Technical Design Document

Lifestyle Concierge Platform

Version 1.0

Table of Contents

[1. Introduction 3](#_Toc469906770)

[1.1. Context 3](#_Toc469906771)

[1.2. Purpose of This Document 3](#_Toc469906772)

[1.3. Definitions, Acronyms and Abbreviations 3](#_Toc469906773)

[2. Lifestyle Concierge Platform Architecture and Design 4](#_Toc469906774)

[2.1. Functional Overview 4](#_Toc469906775)

[2.1.1. Main System 4](#_Toc469906776)

[2.2. Software Architecture 4](#_Toc469906777)

[2.2.1. System Architecture 4](#_Toc469906778)

[2.2.2. System Design 5](#_Toc469906779)

[2.2.3. Infrastructure Layer 6](#_Toc469906780)

[2.2.4. Presentation Layer 6](#_Toc469906781)

[2.2.5. Distributed Services Layer 6](#_Toc469906782)

[2.2.6. Application Layer 6](#_Toc469906783)

[2.2.7. Domain Model Layer 6](#_Toc469906784)

[2.2.8. Data Persistence Layer 7](#_Toc469906785)

[2.3. External Interfaces 7](#_Toc469906786)

[3. Important Implementation Details 8](#_Toc469906787)

[3.1. Solution Structure and Project Naming Conventions 8](#_Toc469906788)

[3.2. Implementation of N-layered Architecture 8](#_Toc469906789)

[3.2.1. Mappings of Layer Components to Source Code 8](#_Toc469906790)

[3.2.2. Mapping of Lifestyle Concierge Platform’s Systems to Source Code 8](#_Toc469906791)

[3.3. The Infrastructure 8](#_Toc469906792)

[3.3.1. Configuration 8](#_Toc469906793)

[3.3.2. Security 8](#_Toc469906794)

[3.3.3. Mail Service Sendgrid SMTP 8](#_Toc469906795)

[3.3.4. Multiple Languages 8](#_Toc469906796)

[3.4. Lifestyle Concierge Platform System 8](#_Toc469906797)

[4. Deployment and Maintenance 9](#_Toc469906798)

[4.1. Deployment Overview 9](#_Toc469906799)

[4.2. System Maintenance 9](#_Toc469906800)

[5. Document Change Record 10](#_Toc469906801)

# Introduction

## Context

To create an in-house platform that will allow Aspire Lifestyles to publish and maintain multiple client websites to cater to client's members concierge and self-serve booking needs with fast turnaround time and short time to market for clients, lower maintenance/ongoing cost and be able to easily leverage past development for new clients and drive continuous improvement on quality of service for current clients.

## Purpose of This Document

## Definitions, Acronyms and Abbreviations

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# Lifestyle Concierge Platform Architecture and Design

## Functional Overview

### Main System

## Software Architecture

### System Architecture

Lifestyle Concierge Platform’s architecture is based on the N-layered architecture. The overview of the architecture is depicted in the diagram below.

Presentation Layer

Distributed Services Layer

Application Layer

Domain Layer

Data Persistence/Access Layer

Infrastructure Layer

External Services

Layer dependency

Data flow

Figure 2 - Lifestyle Concierge Platform’s architecture overview

* **Presentation Layer** Displays the user interface and accepts user inputs. Data format validation is also performed on this layer.
* **Distributed Services Layer** Provides report generation and housekeep services.
* **Application Layer** Coordinates features provided by domain, data access, and infrastructure layers to perform application tasks.
* **Domain Layer** Represents business concepts and implements business rules.
* **Data Persistence Infrastructure Layer** Provides functionality to persist and access data. Data sources that are persisted/accessed by this layer are databases.
* **Infrastructure Layer** Provides utilities that are used by other layers.

### System Design

The following diagram depicts the architecture of Lifestyle Concierge Platform system in more details.

Database

Dependency

Indirect dependency

**Presentation**

**Application Server**

**Presentation Layer**

**Distributed Services**(ASP.NET Web API)

**Application Layer**

**Domain Model Layer**

**Data Persistence/Access Layer**

External Services

**Infrastructure Layer**

Security

Mail Service Sendgrid SMTP

…

Web Client

(Concierge Website, Concierge Admin)

MVC 5, HTML 5, CSS3, jQuery

Component Services

DTOs

Domain Entities

Domain Services

Repository Contracts

Repositories

Data Mappings

Persistence (Entity Framework)

Service Agents

Figure 3 - Lifestyle Concierge Platform's architecture in details

The following sections describe each layer, including design of sub-layers / components, design decisions, location of each component in the source code, and important notices.

### Infrastructure Layer

Utilities classes and methods that can be used by all other layers are placed in the infrastructure layer. This layer is implemented in project Aspire.Sitemanager.Infrastructure. This layer consists of the following facilities.

### Presentation Layer

Presentation layer is located inside a tier with the same name. This placement is due to the separation of client terminals from the application servers. There are two types of clients in this layer.

#### Web Client LCP Website

A user uses a web client LCP Website, typically a web browser, to access to the functionality of the system. ASP.NET MVC version 5.0 is used to implement this kind of user interface. In the source code, this user interface is implemented by the following projects.

#### Web Client LCP Admin

A user uses a web client LCP Website, typically a web browser, to access to the functionality of the system. ASP.NET MVC version 5.0 is used to implement this kind of user interface. In the source code, this user interface is implemented by the following projects.

### Distributed Services Layer

Distributed Services is a thin layer that exposes some functionality of the application server to Mobile App.

* Apps: IOS + Android

### Application Layer

Application layer coordinates features of lower layers to perform application tasks.

### Domain Model Layer

Domain model layer is the heart of the system. It consists of some components

* **Domain entities:** These objects are used to represent the concepts of the business domain. Domain entities host data that can be persisted in a data store for later use; they also contain domain logic of the business flows.
* **Domain services:** When business logic does not fit into a single domain entity, it is placed inside a domain service. Domain services are stateless objects that coordinate operations composed by domain entities.
* **Repository contracts:** A repository represents as in-memory collection of entities of the same type. A domain service can use a repository object to query data required to perform its logic. To enable better separation of concerns and loosely coupling, repository contracts are placed in the domain model layer, while the implementations are placed in the data persistence layer.
* **Finder contracts:** Entity Framework is used to implement repositories, and LINQ to Entities is used to perform queries. Sometimes, Entity Framework cannot translate a query written in LINQ syntax into an SQL query that can be executed efficiently by the database server. Finder objects are created to tackle this performance issue. A finder is an object that is used to query the data source efficiently by directly executing an SQL query against the database. Finders are intended to complement repositories for performance reason, they are not meant to replace repositories.

### Data Persistence Layer

Data persistence layer is responsible for querying and persisting data to the data stores. There are several components in this layer.

* Implementations of repositories.
* Implementation of finders.
* Data mapping classes that contain configuration of how to map domain model objects to physical database model.
* Service agents that encapsulate access to data sources external to Lifestyle Concierge Platform system.

## External Interfaces

* SOLR Indexing

# Important Implementation Details

This section gives details about the implementation of the Lifestyle Concierge Platform system. It is intended for developers who work with the source code of the solution.

## Solution Structure and Project Naming Conventions

This section describes projects in the solution of Lifestyle Concierge Platform system. It also documents the naming conventions that developers should adhere to when modifying the source code.

Figure Source Code here

## Implementation of N-layered Architecture

### Mappings of Layer Components to Source Code

### Mapping of Lifestyle Concierge Platform’s Systems to Source Code

## The Infrastructure

* Infrastructure components are meant to be used by all layers, as shown in Figure 3 - Lifestyle Concierge Platform's architecture in details. Therefore, *Aspire.Sitemanager.Infrastructure* assembly can be referenced by projects in other layers in the solution. As a result, this project should host only components/utilities that are common enough to be shared among all other layers.
* Unity Application Block (Unity for short) is used as the dependency injection container through the projects. For now, constructor injection is the only injection technique in use. For more information on Unity integration, please refer to *Bootstrapper* classes inside projects.
* Mapping between types is facilitated by using AutoMapper. Mapping registrations are declared in *MapperBootstrapper* classes, which are placed at the root of projects.

### Configuration

### Security

### Mail Service Sendgrid SMTP

### Multiple Languages

## Lifestyle Concierge Platform System

# Deployment and Maintenance

## Deployment Overview

* Load balancer

## System Maintenance

# Document Change Record

| Version | Date | Author | Comment |
| --- | --- | --- | --- |
| 1.0 |  | S3 Team | Initial document creation |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |