recordatorio: id, tarea_id, minutos_antes, activo, enviado_en

5. Especificación por recurso (cuerpos de ejemplo)

5.1 Etiquetas
POST /etiquetas (body):

```
{
 "nombre": "Examen",
 "color_hex": "#2563EB"
}
```

Respuesta 201:

```
{
 "id": 7,
 "nombre": "Examen",
 "color_hex": "#2563EB",
 "usuario_id": 1
}
```

5.2 Materias

POST /materias (body):

```
{
 "nombre": "Programación Web II"
}
```

Respuesta 201:

```
{
 "id": 3,
 "nombre": "Programación Web II",
 "usuario_id": 1
}
```

5.3 Tareas

POST /tareas (body):

```
{
 "materia_id": 3,
 "titulo": "Informe de redes",
 "descripcion": "Capas OSI y topologías LAN",
 "vence_en": "2025-10-20T14:00:00Z",
 "prioridad": "Alta"
}
```

PATCH /tareas/{id} (body, parcial):

```
{
 "completada": true,
 "silenciada": false
}
```

POST /tareas/{id}/etiquetas/{etiquetaId}: No tiene body; asocia la etiqueta a la tarea.

5.4 Recordatorios

POST /tareas/{tareaId}/recordatorios (body):

```
{
  "minutos_antes": 30
}
```

PATCH /recordatorios/{id} (body, parcial):

```
{
  "activo": false
}
```

6. Seguridad y encabezados

Enviar en cada solicitud autenticada: Authorization: Bearer <jwt>

Content-Type: application/json en operaciones con body.

Todas las operaciones que modifican recursos requieren un usuario autenticado.

7. Códigos de estado

200 OK: Respuestas exitosas de lectura o actualización.

201 Created: Recurso creado exitosamente.

204 No Content: Recurso eliminado.

400 Bad Request: Datos inválidos.

401 Unauthorized: Token ausente o inválido.

403 Forbidden: Prohibido para el usuario actual.

404 Not Found: Recurso inexistente.

409 Conflict: Conflicto al crear o asociar recursos.

422 Unprocessable Entity: Validaciones de dominio.

500 Internal Server Error: Error inesperado.

8. Frontend (React)

Componentes principales: App, LoginComponent, HomeScreen, TutorialOverlay.

Integración con API a través de un módulo api.js con funciones listMateriasMine, listTareas, listEtiquetas, createTarea, patchTarea, deleteTarea, listRecordatorios, createRecordatorio, patchRecordatorio, deleteRecordatorio.

El frontend guarda el token JWT en localStorage y lo adjunta en cada solicitud mediante Authorization: Bearer <jwt>.

Incluye: filtros, orden, progreso por materia, modal de nueva tarea y notificaciones locales.

9. Roadmap inmediato

1) Conectar todos los métodos del módulo api.js a los endpoints listados.

2) Manejo centralizado de errores y expiración de token (interceptor).

3) Soporte offline con IndexedDB y sincronización cuando haya red.

4) Tests de integración en backend y pruebas e2e básicas en frontend.