Attorney Change and Attorney Company Implementation Documentation

Current Architecture

Database Structure

1. Single 'Attorneys' table with a 'CompanyName' column
2. Direct relationship to 'Organizations' table
3. Company information stored as string values

Current Limitations

1. **Data Redundancy**

   - Company names stored repeatedly for each attorney

   - Potential for inconsistencies in company naming

   - No standardization of company information

2. **Limited Company Information**

   - No ability to store company contact details

   - No historical tracking of attorney-company relationships

   - No way to manage company-specific data

3. **Frontend Constraints**

   - Company selection based on string matching

   - No validation of company names (i.e. duplicates)

   - Difficult to maintain company data consistency/integrity

Proposed (Developed) Architecture

New Database Structure

1. **Attorneys Table**

   - Removed ‘CompanyName’ column

   - Maintains core attorney information

   - Keeps organization relationship

2. **New AttorneyCompany Table**

   - Dedicated storage for company information

   - Proper fields for contact details

   - Organization relationship maintained

3. **New AttorneyCompanyLinks Table**

   - Many-to-many relationship

   - Temporal tracking of relationships

   - Active status tracking

Service Layer Changes

1. **New AttorneyCompany Service**

   - Dedicated company management

   - Proper separation of concerns

   - Standardized CRUD operations

2. **Modified Attorney Service**

   - Removed company-specific operations

   - Cleaner, more focused responsibility

   - Improved maintainability

Frontend Improvements

1. **Enhanced Company Selection**
   - Dropdown populated from company table
   - Consistent company names
   - Add new company capability

2. **Better User Experience**
   - Clear separation of company and attorney details
   - Improved data validation
   - More intuitive data entry

Benefits of New Architecture

1. **Data Integrity**
   - Elimination of duplicate company names
   - Consistent company information
   - Proper relationship tracking
   - Reduced risk of data inconsistencies

2. **Flexibility**
   - Attorneys can be associated with multiple companies
   - Historical tracking of company relationships
   - Support for company transfers and changes
   - Future-proof for additional company-related features

3. **Maintainability**
   - Clear separation of concerns
   - Proper service organization
   - Easier to extend and modify

- Better code organization

4. **User Experience**
   - More reliable company selection
   - Improved data entry workflow
   - Better data validation
   - Clearer relationship management

5. **Performance**
   - Reduced data redundancy
   - More efficient queries
   - Better indexing possibilities
   - Improved data retrieval

6. **Business Value**
   - Better tracking of attorney-company relationships
   - Historical data for analysis
   - Improved reporting capabilities
   - Enhanced data quality for business decisions

Implementation Considerations

1. **Data Migration**
   - Create new tables
   - Migrate existing company names
   - Create relationship records
   - Validate data consistency

2. **Backend Changes**

   - New controller endpoints

   - Updated service layer

   - Modified data access

   - New validation rules

3. **Frontend Updates**

   - New company management modal

   - Updated forms and validation

   - Modified data binding

   - Enhanced user interface

4. **Testing Requirements**

   - Unit tests for new services

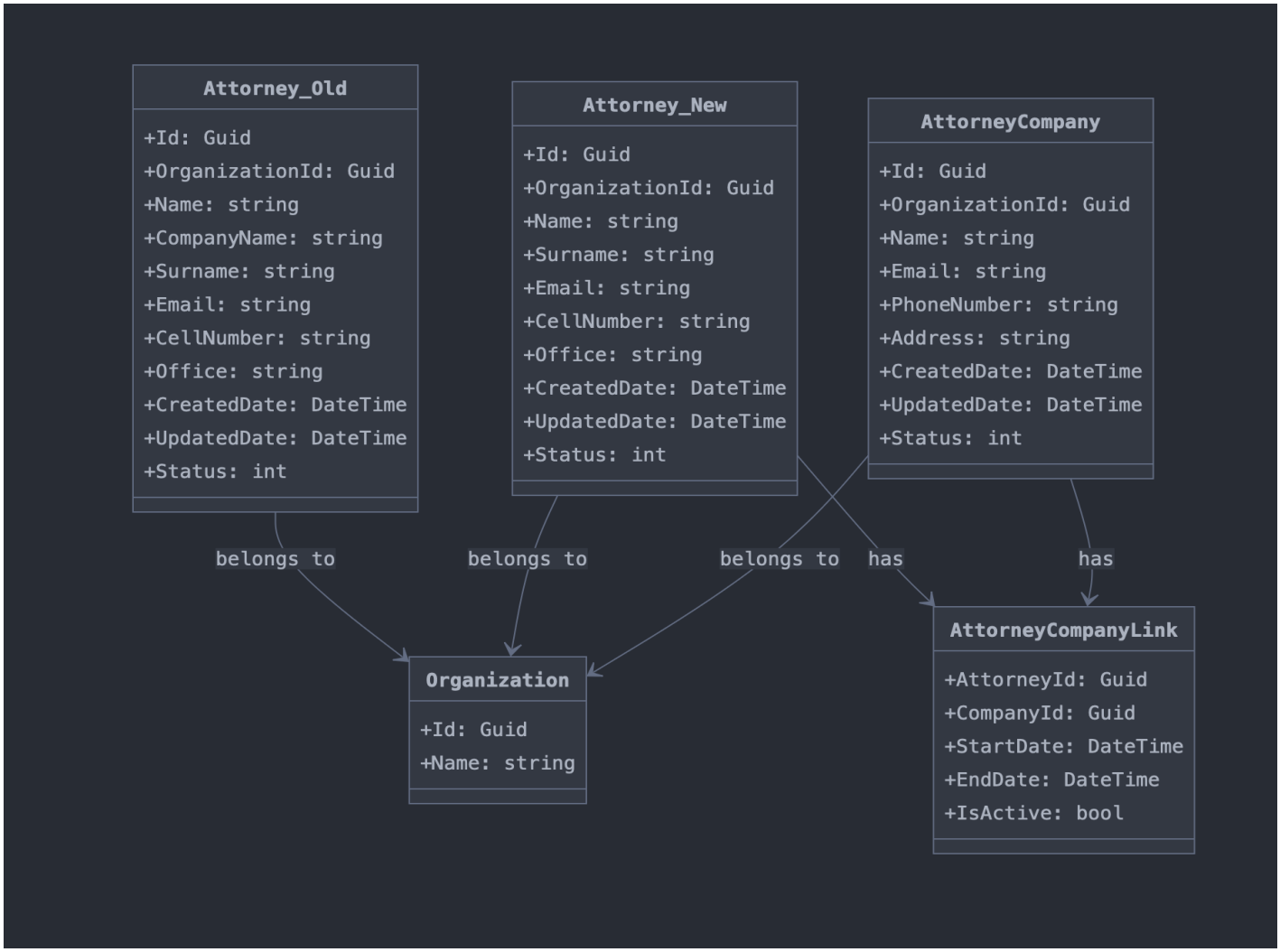
   - Integration tests for relationships

   - UI testing for new features

Total Working Hours

- Adjustment from 7 hours to 20
    - This includes removing of duplicate data, create new companies, linking current users (Attorneys) to correct company,

Architectural Change Diagram



Summary

The key architectural changes include:

1. Moving from a simple string-based company reference to a proper relational model
2. Adding temporal tracking of attorney-company relationships
3. Improving data integrity and reducing redundancy
4. Providing better separation of concerns in the service layer