# Emtelaak Development Approach and Tech Stack

## Development Philosophy

The Emtelaak platform will be built using modern software development practices with an emphasis on:

1. **Quality-First Development**
   * Test-driven development (TDD) where appropriate
   * Comprehensive automated testing (unit, integration, system)
   * Continuous code quality monitoring
2. **Agile Development Process**
   * Two-week Scrum sprints
   * Daily stand-ups
   * Sprint planning, reviews, and retrospectives
   * Continuous backlog refinement
3. **DevSecOps Culture**
   * Security integrated throughout the development lifecycle
   * Automated security scanning
   * Infrastructure as code
   * Continuous integration and delivery
4. **Documentation-Driven Development**
   * API-first approach with OpenAPI specifications
   * Comprehensive technical documentation
   * Knowledge sharing and collaboration

## Technology Stack

### Backend Technology

| Component | Technology | Rationale |
| --- | --- | --- |
| Primary Framework | ASP.NET Core 8 | Latest stable release with long-term support, performance improvements, enhanced security |
| API Architecture | RESTful APIs with OpenAPI | Industry standard, well-documented, client-agnostic |
| Authentication | IdentityServer4 | Open source, OAuth 2.0 and OpenID Connect implementation, mature library |
| ORM | Entity Framework Core | ORM with LINQ support, migrations, and rich querying capabilities |
| Messaging | MediatR + Azure Service Bus | CQRS pattern implementation, reliable messaging |
| Background Processing | Hangfire | Reliable background job processing, dashboard, recurring jobs |
| Caching | Redis | Distributed caching, high performance, pub/sub capabilities |
| Logging | Serilog + Application Insights | Structured logging, comprehensive monitoring |
| Testing | xUnit, Moq, FluentAssertions | Industry standard testing tools with good .NET support |

### Database Technology

| Component | Technology | Rationale |
| --- | --- | --- |
| Primary Database | SQL Server | ACID compliance, enterprise reliability, rich ecosystem |
| Document Storage | Azure Blob Storage | Scalable document storage, versioning, CDN integration |
| Caching | Redis | In-memory data structure store, high performance |
| Search | Elasticsearch | Full-text search capabilities, analytics |

### Mobile Technology

| Component | Technology | Rationale |
| --- | --- | --- |
| Mobile App Framework | Flutter | Cross-platform development with a single codebase, high performance, rich UI capabilities |
| Programming Language | Dart | Modern language designed for client-optimized applications |
| State Management | Provider/Bloc/Riverpod | Reactive state management patterns for Flutter |
| Map Integration | Google Maps Flutter | Comprehensive mapping capabilities for property locations |
| Virtual Tours | Panorama/360° viewers for Flutter | Immersive property viewing experience |
| Mobile Backend | Azure App Service + API Management | Scalable, secure mobile backend |
| Push Notifications | Firebase Cloud Messaging | Cross-platform push notification service |
| Analytics | Firebase Analytics | User analytics, crash reporting, and usage statistics |
| Offline Support | Hive/SQLite | Local data persistence for offline functionality |

### DevOps & Infrastructure

| Component | Technology | Rationale |
| --- | --- | --- |
| Source Control | Git + Azure DevOps Repos | Distributed version control, branching strategies |
| CI/CD | Azure DevOps Pipelines | Automated builds, testing, and deployment |
| Infrastructure as Code | Terraform + Azure Resource Manager | Reproducible infrastructure, version controlled |
| Monitoring | Application Insights + Azure Monitor | Comprehensive application and infrastructure monitoring |
| Containerization | Docker + Azure Container Registry | Consistent environments, simplified deployment |
| Orchestration | Azure Kubernetes Service (if needed) | Scalable container orchestration for complex deployments |

## Development Standards

### Coding Standards

1. **.NET Coding Conventions**
   * Follow Microsoft’s .NET coding conventions
   * Use latest C# language features appropriately
   * Enforce with StyleCop and .editorconfig
2. **API Design**
   * RESTful resource naming
   * Consistent HTTP verb usage
   * Standardized error responses
   * Comprehensive API documentation
3. **Security Practices**
   * OWASP Top 10 awareness and prevention
   * Regular security code reviews
   * Dependency vulnerability scanning
   * Sensitive data handling protocols

### Testing Strategy

1. **Unit Testing**
   * Business logic and service classes
   * Repository and data access layer
   * Target 80%+ code coverage for core components
2. **Integration Testing**
   * API endpoints
   * Database interactions
   * External service integrations
3. **System Testing**
   * End-to-end workflows
   * Performance and load testing
   * Security testing
4. **Mobile Testing**
   * Cross-device compatibility
   * Offline/spotty connection behavior
   * UI/UX testing across screen sizes

### Release Management

1. **Environment Strategy**
   * Development environment (per developer)
   * Integration environment (continuous integration)
   * Testing environment (QA and UAT)
   * Staging environment (pre-production)
   * Production environment
2. **Branching Strategy**
   * Feature branches for development
   * Pull request workflow with code reviews
   * Main/master branch protection
   * Release branches for deployed versions
3. **Versioning**
   * Semantic versioning (MAJOR.MINOR.PATCH)
   * Automated version incrementing
   * Release notes generation

## Quality Assurance Approach

1. **Automated Quality Gates**
   * Static code analysis
   * Unit test coverage
   * Security scanning
   * Performance benchmarks
2. **Manual QA Process**
   * User story acceptance testing
   * Exploratory testing
   * Usability testing
   * Regression testing
3. **Non-Functional Testing**
   * Load and stress testing
   * Security penetration testing
   * Accessibility testing
   * Disaster recovery testing

## Documentation

1. **Technical Documentation**
   * Architecture documentation
   * API documentation (Swagger/OpenAPI)
   * Database schema documentation
   * Integration documentation
2. **Process Documentation**
   * Development workflow
   * Release procedures
   * Incident response playbooks
   * Support and maintenance guides
3. **End-User Documentation**
   * User manuals
   * Administrator guides
   * API consumer documentation
   * Mobile app guides