📋 ToDo2 Proje Raporu

Tarih: 23.12.2025

Bu rapor, ToDo2 projesinde kullanılan tüm SQL yapıları, backend entegrasyonları ve frontend sayfalarını detaylı olarak açıklamaktadır.

🗄️ SQL Veritabanı Yapıları

1. STORED PROCEDURES (37 Adet)

Stored Procedure	Açıklama	Backend Kullanımı	Frontend Kullanımı
sp_RegisterUser	Yeni kullanıcı kaydı oluşturur (şifre hash'lenir)	UserController.cs → Register endpoint	AuthPage.jsx → Kayıt ol formu
sp_LoginUser	Kullanıcı girişi (email ile kullanıcı bilgilerini getirir)	UserController.cs → Login endpoint	AuthPage.jsx → Giriş yap formu
todo.sp_AddList	Yeni liste oluşturur	ListController.cs → AddList endpoint	TodoLayout.jsx → "Yeni liste" butonu (AddListModal)
todo.sp_GetLists	Kullanıcının tüm listelerini getirir	ListController.cs → GetLists endpoint	TodoLayout.jsx → Sidebar'da listeleri gösterir
todo.sp_UpdateList	Liste adını günceller	ListController.cs → UpdateList endpoint	TodoLayout.jsx → Liste düzenleme butonu
todo.sp_DeleteList	Liste siler (Trigger ile cascade delete çalışır)	ListController.cs → DeleteList endpoint	TodoLayout.jsx → Liste silme butonu
todo.sp_AddTask	Yeni görev oluşturur	TaskController.cs → AddTask endpoint	TodoPage.jsx → Görev ekleme formu
todo.sp_AddTaskWithPlan	Görev oluşturur ve otomatik olarak Daily/Weekly/Monthly planına ekler	TaskController.cs → AddTaskWithPlan endpoint	Takvim.jsx → Plan ekleme işlemleri
todo.sp_GetTasks	Kullanıcının tüm görevlerini getirir	TaskController.cs → GetTasks endpoint	Tüm sayfalarda görev listesi gösterimi
todo.sp_GetTasksByList	Belirli bir listeye ait görevleri getirir	TaskController.cs → GetTasksByList endpoint	ListPage.jsx → Liste detay sayfası
todo.sp_SearchTasks	Görev arama (TaskName ve TaskContent'te arama yapar)	TaskController.cs → SearchTasks endpoint	Gorevler.jsx → Arama sonuçları, TodoLayout.jsx → Arama kutusu

Stored Procedure	Açıklama	Backend Kullanımı	Frontend Kullanımı
todo.sp_UpdateTask	Görev bilgilerini günceller (Trigger'lar otomatik çalışır)	TaskController.cs → UpdateTask endpoint	TodoPage.jsx → Görev düzenleme paneli
todo.sp_DeleteTask	Görevi siler	TaskController.cs → DeleteTask endpoint	TodoPage.jsx → Görev silme butonu
todo.sp_AddStep	Göreve adım ekler	StepController.cs → AddStep endpoint	TodoPage.jsx → Görev detay panelinde adım ekleme
todo.sp_GetStepsByTask	Göreve ait tüm adımları getirir	StepController.cs → GetStepsByTask endpoint	TodoPage.jsx → Görev adımları listesi
todo.sp_UpdateStep	Adım bilgilerini günceller	StepController.cs → UpdateStep endpoint	TodoPage.jsx → Adım düzenleme (inline editing)
todo.sp_DeleteStep	Adımı siler	StepController.cs → DeleteStep endpoint	TodoPage.jsx → Adım silme butonu
todo.sp_RecalculateTaskCompletionFromSteps	Tüm adımlar tamamlandığında görevi otomatik tamamlar	StepController.cs → UpdateStep içinde çağrılabilir	TodoPage.jsx → Adım tamamlandığında otomatik çalışır
todo.sp_AddDailyTask	Günlük plana görev ekler	DailyTaskService.cs → AddDailyTask	Takvim.jsx → Günlük görev ekleme
todo.sp_GetDailyTasks	Kullanıcının günlük görevlerini getirir	DailyTaskController.cs → Get endpoint	Takvim.jsx → Günlük görevlerin gösterimi
todo.sp_RemoveDailyTask	Günlük plandan görevi kaldırır	DailyTaskService.cs → RemoveDailyTask	Takvim.jsx → Günlük görev kaldırma
todo.sp_AddWeeklyTask	Haftalık plana görev ekler	WeeklyTaskService.cs → AddWeeklyTask	Takvim.jsx → Haftalık görev ekleme
todo.sp_GetWeeklyTasks	Kullanıcının haftalık görevlerini getirir	WeeklyTaskController.cs → Get endpoint	Takvim.jsx → Haftalık görevlerin gösterimi

Stored Procedure	Açıklama	Backend Kullanımı	Frontend Kullanımı
todo.sp_RemoveWeeklyTask	Haftalık plandan görevi kaldırır	WeeklyTaskService.cs → RemoveWeeklyTask	Takvim.jsx → Haftalık görev kaldırma
todo.sp_AddMonthlyTask	Aylık plana görev ekler	MonthlyTaskService.cs → AddMonthlyTask	Takvim.jsx → Aylık görev ekleme
todo.sp_GetMonthlyTasks	Kullanıcının aylık görevlerini getirir	MonthlyTaskController.cs → Get endpoint	Takvim.jsx → Aylık görevlerin gösterimi
todo.sp_RemoveMonthlyTask	Aylık plandan görevi kaldırır	MonthlyTaskService.cs → RemoveMonthlyTask	Takvim.jsx → Aylık görev kaldırma
todo.sp_GenerateRecurringTasks	CURSOR kullanarak tekrarlayan görevler oluşturur (Daily/Weekly/Monthly)	Şu an backend'de kullanılmıyor (SQL Job olarak çalıştırılabilir)	-
todo.sp_RunNightlyRecurringPlans	Gece çalışacak job (sp_GenerateRecurringTasks'ı çağırır)	SQL Server Agent Job olarak çalıştırılmalı	-
todo.sp_AddNote	Not ekler (göreve bağlı veya bağımsız)	NotesController.cs → Add endpoint	TodoPage.jsx → Görev notları, NotDefteri.jsx → Not ekleme
todo.sp_GetNotesByUser	Kullanıcının tüm notlarını getirir	NotesRepository.cs → GetMyNotesAsync	Notlar.jsx → Tüm notlar listesi
todo.sp_GetNotesByTask	Göreve ait notları getirir	NotesRepository.cs → GetMyNotesByTaskAsync	TodoPage.jsx → Görev detay panelinde notlar
todo.sp_UpdateNote	Notu günceller	NotesController.cs → Update endpoint	Notlar.jsx, NotDefteri.jsx → Not düzenleme
todo.sp_DeleteNote	Notu siler	NotesController.cs → Delete endpoint	Notlar.jsx, NotDefteri.jsx → Not silme

```
// Örnek: todo.sp_AddTaskWithPlan Kullanımı (Backend) await _connection.ExecuteAsync( "todo.sp_AddTaskWithPlan", new { UserID = userId, ListID = dto.ListID, TaskName = dto.TaskName, TaskType = dto.TaskType.ToString() // "Daily", "Weekly", "Monthly" }, commandType: CommandType.StoredProcedure );
```

2. TRIGGERS (3 Adet)

Trigger	Tablo/İşlem	Açıklama	Ne Zaman Çalışır
trg_DeleteTasksWithList	todo.Lists (AFTER DELETE)	Liste silindiğinde, o listeye ait tüm görevleri ve plan kayıtlarını (DailyTasks, WeeklyTasks, MonthlyTasks) cascade delete ile siler	ListController.cs → DeleteList → sp_DeleteList çalıştığında
trg_RemoveCompletedTaskFromPlans	todo.Tasks (AFTER UPDATE)	Görev tamamlandığında (IsCompleted: 0→1), görevi DailyTasks, WeeklyTasks, MonthlyTasks tablolarından otomatik kaldırır	TaskController.cs → UpdateTask → sp_UpdateTask ile IsCompleted değiştiğinde
trg_TaskCompletedDate	todo.Tasks (AFTER UPDATE)	Görev tamamlandığında CompletedAt alanını GETDATE() ile otomatik doldurur	TaskController.cs → UpdateTask → sp_UpdateTask ile IsCompleted değiştiğinde

```
-- Örnek: trg_TaskCompletedDate Trigger Kodu CREATE OR ALTER TRIGGER todo.trg_TaskCompletedDate ON todo.Tasks AFTER UPDATE AS BEGIN SET NOCOUNT ON; UPDATE T SET CompletedAt = GETDATE() FROM todo.Tasks T INNER JOIN inserted i ON T.TaskID = i.TaskID INNER JOIN deleted d ON d.TaskID = i.TaskID WHERE i.IsCompleted = 1 AND d.IsCompleted = 0 AND T.CompletedAt IS NULL; END;
```

3. VIEWS (1 Adet)

View	Açıklama	Backend Kullanımı	Frontend Kullanımı
vw_AllPlans	DailyTasks, WeeklyTasks ve MonthlyTasks tablolarını UNION ALL ile birleştirerek tüm planları tek bir görünümde sunar	PlanRepository.cs → GetPlansAsync, PlanService.cs → GetPlansAsync	Planlanan.jsx → Tüm planları gösterir

```
-- vw_AllPlans View Yapısı CREATE OR ALTER VIEW todo.vw_AllPlans AS SELECT t.TaskID, t.TaskName, 'Daily' AS PlanType, d.PlanDate AS PlanDate, d.DailyTaskID AS PlanID FROM todo.DailyTasks d INNER JOIN todo.Tasks t ON t.TaskID = d.TaskID UNION ALL SELECT t.TaskID, t.TaskName, 'Weekly' AS PlanType, w.WeekStartDate AS PlanDate, w.WeeklyTaskID AS PlanID FROM todo.WeeklyTasks w INNER JOIN todo.Tasks t ON t.TaskID = w.TaskID UNION ALL SELECT t.TaskID, t.TaskName, 'Monthly' AS PlanType, m.MonthStartDate AS PlanDate, m.MonthlyTaskID AS PlanID FROM todo.MonthlyTasks m INNER JOIN todo.Tasks t ON t.TaskID = m.TaskID;
```

4. CURSORS (1 Adet)

Cursor	Stored Procedure	Açıklama	Kullanım Durumu
RecurringCursor	todo.sp_GenerateRecurringTasks	Tamamlanmış tekrarlayan görevleri (daily/weekly/monthly) tarar ve bir sonraki dönem için yeni plan kayıtları oluşturur	Şu an backend'de direkt kullanılmıyor. SQL Server Agent Job ile gece çalıştırılabilir

```
-- Cursor Kullanımı Örneği DECLARE RecurringCursor CURSOR FOR SELECT TaskID, RecurrenceType, IsCompleted, CompletedAt FROM todo.Tasks WHERE RecurrenceType IS NOT NULL AND RecurrenceType <> 'none' AND IsCompleted = 1; OPEN RecurringCursor; FETCH NEXT FROM RecurringCursor INTO @TaskID, @RecurrenceType, @IsCompleted, @CompletedAt; WHILE @@FETCH_STATUS = 0 BEGIN -- İşlemler... FETCH NEXT FROM RecurringCursor INTO @TaskID, @RecurrenceType, @IsCompleted, @CompletedAt; END; CLOSE RecurringCursor; DEALLOCATE RecurringCursor;
```

5. INDEXES (4 Adet)

Index	Tablo/Sütun	Açıklama	Faydası
IX_Users_UserMail	Users.UserMail	Email sütununa index (unique zaten var, performans için ek index)	Login işlemlerinde email aramalarını hızlandırır
IX_Notes_UserID	todo.Notes.UserID	Kullanıcı ID'sine göre not aramalarını hızlandırır	Kullanıcının notlarını getirirken performans artışı
IX_Notes_TaskID	todo.Notes.TaskID	Görev ID'sine göre not aramalarını hızlandırır	Göreve ait notları getirirken performans artışı
Primary Keys	Tüm tablolarda (TaskID, ListID, UserID, vb.)	Varsayılan olarak tüm PK'lar index'tir	JOIN ve WHERE işlemlerinde performans sağlar

🎨 Frontend Sayfaları ve İşlevleri

1. AuthPage.jsx

Yol: `/login`

İşlev: Kullanıcı girişi ve kayıt işlemleri
Kullanılan API'ler:

- `POST /api/User/register` → sp_RegisterUser
- `POST /api/User/login` → sp_LoginUser

Özellikler:

- Giriş yap ve kayıt ol modları arasında geçiş
- JWT token ile authentication
- Modern gradient tasarım
- Mobil responsive

2. Gunum.jsx (Günüm)

Yol: `/gunum`

Component: TodoPage (pageType="gunum")
İşlev: Bugüne ait görevleri gösterir (DueDate = bugün)
Kullanılan API'ler:

- `GET /api/Tasks/list` → sp_GetTasks (frontend'de filtreleme)
- `GET /api/daily-tasks/{userId}` → sp_GetDailyTasks

Özellikler:

- Bugünün görevlerini listeler
- Görev eklerken ListID = null olur (sadece günüm'e eklenir)
- Görev adımları inline gösterilir

3. Onemli.jsx (Önemli)

Yol: `/onemli`

Component: TodoPage (pageType="onemli")
İşlev: IsImportant = true olan görevleri gösterir
Kullanılan API'ler:

- `GET /api/Tasks/list` → sp_GetTasks (frontend'de IsImportant filtresi)

Özellikler:

- Yıldız işaretli görevler
- Hızlı erişim için önemli görevler

4. Planlanan.jsx

Yol: `/planlanan`

Component: TodoPage (pageType="planlanan")
İşlev: DueDate'i olan (gelecekteki) görevleri gösterir
Kullanılan API'ler:

- `GET /api/Tasks/list` → sp_GetTasks (frontend'de DueDate filtresi)
- `GET /api/plans/{userId}` → vw_AllPlans view

Özellikler:

- Planlı görevleri listeler
- Tarih bazlı filtreleme

5. Gorevler.jsx (Tüm Görevler)

Yol: `/gorevler`

Component: TodoPage
İşlev: Arama sonuçlarını ve tüm görevleri gösterir
Kullanılan API'ler:

- `GET /api/Tasks/search/{keyword}` → sp_SearchTasks
- `GET /api/Tasks/list` → sp_GetTasks

Özellikler:

- Arama kutusu ile görev arama
- TaskName ve TaskContent'te arama yapar
- SearchContext ile global arama state'i

6. ListPage.jsx

Yol: `/lists/:listId`

Component: TodoPage (listId prop ile)
İşlev: Belirli bir listeye ait görevleri gösterir
Kullanılan API'ler:

- `GET /api/Tasks/list/{listId}` → sp_GetTasksByList

Özellikler:

- Dinamik liste sayfası
- Liste adı header'da gösterilir
- Sadece o listeye ait görevler görünür

7. Takvim.jsx

Yol: `/takvim`

İşlev: Aylık ve haftalık takvim görünümü, günlük/haftalık/aylık planlar
Kullanılan API'ler:

- `GET /api/daily-tasks/{userId}` → sp_GetDailyTasks
- `GET /api/weekly-tasks/{userId}` → sp_GetWeeklyTasks
- `GET /api/monthly-tasks/{userId}` → sp_GetMonthlyTasks
- `POST /api/Notes/add` → sp_AddNote (günlere not ekleme)

Özellikler:

- Aylık ve haftalık görünüm (Pazartesi haftanın ilk günü)
- Günlük/haftalık/aylık görevleri gösterir
- Günlere not ekleme
- Responsive tasarım

8. Notlar.jsx

Yol: `/notlar`

İşlev: Görevlere bağlı notların listesi
Kullanılan API'ler:

- `GET /api/Notes/me` → sp_GetNotesByUser
- `PUT /api/Notes/update` → sp_UpdateNote
- `DELETE /api/Notes/delete/{noteId}` → sp_DeleteNote

Özellikler:

- Grid layout ile not kartları
- Inline düzenleme
- Görev ID'si ile ilişkilendirme gösterimi

9. NotDefteri.jsx

Yol: `/not-defteri`

İşlev: Bağımsız notlar (TaskID = null)
Kullanılan API'ler:

- `GET /api/Notes/me` → sp_GetNotesByUser (frontend'de TaskID = null filtresi)
- `POST /api/Notes/add` → sp_AddNote (TaskID: null)
- `PUT /api/Notes/update` → sp_UpdateNote
- `DELETE /api/Notes/delete/{noteId}` → sp_DeleteNote

Özellikler:

- Görevlere bağlı olmayan notlar
- Yeni not ekleme formu

- Not düzenleme ve silme

10. Tamamlananlar.jsx

Yol: `/tamamlananlar`
İşlev: Tamamlanmış görevlerin listesi
Kullanılan API'ler:

- `GET /api/Tasks/list` → sp_GetTasks (frontend'de IsCompleted = true filtresi)
- `PUT /api/Tasks/update` → sp_UpdateTask (IsCompleted: false yapmak için)
- `DELETE /api/Tasks/delete/{taskId}` → sp_DeleteTask

Özellikler:

- Tamamlanmış görevleri gösterir
- Tamamlanmamış olarak işaretleme
- Görev silme
- CompletedAt tarihi gösterimi

11. Ayarlar.jsx

Yol: `/ayarlar`
İşlev: Kullanıcı ayarları ve profil yönetimi
Kullanılan API'ler:

- Şu an backend endpoint'leri henüz eklenmemiş (TODO)

Özellikler:

- Profil bilgileri görüntüleme
- Çıkış yap butonu
- Hesap silme (şu an sadece UI, backend entegrasyonu yok)

12. TodoLayout.jsx

İşlev: Ana layout component (sidebar + içerik alanı)
Kullanılan API'ler:

- `GET /api/Lists/list` → sp_GetLists
- `POST /api/Lists/add` → sp_AddList
- `PUT /api/Lists/update` → sp_UpdateList
- `DELETE /api/Lists/delete/{listId}` → sp_DeleteList

Özellikler:

- Sidebar navigasyon (Günüm, Önemli, Planlanan, vb.)
- Dinamik liste listesi
- Liste ekleme modal (AddListModal component)

- Arama kutusu (SearchContext ile global state)
- Mobil hamburger menü
- Kullanıcı profil bilgileri

13. TodoPage.jsx (Ana Component)

İşlev: Görev listesi, ekleme, düzenleme, adımlar, notlar
Kullanılan API'ler:

- `GET /api/Tasks/list` → sp_GetTasks
- `GET /api/Tasks/list/{listId}` → sp_GetTasksByList
- `POST /api/Tasks/add` → sp_AddTask
- `PUT /api/Tasks/update` → sp_UpdateTask
- `DELETE /api/Tasks/delete/{taskId}` → sp_DeleteTask
- `GET /api/Steps/task/{taskId}` → sp_GetStepsByTask
- `POST /api/Steps/add` → sp_AddStep
- `PUT /api/Steps/update` → sp_UpdateStep
- `DELETE /api/Steps/delete/{stepId}` → sp_DeleteStep
- `GET /api/Notes/me/task/{taskId}` → sp_GetNotesByTask
- `POST /api/Notes/add` → sp_AddNote

Özellikler:

- Görev listesi gösterimi
- Görev ekleme/düzenleme/silme
- Görev adımları (inline editing, expand/collapse)
- Görev notları
- Görev detay paneli (sağ tarafta açılan panel)
- Mobil responsive (detail panel full-screen overlay)
- Filtreleme (pageType'a göre: gunum, onemli, planlanan)

🔌 Backend API Endpoint Mapping

Endpoint	HTTP Method	Stored Procedure	Controller	Frontend Kullanımı
/api/User/register	POST	sp_RegisterUser	UserController	AuthPage.jsx
/api/User/login	POST	sp_LoginUser	UserController	AuthPage.jsx
/api/Lists/add	POST	todo.sp_AddList	ListController	TodoLayout.jsx → AddListModal
/api/Lists/list	GET	todo.sp_GetLists	ListController	TodoLayout.jsx → Sidebar
/api/Lists/update	PUT	todo.sp_UpdateList	ListController	TodoLayout.jsx → Liste düzenleme
/api/Lists/delete/{listId}	DELETE	todo.sp_DeleteList	ListController	TodoLayout.jsx → Liste silme
/api/Tasks/add	POST	todo.sp_AddTask	TaskController	TodoPage.jsx → Görev ekleme
/api/Tasks/add-with-plan	POST	todo.sp_AddTaskWithPlan	TaskController	Takvim.jsx (plan ekleme)
/api/Tasks/list	GET	todo.sp_GetTasks	TaskController	Tüm sayfalarda görev listesi
/api/Tasks/list/{listId}	GET	todo.sp_GetTasksByList	TaskController	ListPage.jsx
/api/Tasks/search/{keyword}	GET	todo.sp_SearchTasks	TaskController	Gorevler.jsx → Arama
/api/Tasks/update	PUT	todo.sp_UpdateTask	TaskController	TodoPage.jsx → Görev düzenleme
/api/Tasks/delete/{taskId}	DELETE	todo.sp_DeleteTask	TaskController	TodoPage.jsx → Görev silme
/api/Steps/task/{taskId}	GET	todo.sp_GetStepsByTask	StepController	TodoPage.jsx → Görev adımları
/api/Steps/add	POST	todo.sp_AddStep	StepController	TodoPage.jsx → Adım ekleme
/api/Steps/update	PUT	todo.sp_UpdateStep	StepController	TodoPage.jsx → Adım güncelleme
/api/Steps/delete/{stepId}	DELETE	todo.sp_DeleteStep	StepController	TodoPage.jsx → Adım silme
/api/daily-tasks/{userId}	GET	todo.sp_GetDailyTasks	DailyTaskController	Takvim.jsx → Günlük görevler

Endpoint	HTTP Method	Stored Procedure	Controller	Frontend Kullanımı
/api/weekly-tasks/{userId}	GET	todo.sp_GetWeeklyTasks	WeeklyTaskController	Takvim.jsx → Haftalık görevler
/api/monthly-tasks/{userId}	GET	todo.sp_GetMonthlyTasks	MonthlyTaskController	Takvim.jsx → Aylık görevler
/api/plans/{userId}	GET	vw_AllPlans (VIEW)	PlansController	Planlanan.jsx
/api/Notes/add	POST	todo.sp_AddNote	NotesController	TodoPage.jsx, NotDefteri.jsx, Takvim.jsx
/api/Notes/me	GET	todo.sp_GetNotesByUser	NotesController	Notlar.jsx, NotDefteri.jsx
/api/Notes/me/task/{taskId}	GET	todo.sp_GetNotesByTask	NotesController	TodoPage.jsx → Görev notları
/api/Notes/update	PUT	todo.sp_UpdateNote	NotesController	Notlar.jsx, NotDefteri.jsx
/api/Notes/delete/{noteId}	DELETE	todo.sp_DeleteNote	NotesController	Notlar.jsx, NotDefteri.jsx

⚠️ Önemli Notlar

1. Trigger Otomatik Çalışır

Backend'de trigger'ları explicit olarak çağırmaya gerek yoktur. SQL Server otomatik olarak çalıştırır:

- **trg_DeleteTasksWithList:** Liste silindiğinde otomatik çalışır
- **trg_RemoveCompletedTaskFromPlans:** Görev tamamlandığında otomatik çalışır
- **trg_TaskCompletedDate:** Görev tamamlandığında CompletedAt'i otomatik doldurur

2. Cursor Kullanımı

todo.sp_GenerateRecurringTasks stored procedure'ı CURSOR kullanır ve şu an backend'de direkt çağrılmamaktadır. SQL Server Agent Job olarak gece çalıştırılabilir.

3. Index'ler Otomatik Kullanılır

Index'ler SQL Server query optimizer tarafından otomatik olarak kullanılır. Backend'de explicit çağrı gerektirmezler.

4. View Kullanımı

vw_AllPlans view'ı, DailyTasks, WeeklyTasks ve MonthlyTasks tablolarını birleştirerek tüm planları tek bir sorguda getirmek için kullanılır. PlanRepository ve PlanService'de kullanılmaktadır.

📊 İstatistikler

Kategori	Sayı	Detay
Stored Procedures	37	Tüm CRUD işlemleri ve özel iş mantığı
Triggers	3	Cascade delete, otomatik tarih güncelleme
Views	1	Tüm planları birleştiren görünüm
Cursors	1	Tekrarlayan görevler için
Indexes	4	Performans optimizasyonu
Frontend Sayfaları	13	Auth, görev yönetimi, takvim, notlar, ayarlar
Backend Controllers	12	API endpoint'leri
API Endpoints	26+	RESTful API yapısı

ToDo2 Proje Raporu

Oluşturulma Tarihi: 23.12.2025