Task Management API Request Handling Sequence Diagrams

This document contains sequence diagrams for the task management API endpoints, showing the interaction flow between frontend applications, API service, and database with simplified internal processing.

Participants Legend

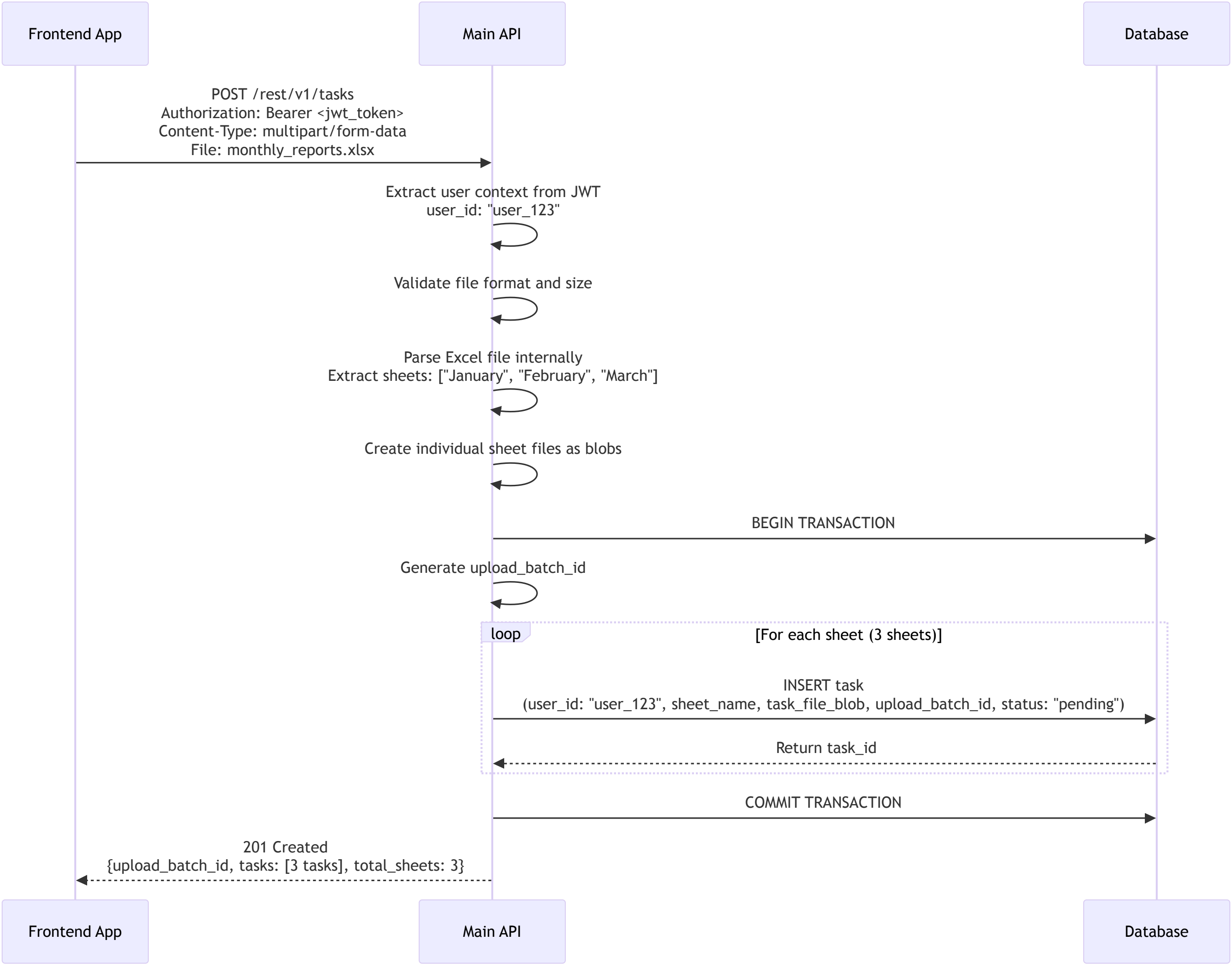
- **Frontend**: Internal frontend applications (Task Management UI)
- **API**: Main task management API service (handles all processing internally)
- **DB**: Primary Database (tasks and files)

Endpoints

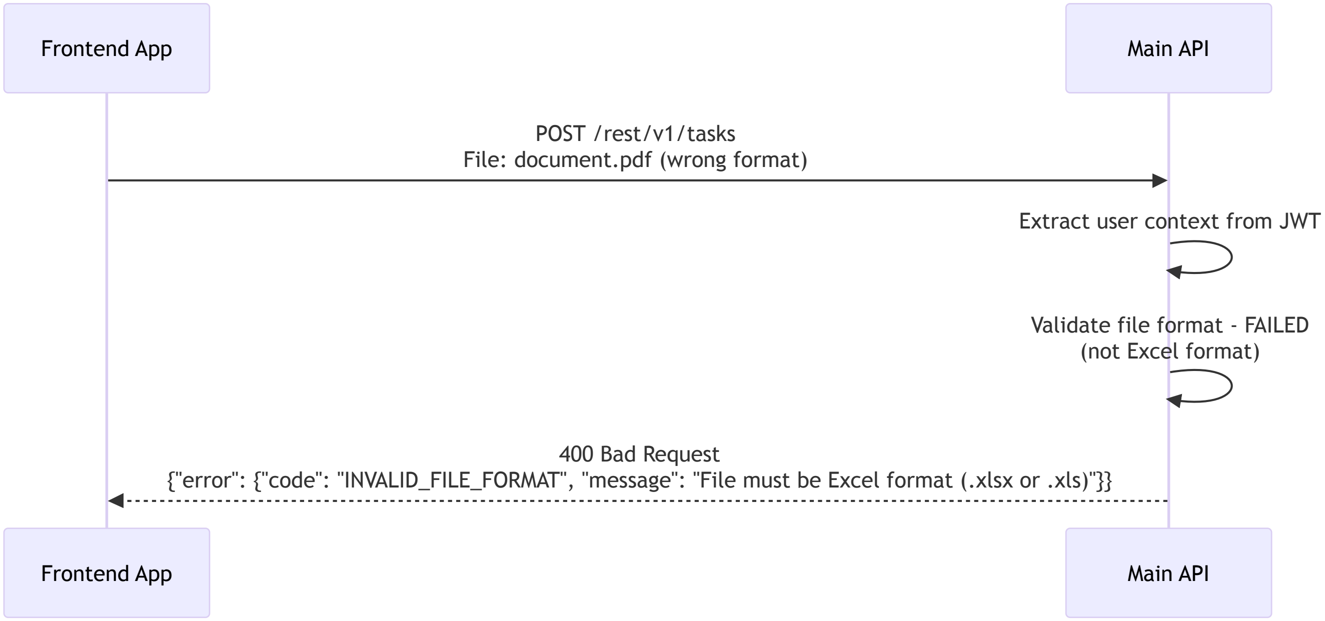
Method	Endpoint	Description	Auth Level
POST	`/rest/v1/tasks`	Upload Excel file and create tasks	User
GET	`/rest/v1/tasks`	List user's tasks with filtering (no blob data)	User
GET	`/rest/v1/tasks/{id}`	Get specific task details with blob files	User
PUT	`/rest/v1/tasks/{id}`	Update/cancel a task	Owner
DELETE	`/rest/v1/tasks/{id}`	Delete a task	Owner

1. POST /rest/v1/tasks - Excel File Upload and Task Creation

Success Flow - Multi-Sheet Excel Upload

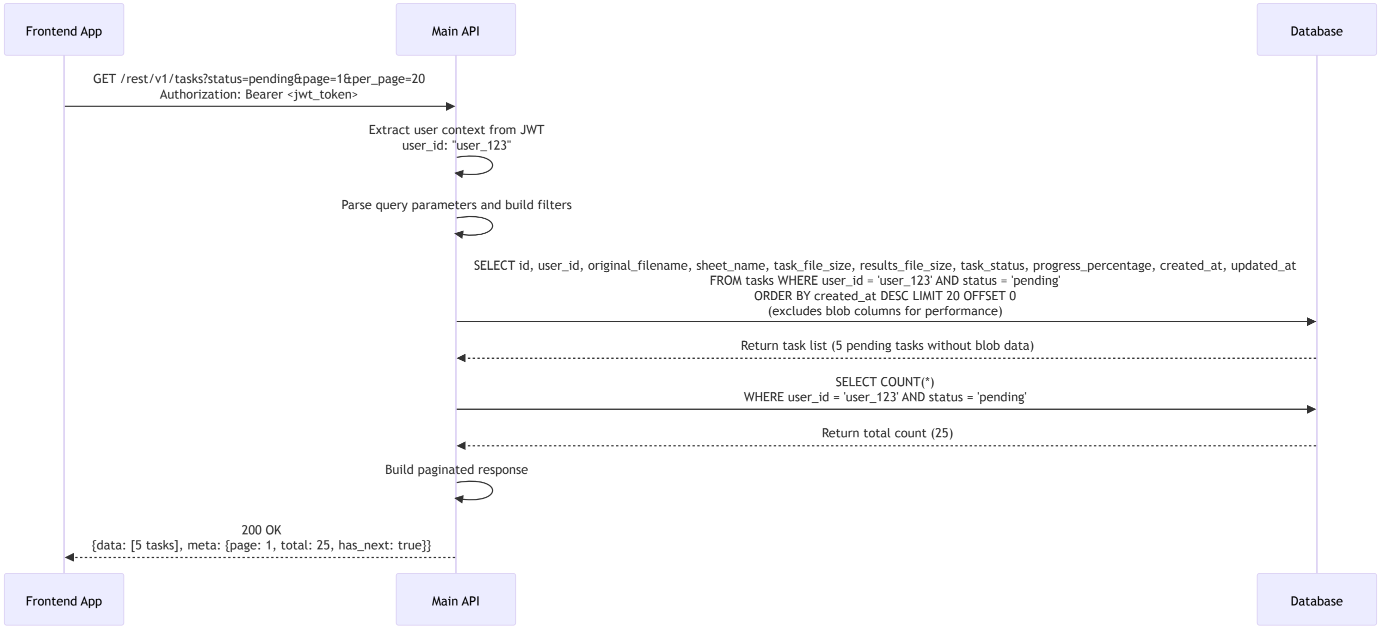

Frontend App
Main API
Database
POST /rest/v1/tasks
Authorization: Bearer <jwt_token>
Content-Type: multipart/form-data
File: monthly_reports.xlsx
Extract user context from JWT
user_id: "user_123"
Validate file format and size
Parse Excel file internally
Extract sheets: ["January", "February", "March"]
Create individual sheet files as blobs
BEGIN TRANSACTION
Generate upload_batch_id
loop
[For each sheet (3 sheets)]
INSERT task
(user_id: "user_123", sheet_name, task_file_blob, upload_batch_id, status: "pending")
Return task_id
COMMIT TRANSACTION
201 Created
{upload_batch_id, tasks: [3 tasks], total_sheets: 3}
Frontend App
Main API
Database

Error Flow - Invalid Excel File


Frontend App
Main API
POST /rest/v1/tasks
File: document.pdf (wrong format)
Extract user context from JWT
Validate file format - FAILED
(not Excel format)
400 Bad Request
{"error": {"code": "INVALID_FILE_FORMAT", "message": "File must be Excel format (.xlsx or .xls)"}}
Frontend App
Main API

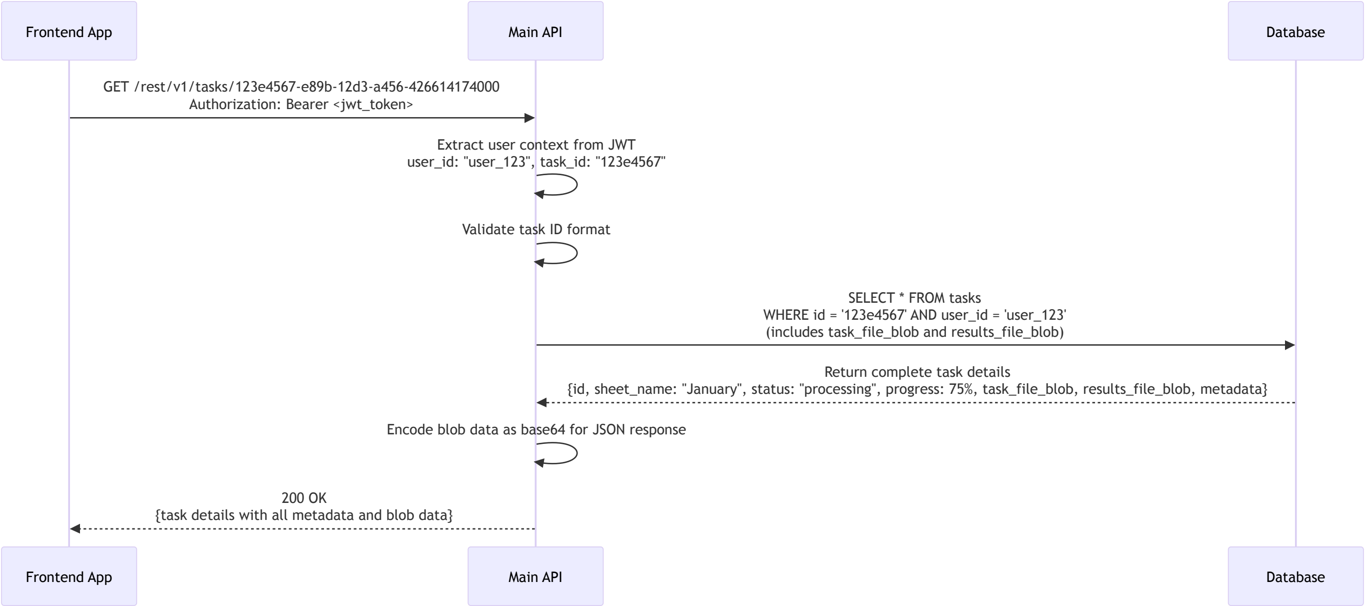
2. GET /rest/v1/tasks - List User Tasks (No Blob Data)

Success Flow with Filtering

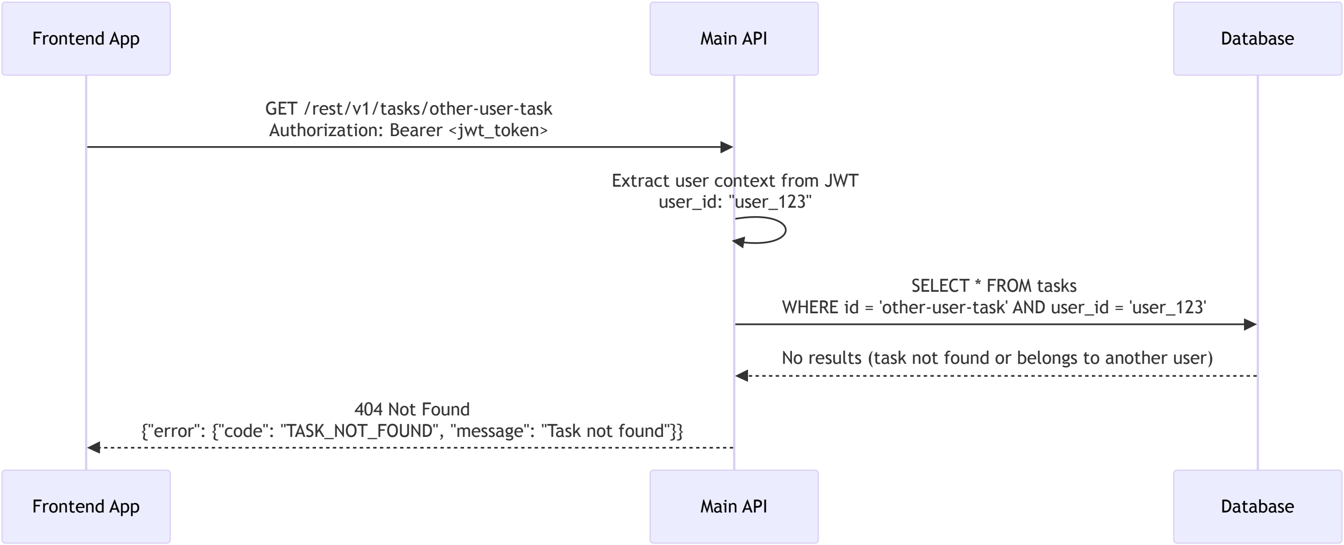

Frontend App
Main API
Database
GET /rest/v1/tasks?status=pending&page=1&per_page=20
Authorization: Bearer <jwt_token>
Extract user context from JWT
user_id: "user_123"
Parse query parameters and build filters
SELECT id, user_id, original_filename, sheet_name, task_file_size, results_file_size, task_status, progress_percentage, created_at, updated_at
FROM tasks WHERE user_id = 'user_123' AND status = 'pending'
ORDER BY created_at DESC LIMIT 20 OFFSET 0
(excludes blob columns for performance)
Return task list (5 pending tasks without blob data)
SELECT COUNT(*)
WHERE user_id = 'user_123' AND status = 'pending'
Return total count (25)
Build paginated response
200 OK
{data: [5 tasks], meta: {page: 1, total: 25, has_next: true}}
Frontend App
Main API
Database

3. GET /rest/v1/tasks/{id} - Get Task Details with Blob Data

Success Flow

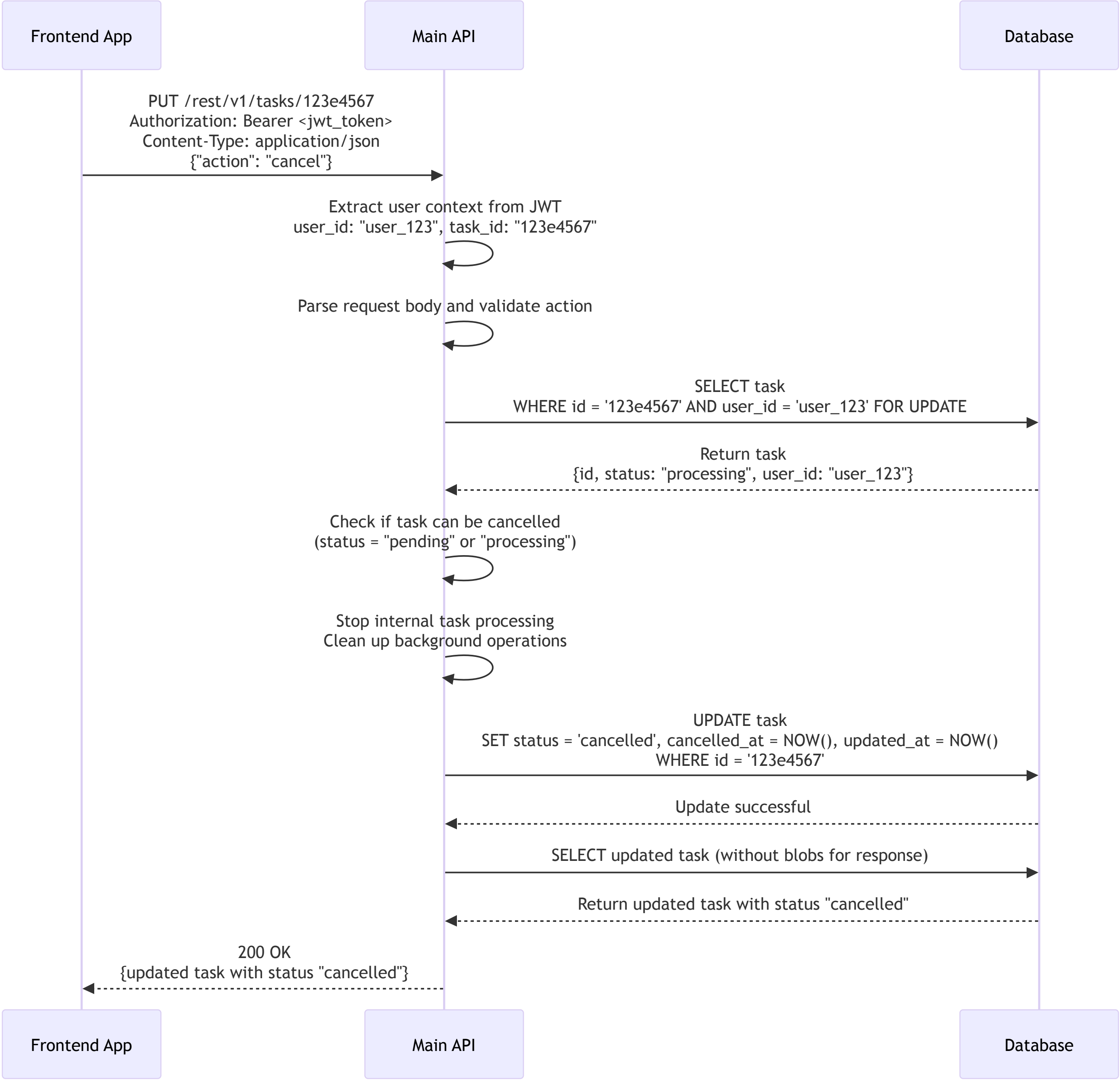

Frontend App
Main API
Database
GET /rest/v1/tasks/123e4567-e89b-12d3-a456-426614174000
Authorization: Bearer <jwt_token>
Extract user context from JWT
user_id: "user_123", task_id: "123e4567"
Validate task ID format
SELECT * FROM tasks
WHERE id = '123e4567' AND user_id = 'user_123'
(includes task_file_blob and results_file_blob)
Return complete task details
{id, sheet_name: "January", status: "processing", progress: 75%, task_file_blob, results_file_blob, metadata}
Encode blob data as base64 for JSON response
200 OK
{task details with all metadata and blob data}
Frontend App
Main API
Database

Error Flow - Task Not Found or Access Denied

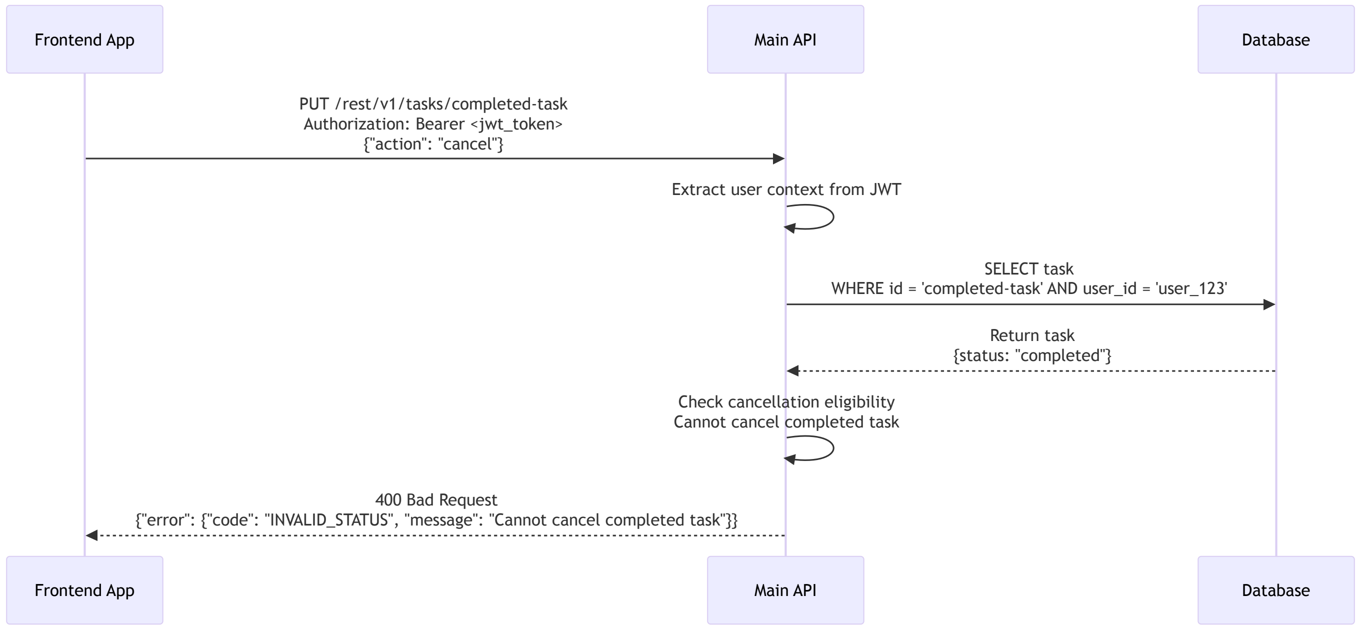

Frontend App
Main API
Database
GET /rest/v1/tasks/other-user-task
Authorization: Bearer <jwt_token>
Extract user context from JWT
user_id: "user_123"
SELECT * FROM tasks
WHERE id = 'other-user-task' AND user_id = 'user_123'
No results (task not found or belongs to another user)
404 Not Found
{"error": {"code": "TASK_NOT_FOUND", "message": "Task not found"}}
Frontend App
Main API
Database

4. PUT /rest/v1/tasks/{id} - Update/Cancel Task

Success Flow - Cancel Task

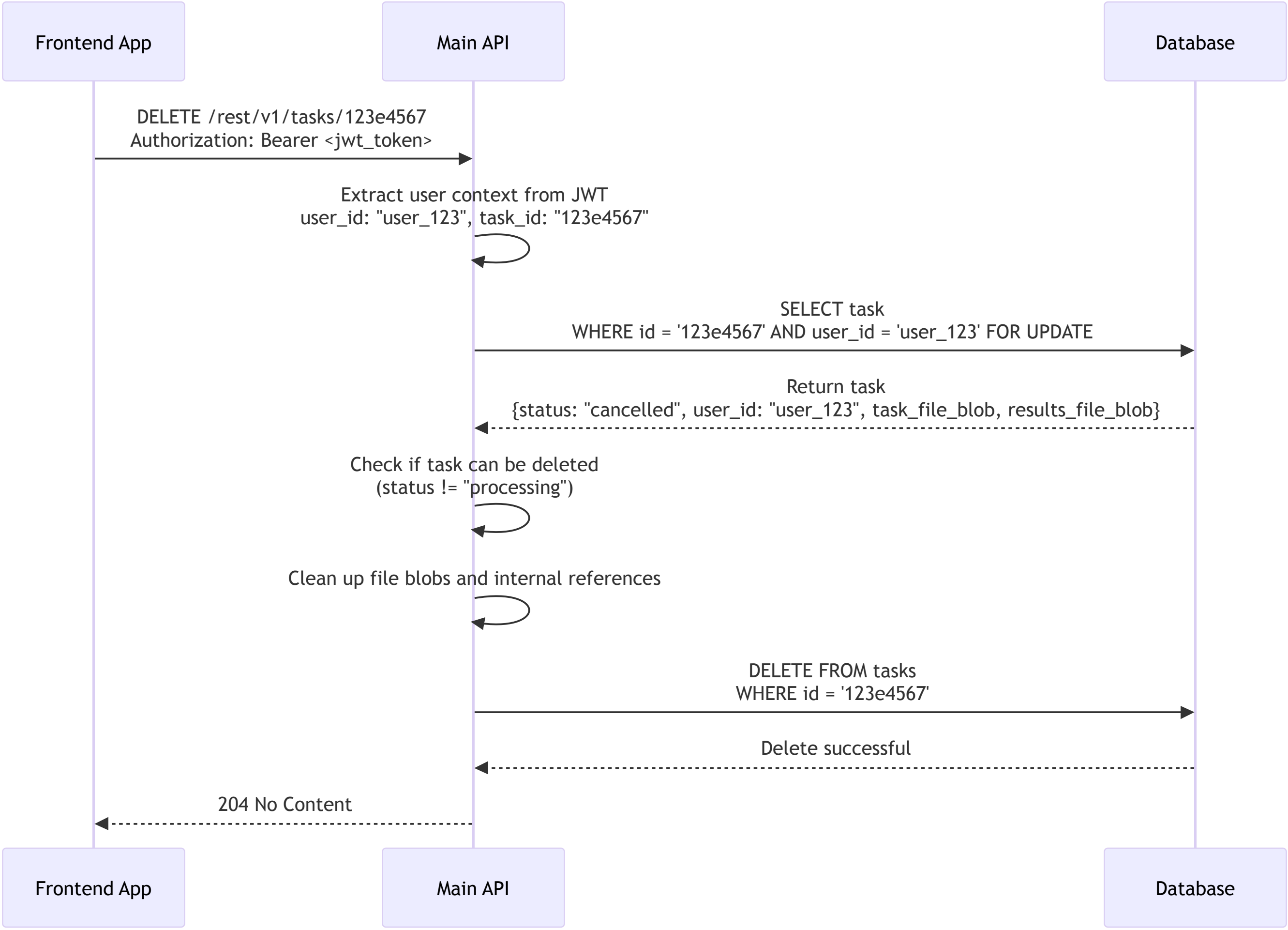

Frontend App
Main API
Database
PUT /rest/v1/tasks/123e4567
Authorization: Bearer <jwt_token>
Content-Type: application/json
{"action": "cancel"}
Extract user context from JWT
user_id: "user_123", task_id: "123e4567"
Parse request body and validate action
SELECT task
WHERE id = '123e4567' AND user_id = 'user_123' FOR UPDATE
Return task
{id, status: "processing", user_id: "user_123"}
Check if task can be cancelled
(status = "pending" or "processing")
Stop internal task processing
Clean up background operations
UPDATE task
SET status = 'cancelled', cancelled_at = NOW(), updated_at = NOW()
WHERE id = '123e4567'
Update successful
SELECT updated task (without blobs for response)
Return updated task with status "cancelled"
200 OK
{updated task with status "cancelled"}
Frontend App
Main API
Database

Error Flow - Cannot Cancel Completed Task

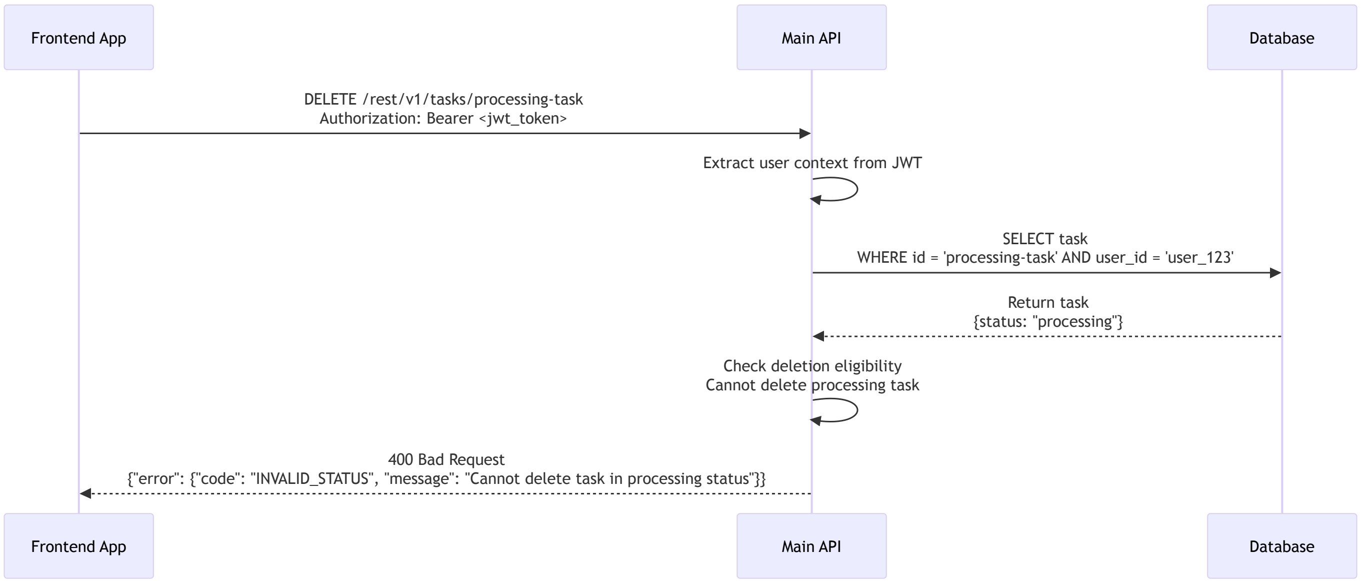

Frontend App
Main API
Database
PUT /rest/v1/tasks/completed-task
Authorization: Bearer <jwt_token>
{"action": "cancel"}
Extract user context from JWT
SELECT task
WHERE id = 'completed-task' AND user_id = 'user_123'
Return task
{status: "completed"}
Check cancellation eligibility
Cannot cancel completed task
400 Bad Request
{"error": {"code": "INVALID_STATUS", "message": "Cannot cancel completed task"}}
Frontend App
Main API
Database

5. DELETE /rest/v1/tasks/{id} - Delete Task

Success Flow


Frontend App
Main API
Database
DELETE /rest/v1/tasks/123e4567
Authorization: Bearer <jwt_token>
Extract user context from JWT
user_id: "user_123", task_id: "123e4567"
SELECT task
WHERE id = '123e4567' AND user_id = 'user_123' FOR UPDATE
Return task
{status: "cancelled", user_id: "user_123", task_file_blob, results_file_blob}
Check if task can be deleted
(status != "processing")
Clean up file blobs and internal references
DELETE FROM tasks
WHERE id = '123e4567'
Delete successful
204 No Content
Frontend App
Main API
Database

Error Flow - Cannot Delete Processing Task


Frontend App
Main API
Database
DELETE /rest/v1/tasks/processing-task
Authorization: Bearer <jwt_token>
Extract user context from JWT
SELECT task
WHERE id = 'processing-task' AND user_id = 'user_123'
Return task
{status: "processing"}
Check deletion eligibility
Cannot delete processing task
400 Bad Request
{"error": {"code": "INVALID_STATUS", "message": "Cannot delete task in processing status"}}
Frontend App
Main API
Database

Request/Response Flow Summary

Key Features Demonstrated

- **RESTful Endpoints**: All endpoints follow `/rest/v1/` pattern with proper HTTP methods
- **JWT Authentication**: User context extraction and validation on every request
- **User Ownership**: Tasks are scoped to the authenticated user
- **Internal Processing**: All file parsing and task processing handled within the main API
- **Performance Optimization**: List endpoint excludes blob data, detail endpoint includes blobs
- **Error Handling**: Comprehensive error responses with proper HTTP status codes
- **Database Transactions**: Atomic operations for data consistency
- **File Management**: Blob storage and retrieval handled internally with separate fields for task and results

Performance Characteristics

- **Optimized Listing**: GET /tasks excludes blob data for fast pagination
- **Complete Details**: GET /tasks/{id} includes all blob data for full task information
- **Simplified Architecture**: No external service dependencies
- **Direct Database Access**: Minimal latency for data operations
- **Internal File Processing**: No network overhead for file operations
- **Background Processing**: Non-blocking task execution with results storage
- **Efficient Querying**: Optimized database queries with proper indexing