**Architecture Description** of

The Layered architecture for

Optimum Hair Finder

**Contents**

**1 Identifying information**

* 1. Identifying information
  2. Supplementary information
  3. Other information

1.3.1 Overview

1.3.2 Architecture evaluations

1.3.3 Rationale for key decisions

1. **Stakeholder and concerns**
   1. Stakeholders
   2. Concerns

2.3 Concern-Stakeholder traceability

1. **Viewpoints**
   1. **Functional**
   2. Overview
   3. Concerns and stakeholders
   4. Model kinds
   5. Functional structure model
   6. Operations on views
   7. Correspondence rules

3.10 Sources

4.1 **Information**

* 1. Overview
  2. Concerns and stakeholders
  3. Model kinds
     1. Static information structure model
     2. Information flow model

4.6 Operations on views

4.7 Correspondence rules

* 1. Sources

5.1 **Development**

* 1. Overview
  2. Concerns and stakeholders
  3. Model kinds
  4. Module structure model
  5. Operations on views
  6. Correspondence rules

5.10 Sources

6.1 **Operational**

6.2 Overview

* 1. Concerns and stakeholders
  2. Model kinds

6.5 Administration model

* 1. Operations on views

6.7 Correspondence rules

6.10 Sources

**4 Views**

4.1 <view name>

4.1.1 Models

4.1.2 <model name>

* + 1. Known Issues with view

**5 Consistency and correspondences**

5.1 Known inconsistencies

5.2 Correspondences in the Architecture Document

5.3 Correspondence rules

**A Architecture decisions and rationale**

A.1

**Introduction**

The introduction section of the Software Architecture and Design document provides the main architectural pattern of the system, its design concepts and implementation and supplementary information on the system. Additional information; like an overall view of the system and evaluations of the system, to support the main architectural model will also is included in this section of the document.

* 1. **Identifying information**

**The Layered Architecture**

The Layered or Tiered software architecture is organized in horizontal layers, each layer performing a specific role within the system. The layered architecture has the standard four layers in the **Optimum Hair Finder** software system. Each layer in the architecture has its own tasks and responsibilities which communicate with each other in an ordered approach. Each layer is an independent component that contributes the total functionality of the systems’ design. The following are the layers in the proposed system:

* + 1. **Presentation layer**

Handles the user interface and additional communications with users; in the proposed system this will be the web application.

* + 1. **Business layer**

Also called the domain logic layer, is responsible for the organisation of the system: executing specific user requests that are sent into the system.

* + 1. **Persistence layer**

Manages the interaction to the database; converts requests and puts them in proper form in order to send these requests to the database.

* + 1. **Database layer**

This layer deals with all stored data on database and accessibility for user requests through to the system.

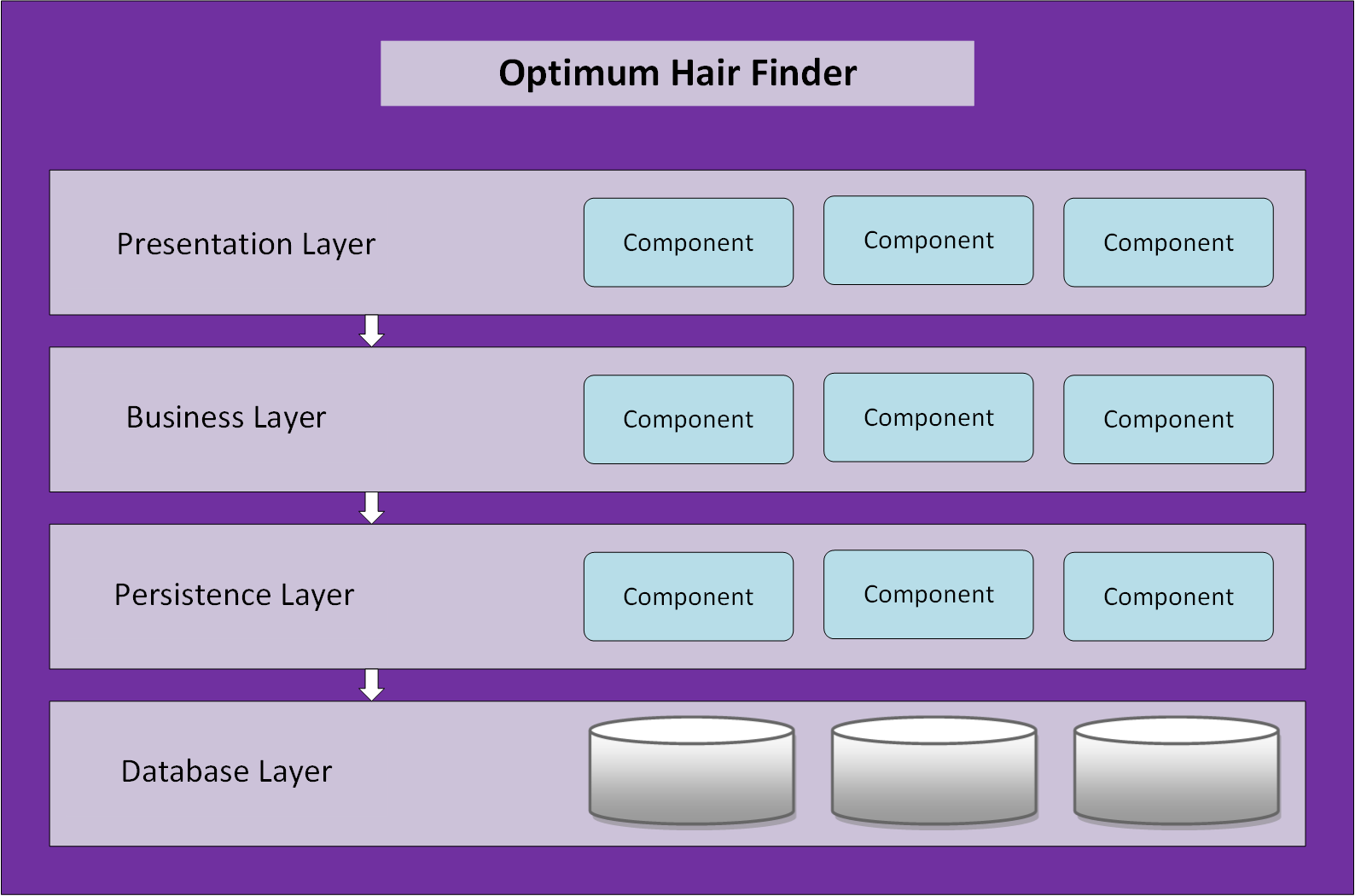


Figure 1: Layered Architecture Pattern

**1.2** **Supplementary information**

Authors: Nova Team

Raesetje Bonjo Sefala 844165

Laura Bokgoshi 792647

Siphamandla Mzobe 850456

Tshepo Molefe 705457

Keku Mashego 671000

Reviewers: Raesetje Bonjo Sefala, Laura Bokgoshi, Siphamandla Mzobe, Tshepo Molefe, Keku Mashego

Approving authority: Raesetje Bonjo Sefala

Issuing organization: Nova

Date of issue: First version 15 August, 2016

Status: Work in progress

Change history: First version 15 August, 2016: view salon, search for salon, search for hairstyle, search for prices, find proximity

Second version 12 September, 2016: view salon, search for salon, search for hairstyle, search for prices, find proximity, book appointment, view hairstyles

Version control information: <https://github.com/Makgoale-Bokgoshi/Optimum-Hair-Finder>

First version 15 August, 2016

Second version 12 September, 2016

Third version 26 September, 2016

Final version: October, 2016

Configuration management information: No updates yet.

Scope: Allows users to search hairstyle of choice on system and receive results of salons that offer that hairstyle, prices, location and distance to salons. System also allows user accounts, so users can login for a personal feel and view salon details and book appointments with salons.

Context: This document contains the software architecture description of the Optimum Hair Finder system created by Nova. The descriptions are detailed using architecture design information and UML interaction diagrams. This document is prepared using ISO/IEC/IEEE 42010, Systems and software engineering—The Architecture Description Template.

Summary: This is document is produced to read and provides the architectural design and diagrams to outline the Optimum Hair Finder software system.

Glossary:

UML:

SAD:

AD:

References:

* ISO/IEC/IEEE 42010, Systems and software engineering—The Architecture Description Template, Template prepared by: Rich Hilliard
* Software Architecture Patterns, Understanding Common Architecture Patterns and When to Use Them, Mark Richards
* Microservice Architectures, Dr. Andreas Schroeder
* Architectural Blueprints—The “4+1” View Model of Software Architecture, Philippe Kruchten
* Software Engineering, Ivan Marsic
  1. **Other information**

**1.3.1 Overview**

The remainder of this document will consist of: evaluation of this architectural model, the stakeholders and concerns of the architecture design, the viewpoints of this architectural design, views of the architecture of the system, consistency and correspondences in the architecture design and the architecture decisions and rationale.

1. **Architecture evaluations**

**Overall agility**

Rating: Low

Making changes to system takes time even with the layered system we have since system design and implementations is a big system.

**Ease of deployment**

Rating: Low

Changes made to the system affect its ability of it to be easily deployed especially the system is a big one. Change to one layer will not cause entire system to be updated only the section that deals with that change. So, all components related to the one that’s being changed may need to be re-evaluated and redeployed first before integration into system and redeployment of the entire system.

**Testability**

Rating: High

Each layer can be tested individually after each modification or before the deployment of the whole system.

**Performance**

Rating: Low

The interactivity in the use of layers causes time problems as some requests do not need passing through certain layers but still have to and so wasting time and therefore affecting the performance of the entire system.

**Scalability**

Rating: Low

The system will be hard to scale since it consists of multiple layers which are all linked somehow and follow certain pattern of interactivity. The only way to scale the system would be to split the layers up then scale each layer before integration again, which is not efficient.

**Ease of development**

Rating: High

The use of layers makes development of the software manageable because the different layers and components within will be familiar to developers.

**Complexity**

Rating: Low

The use of layers makes development of the software complex when it comes to the flow or organisation of the system. Certain requests are simple but because of the set structure of the design there is no way to pre-program certain methods for simpler request. Each step and verification will need to be checked in order for the system to run.

**Loose Coupling**

Rating: Low

The ordered approach of the interaction between layers causes problems when it comes to alteration of the software. If modifications are made to a layer, components within that layer are affected but not the other layers. Direct access to other layer is not allowed because of the many and methodical interdependencies between layers that cannot be randomly interrupted.

1. **Rationale for key decisions**

The system needed for Optimum Hair Finder was one that would allow specific **queries** access to a **database**. Depending on the type of query, certain extra steps would try to narrow down the request to the simplest form to be used to access the database. The system architecture design needed was one that would meet these requirements and the layered architecture came closest to meeting the requirements.

The layered architecture design offers a choice of a number of layers and a database layer. Since our system uses queries and queries are in a simple form with no room for misunderstandings, the use of a few layers would allows us to execute queries and produce results from the database. Our architecture design uses the four standard layers. The presentation, business logic, persistence and database layers together work well for our system.

The layered system in The Optimum Hair Finder software system is chosen because of its overall efficiency. The biggest feature in layered architectures is its **separation of concerns**. This feature allows components within a specific layer to interact with requests or instructions that concerns that layer only. This architectural model allows us to deal with the presentation, business logic, persistence and database of our system all individually. The organisation of these components of the system is more manageable and proficient with relation to the Optimum Hair Finder software system.

Another big feature in layered architectures is its **layer isolation** feature. The layers in a layered architecture are closed. This means that as a request moves from one layer to another layer, it must go through the layer below it to get to the next layer below that one. This feature allows the modification of details or implementation within a specific layer which will not affect the rest of the system, however, this continuous interactivity causes the change of such an architectural design to be very difficult and expensive.

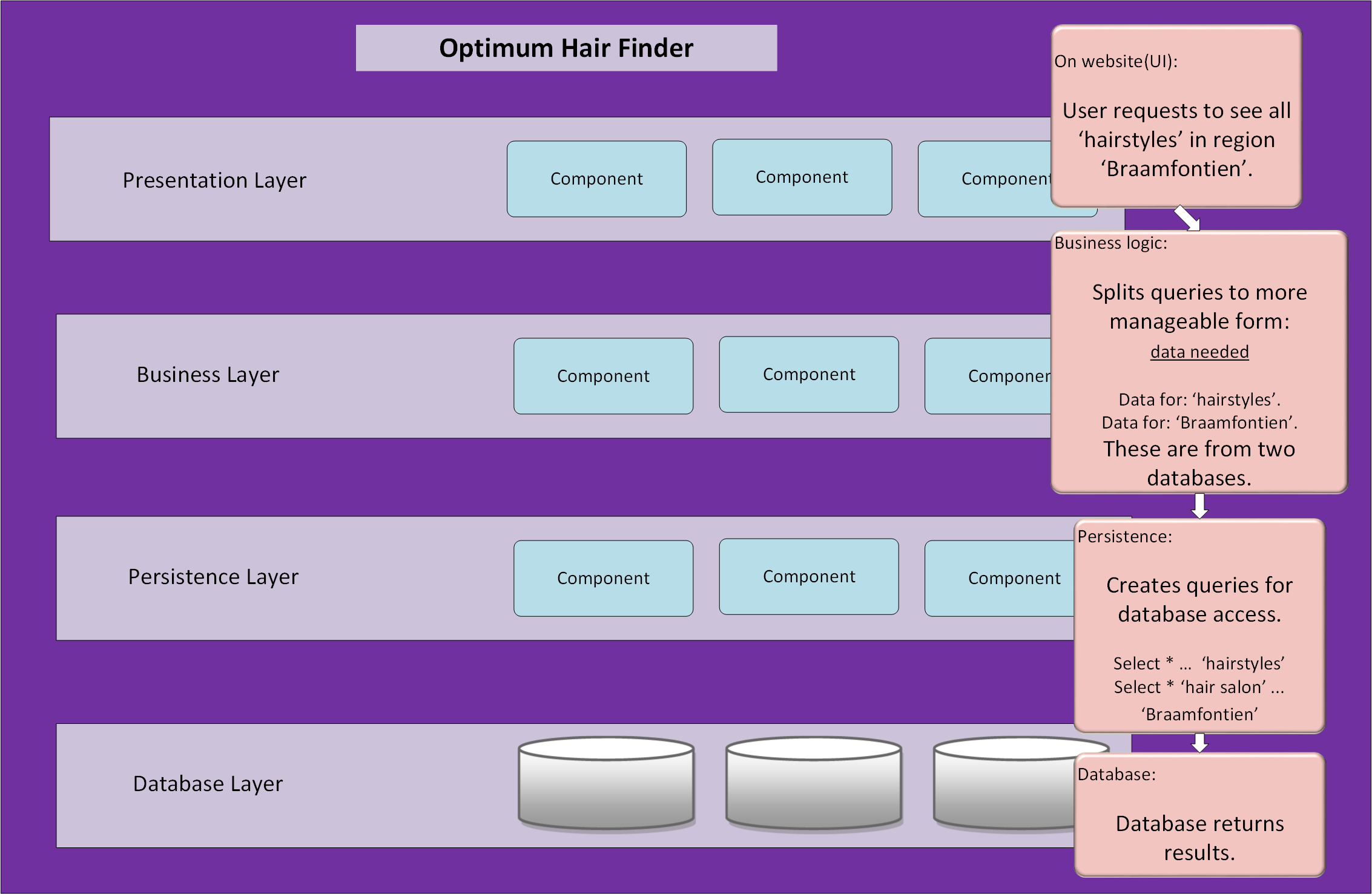


Figure 2: Example of use of Layered Architecture in system

**Stakeholders and concerns**

This chapter contains the key roles of the stakeholders of the architecture, the stakeholders’ concerns for that architecture, and the traceability of concerns to stakeholders.

**2.1** **Stakeholders**

The stakeholder for the architecture is the Nova Team, the users of our system and Dr. Terence van Zyl the lecturer.

Users: general public

Operators: Nova Team

Acquirers: None

Owners: Nova

Suppliers: None;

Developers: Nova Team

Builders: Nova Team

Maintainers: Nova Team

**2.2** **Concerns**

|  |  |
| --- | --- |
|  | Concerns |
| Concern 1 | **What are the purpose(s) of the Optimum Hair Finder?** |
| Concern 2 | **What is the suitability of the architecture for achieving the Optimum Hair Finder’s purpose(s)?** |
| Concern 3 | **How feasible is it to construct and deploy the Optimum Hair Finder?** |
| Concern 4 | **How is the Optimum Hair Finder to be maintained and evolved?** |
| Concern 5 | **How will the development of the Optimum Hair Finder be evaluated?** |
| Concern 6 | **How will Optimum Hair Finder impact its’ users?** |
| Concern 7 | **With relation to the architecture design pattern(Layered architecture); how will the sinkhole anti-pattern affect the overall performance of the system?** |
| Concern 8 | **How will Optimum Hair Finder handle the suggestion that Layered systems appear monolithic?** |
| Concern 9 | **Will Optimum Hair Finder be able to keep design pattern consistent?** |
| Concern 10 | **Are the layers of the system partitioned correctly?** |
| Concern 11 | **Are the layers of the system defined explicitly and are the relations between them logical?** |
| Concern 12 | **Are the system’s components functional, logical and do they relate?** |

Table 1: Table of concerns

**2.3** **Concern–Stakeholder Traceability**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Nova Team | Users | Dr. Terrence |
| Concern 1 | - | X | X |
| Concern 2 | X | X | X |
| Concern 3 | X | - | - |
| Concern 4 | X | - | - |
| Concern 5 | X | X | X |
| Concern 6 | X | X | X |
| Concern 7 | X | - | - |
| Concern 8 | X | - | - |
| Concern 9 | X | - | - |
| Concern 10 | X | - | - |
| Concern 11 | X | - | - |
| Concern 12 | X | - | - |

Table 2: Table of association of stakeholders to concerns in an Architecture Design

**Viewpoints**

**Functional**

**Information**

**Development**

**Operational**

**3.1 Functional**

**3.2 Overview**

The functional viewpoint describes the system’s functional elements; their responsibilities, interfaces, and interactions. This is the main viewpoint of the system since it is responsible for the systems’ functional aspect and will determine the systems’ scalability, security and performance.

Key features:

* indicates relationships
* indicates interactions
* indicates dependencies

**3.3 Concerns and stakeholders**

**3.3.1 Concerns**

|  |
| --- |
| Functional capabilities |
| identity of external entities and services and data used |
| nature and characteristics of external entities |
| identity and responsibilities of external interfaces |
| nature and characteristics of external interfaces |
| Functional design philosophy |
| overall completeness, consistency, and coherence |

**3.3.2 Typical stakeholders**

Developers: Nova Team

Maintainers: Nova Team

**3.4 Model kinds**

**Functional structure model**

**3.5 Functional structure model**

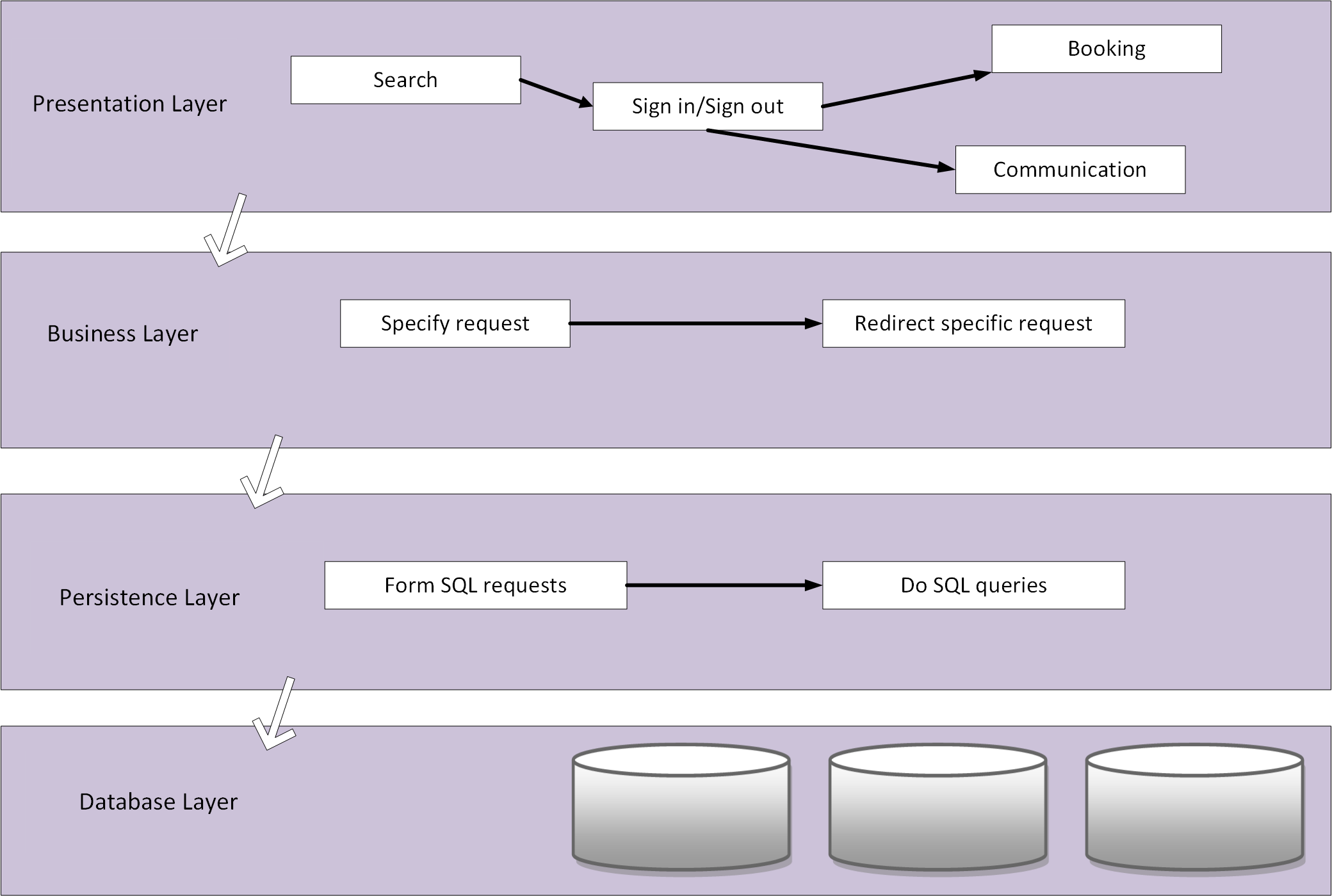


Figure 1: Layered Architecture Pattern

**3.5.1 Functional structure model conventions**

Languages: PHP, SQL, JavaScript

Notations:

Modeling techniques:

Analytical methods:

**3.5.3 Functional structure model correspondence rules**

? Document any correspondence rules associated with the model kind.

See x3.7 for further guidance.

**3.6 Operations on views**

**construction methods** are the means by which views are constructed under this

**interpretation methods**

**analysis methods**

**implementation methods**

**3.7 Correspondence rules**

? Document any correspondence rules defined by this viewpoint or its model kinds.

Usually, these rules will be across models or across views since, constraints within

a model kind will have been specified as part of the conventions of that model kind.

See: ISO/IEC/IEEE 42010, 4.2.6 and 5.7

**3.10 Sources**

[1] Nick Rozanski and Eoin Wood. Software Systems Architectures. Working with stakeholder using viewpoints and perspectives, 2nd edition, 2010.

**4.1 Information**

**4.2 Overview**

This viewpoint describes systems’ storage, manipulation, management, and distribution structures. i.e. the administrative aspect of the system.

Key features:

* indicates content and structure management
* indicates ownership, latency and references
* indicates data migration

**4.3 Concerns and stakeholders**

**4.3.1 Concerns**

|  |
| --- |
| information structure, content and usage |
| information ownership |
| enterprise-owned information |
| volatility of information semantics |
| information storage models |
| information flow, consistency and quality |
| timeliness, latency, and age |
| archiving and information retention |

**4.3.2 Typical stakeholders**

Users: general public

Operators: Nova Team

Acquirers: None

Owners: Nova

Suppliers: None;

Developers: Nova Team

Builders: Nova Team

Maintainers: Nova Team

**4.4 Model kinds**

**Static information structure model**

**Information flow model**

**4.5.1 Static information structure model**

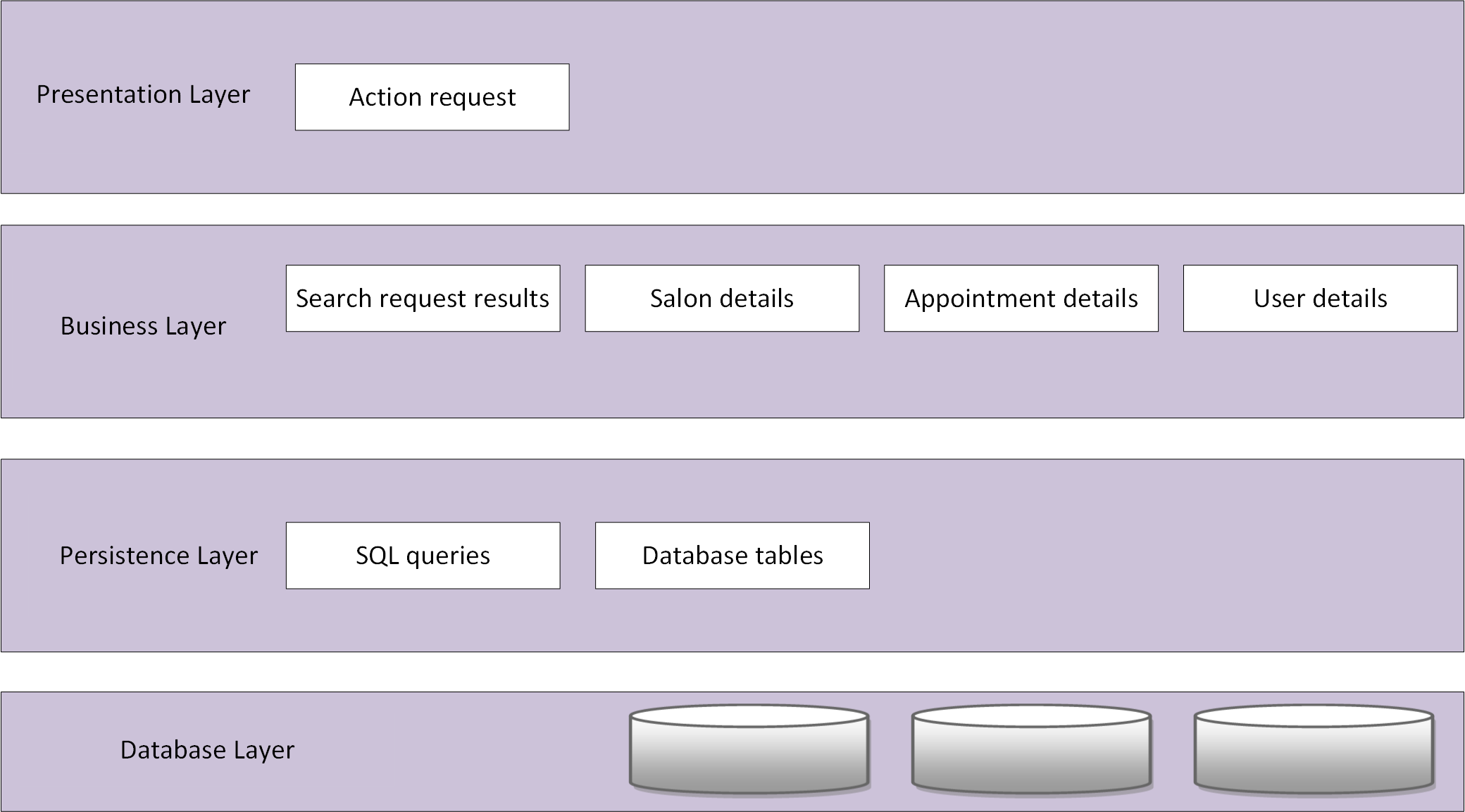


Figure 1: Layered Architecture Pattern

**4.5.1.1 Static information structure model conventions**

Languages: Layman terms

Notations:

Modeling techniques:

Analytical methods:

**4.5.1.3 Static information structure model correspondence rules**

? Document any correspondence rules associated with the model kind.

See x3.7 for further guidance.

**4.5.2 Information flow model**

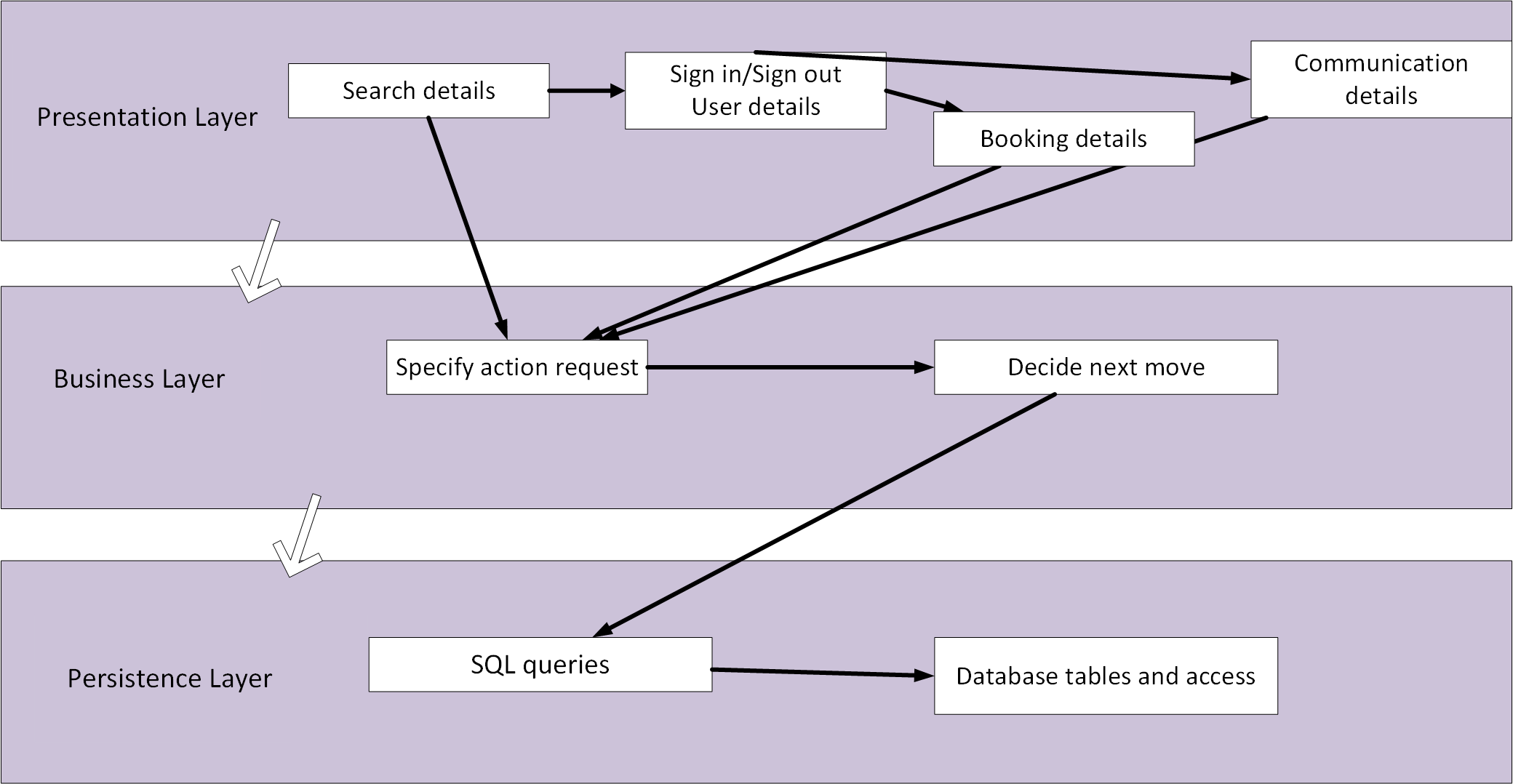


Figure 1: Layered Architecture Pattern

**4.5.2.1 Information flow model conventions**

Languages: Layman terms

Notations:

Modeling techniques:

Analytical methods:

**4.5.2.3 Information flow model correspondence rules**

? Document any correspondence rules associated with the model kind.

See x3.7 for further guidance.

**4.6 Operations on views**

**construction methods** are the means by which views are constructed under this

**interpretation methods**

**analysis methods**

**implementation methods**

**4.7 Correspondence rules**

? Document any correspondence rules defined by this viewpoint or its model kinds.

Usually, these rules will be across models or across views since, constraints within

a model kind will have been specified as part of the conventions of that model kind.

See: ISO/IEC/IEEE 42010, 4.2.6 and 5.7

**4.10 Sources**

[1] Nick Rozanski and Eoin Wood. Software Systems Architectures. Working with stakeholder using viewpoints and perspectives, 2nd edition, 2010.

**5.1 Development**

**5.2 Overview**

The development viewpoint describes the software development process; building the software, testing, maintaining, and enhancing the system.

Key features:

* indicates the code structure and dependencies
* indicates build and configuration management
* indicates system-wide design constraints

**5.3 Concerns and stakeholders**

**5.3.1 Concerns**

|  |
| --- |
| module organization |
| common processing |
| standardization of design |
| standardization of testing |
| instrumentation |
| codeline organization |

**5.3.2 Typical stakeholders**

Developers: Nova Team

Builders: Nova Team

Maintainers: Nova Team

**5.4 Model kinds**

**Module structure model**

**5.5 Module structure model**

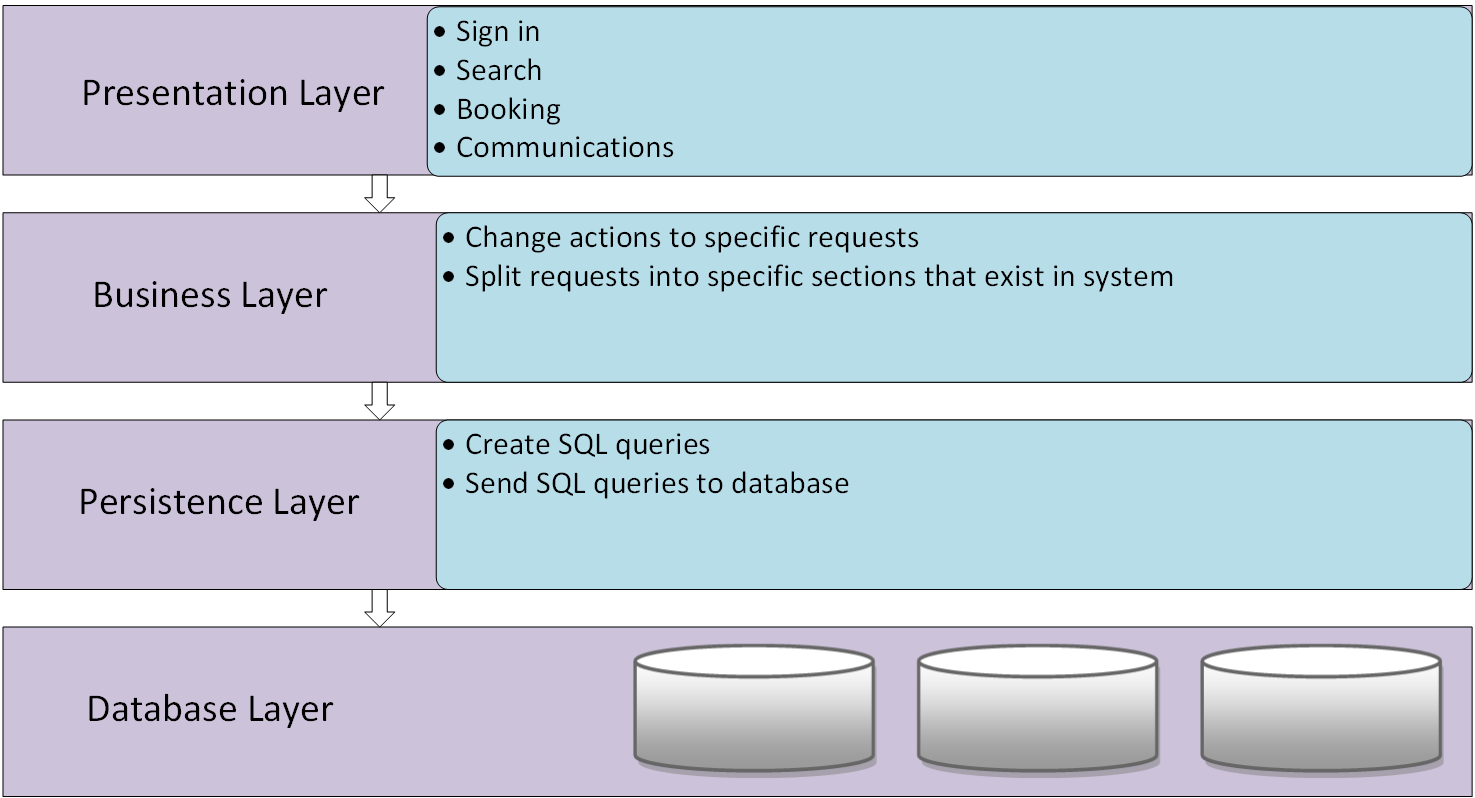


Figure 1: Layered Architecture Pattern

**5.5.1 Module structure model conventions**

Languages: Layman terms, SQL, PHP, JavaScript

Notations:

Modeling techniques:

Analytical methods:

**5.5.3 Module structure model correspondence rules**

? Document any correspondence rules associated with the model kind.

See x3.7 for further guidance.

**5.6 Operations on views**

**construction methods** are the means by which views are constructed under this

**interpretation methods**

**analysis methods**

**implementation methods**

**5.7 Correspondence rules**

? Document any correspondence rules defined by this viewpoint or its model kinds.

Usually, these rules will be across models or across views since, constraints within

a model kind will have been specified as part of the conventions of that model kind.

See: ISO/IEC/IEEE 42010, 4.2.6 and 5.7

**5.10 Sources**

[1] Nick Rozanski and Eoin Wood. Software Systems Architectures. Working with stakeholder using viewpoints and perspectives, 2nd edition, 2010.

**6.1 Operational**

**6.2 Overview**

The operational viewpoint describes how the system will be operated, administered, and supported and is significant. The operation of a system must be planned when the system is being designed, this helps identify system-wide strategies and solutions that might arise in early stages.

Key features:

* indicates installation requirements
* indicates management of system structures
* indicates operation the system

**6.3 Concerns and stakeholders**

**6.3.1 Concerns**

|  |
| --- |
| installation and upgrade |
| functional migration |
| data migration |
| operational monitoring and control |
| alerting |
| configuration management |
| performance monitoring |
| support |
| backup and restore |
| operation in third-party environments |

**6.3.2 Typical stakeholders**

Operators: Nova Team

Developers: Nova Team

Builders: Nova Team

Maintainers: Nova Team

**6.4 Model kinds**

**Administration model**

**6.5 Administration model**

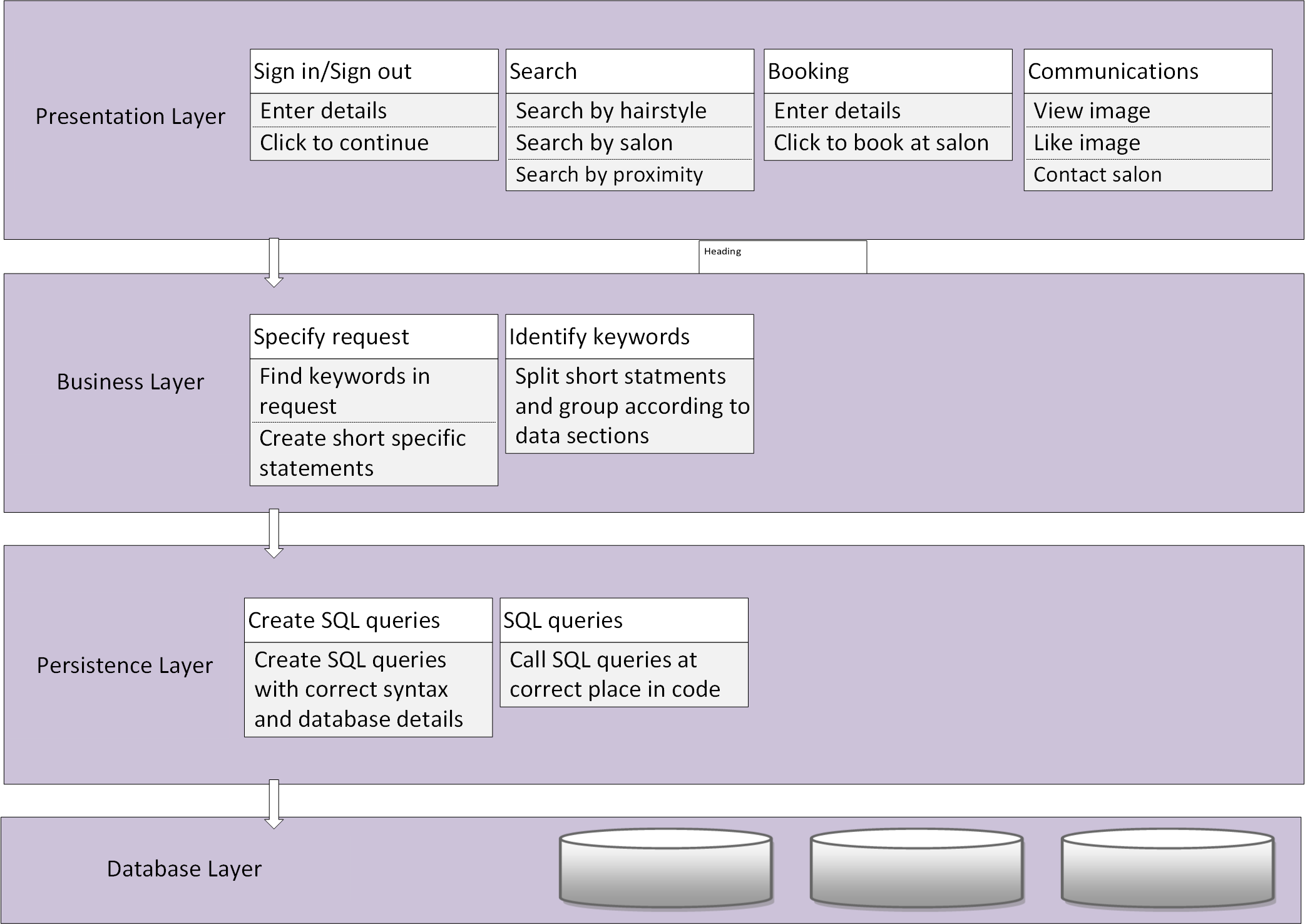


Figure 1: Layered Architecture Pattern

**6.5.1 Administration model conventions**

Languages: SQL, PHP, JavaScript

Notations:

Modeling techniques:

Analytical methods:

**6.5.3 Module structure model correspondence rules**

? Document any correspondence rules associated with the model kind.

See x3.7 for further guidance.

**6.6 Operations on views**

**construction methods** are the means by which views are constructed under this

**interpretation methods**

**analysis methods**

**implementation methods**

**6.7 Correspondence rules**

? Document any correspondence rules defined by this viewpoint or its model kinds.

Usually, these rules will be across models or across views since, constraints within

a model kind will have been specified as part of the conventions of that model kind.

See: ISO/IEC/IEEE 42010, 4.2.6 and 5.7

**6.10 Sources**

[1] Nick Rozanski and Eoin Wood. Software Systems Architectures. Working with stakeholder using viewpoints and perspectives, 2nd edition, 2010.

**Views**

**Logical**

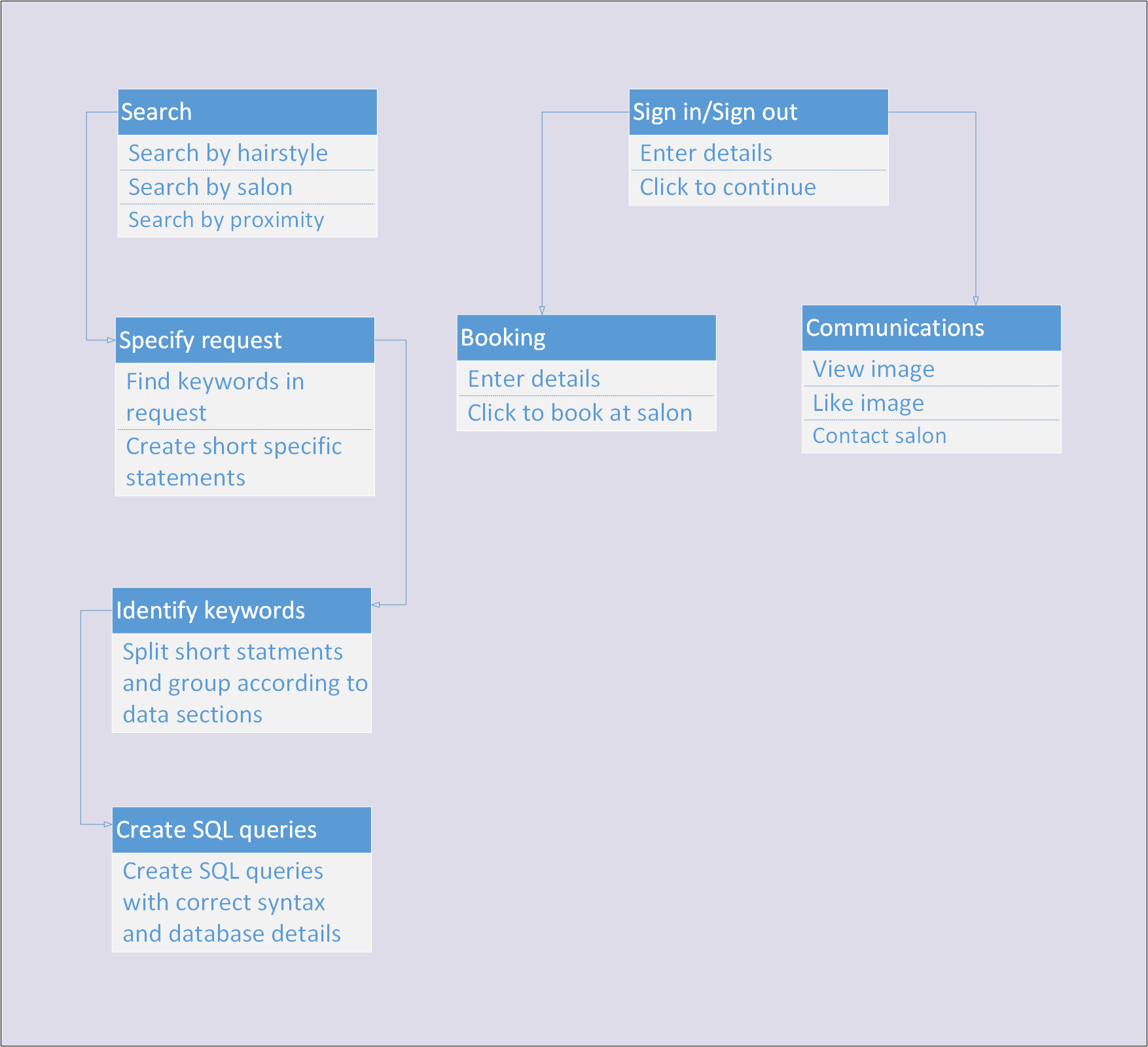
**Process**

**Development**

**Physical**

**4.1 View: Logical**

The logical view outlines the main services the system provides. It is based on the systems’ functional requirements, what the system should provide to the user. This view outlines this view in terms of the objects and classes that make up the system. This view also outlines the architecture of the system; what is the type of architecture used for this system, how many subsystems exist and how they are related. This outline is not only to show functionality that went into the system but to also show common mechanisms and design elements across the various parts of the system. The logical view is defined by the **functional viewpoint**.



**4.1.1 Models**

Functional structure model

**4.1.2 Functional structure model**

**4.1.3 Known Issues with View**

* 1. **View: Process**

The process view describes the main functionality of the system, the non-functional requirements of the software. This view outlines the tasks and distribution, systems’ integrity, fault-tolerance and how the functionality of the system relates to the logic. The process view also outlines the different levels that make up the entire structure of the system. The process view is defined by the operational viewpoint.

**4.2.1 Models**

Administration model

**4.2.2 Administration structure model**

**4.2.3 Known Issues with View**

**4.3 View: Development**

The development view deals with the software module organisation on the software development environment. The software is packaged in a way that allows groups of developers of work on certain parts of the system. This view is represented by module and subsystem diagrams showing all important relationships. This view takes into account internal requirements that allows for easy development, software management and reuse. The process view is defined by the development viewpoint.

**4.3.1 Models**

Module structure model

**4.3.2 Module structure model**

**4.3.3 Known Issues with View**

**4.4 Physical view**

The physical view takes into account the non-functional requirements of the system. The software executes on a network of physical computers that need to be mapped onto the various processing nodes. Usually we use physical parts for development and testing, and we try for efficient physical components so that they do not interfere with the actual code. The process view is defined by the information viewpoint.

**4.4.1 Models**

Static information structure model

Information flow model

**4.4.2 Static information structure model**

**4.4.3 Information flow model**

**4.4.4 Known Issues with View**

**Consistency and correspondences**

This chapter describes consistency requirements, recording of known inconsisten-cies in an AD, and the use and documentation of correspondences and correspon-dence rules.

**5.1** **Known inconsistencies**

? Record any known inconsistencies in the AD.

Although consistent ADs obviously are to be preferred, it is sometimes infeasible or impractical to resolve all inconsistencies for reasons of time, effort, or insufficient information.

2 An architecture description should include an analysis of consistency of its archi-tecture models and its views.

**5.2** **Correspondences in the AD**

? Identify each correspondence in the AD and its participating AD elements. Iden-tify any correspondence rules governing

Correspondences are used to express, record, enforce and analyze consistency be-tween models, views and other AD elements within an architecture description, between ADs, or between an AD and other forms of documentation.

AD elements include instances of stakeholders, concerns, viewpoints and views,

model kinds and models, decisions and rationales. Constructs introduced by view-points and model kinds are also AD elements.

Correspondences are n-ary mathematical relations. Correspondences can be de-picted via tables, via links, or via other forms of association (such as in UML).

**5.3** **Correspondence rules**

? Identify each correspondence rule applying to the AD.

Correspondence rules can be introduced by the AD, by one of its viewpoints, or from an architecture framework or architecture description language being used.

? For each identified correspondence rule, record whether the rule holds (is satisfied) or otherwise record all known violations.

17

**Appendix A**

It is not required by the Standard to capture architecture decisions. This section describes recommendations (“shoulds”) for their recording.

**A.1**

**Bibliography**

Clements, Paul C. et al. Documenting Software Architectures: views and beyond. 2nd. Addison Wesley, 2010.

Finkelstein, A. et al. “Viewpoints: a framework for integrating multiple per-spectives in system development”. In: International Journal of Software Engineering and Knowledge Engineering 2.1 (Mar. 1992), pp. 31–57.

Heesch, Uwe van, Paris Avgeriou, and Rich Hilliard. “A Documentation Framework for Architecture Decisions”. In: The Journal of Systems & Software 85.4 (Apr. 2012), pp. 795–820. DOI: [10.1016/j.jss.2011.](http://dx.doi.org/10.1016/j.jss.2011.10.017) [10.017](http://dx.doi.org/10.1016/j.jss.2011.10.017).

IEEE Std 1471, IEEE Recommended Practice for Architectural Description of Software-Intensive Systems. Oct. 2000.

ISO/IEC/IEEE 42010, Systems and software engineering — Architecture description. Dec. 2011, pp. 1–46.

Ran, Alexander. “ARES Conceptual Framework for Software Architecture”. In: Software Architecture for Product Families Principles and Practice. Ed. by M. Jazayeri, A. Ran, and F. van der Linden. Addison-Wesley, 2000, pp. 1–29.

Rozanski, Nick and Eoin´ Woods. Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives. 2nd. Addison Wes-ley, 2011.

20

**Contents**

|  |  |
| --- | --- |
| [**Using the template**](#page2) | **i** |
| [License](#page2) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | i |
| [Version History](#page4) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 1 |
| [Template editions](#page4) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 1 |
| [Comments](#page4) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| [**1 Introduction**](#page5) | | | **2** |
| [1.1](#page5) | [Identifying information](#page5) . . . . . . . . . . . . . . . . . . . . . . . . . . . | | 2 |
| [1.2](#page5) | [Supplementary information](#page5) . . . . . . . . . . . . . . . . . . . . . . . . | | 2 |
| [1.3](#page6) | [Other information](#page6) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | | 3 |
|  | [1.3.1](#page6) | [Overview (optional)](#page6) . . . . . . . . . . . . . . . . . . . . . . . . . | 3 |
|  | [1.3.2](#page7) | [Architecture evaluations](#page7) . . . . . . . . . . . . . . . . . . . . . . | 4 |
|  | [1.3.3 Rationale for key decisions](#page7) . . . . . . . . . . . . . . . . . . . . . | | 4 |

|  |  |  |
| --- | --- | --- |
| [**2 Stakeholders and concerns**](#page8) | | **5** |
| [2.1](#page8) | [Stakeholders](#page8) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 5 |
| [2.2](#page9) | [Concerns](#page9) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 6 |
| [2.3](#page9) | [Concern–Stakeholder Traceability](#page9) . . . . . . . . . . . . . . . . . . . . . | 6 |

|  |  |  |
| --- | --- | --- |
| [**3 Viewpoints+**](#page10) | | **7** |
| [3.1](#page11) | [<Viewpoint Name>](#page11) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 8 |
| [3.2](#page11) | [Overview](#page11) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 8 |

21

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | [3.3](#page11) | [Concerns and stakeholders](#page11) . . . . . . . . . . . . . . . . . . . . . . . . . | | 8 |
|  |  | [3.3.1](#page11) | [Concerns](#page11) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 8 |
|  |  | [3.3.2](#page12) | [Typical stakeholders](#page12) . . . . . . . . . . . . . . . . . . . . . . . . . | 9 |
|  |  | [3.3.3 “Anti-concerns” (optional)](#page13) . . . . . . . . . . . . . . . . . . . . . . | | 10 |
|  | [3.4](#page13) | [Model kinds+](#page13) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | | 10 |
|  | [3.5](#page13) | [<Model Kind Name>](#page13) . . . . . . . . . . . . . . . . . . . . . . . . . . . . | | 10 |
|  |  | [3.5.1 <Model Kind Name> conventions](#page13) . . . . . . . . . . . . . . . . . | | 10 |
|  |  |  | [I) Model kind languages or notations (optional)](#page14) . . . . . . . . . | 11 |
|  |  |  | [II) Model kind metamodel (optional)](#page14) . . . . . . . . . . . . . . . | 11 |
|  |  |  | [III) Model kind templates (optional)](#page15) . . . . . . . . . . . . . . . . | 12 |
|  |  | [3.5.2 <Model Kind Name> operations (optional)](#page15) . . . . . . . . . . . . | | 12 |
|  |  | [3.5.3 <Model Kind Name> correspondence rules](#page15) . . . . . . . . . . . | | 12 |
|  | [3.6](#page15) | [Operations on views](#page15) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | | 12 |
|  | [3.7](#page16) | [Correspondence rules](#page16) . . . . . . . . . . . . . . . . . . . . . . . . . . . . | | 13 |
|  | [3.8](#page16) | [Examples (optional)](#page16) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | | 13 |
|  | [3.9](#page16) | [Notes (optional)](#page16) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | | 13 |
|  | [3.10](#page16) | [Sources](#page16) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | | 13 |
| [**4**](#page17) | [**Views+**](#page17) | |  | **14** |
|  | [4.1](#page17) | [View: <View Name>](#page17) . . . . . . . . . . . . . . . . . . . . . . . . . . . . | | 14 |
|  |  | [4.1.1](#page17) | [Models+](#page17) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 14 |
|  |  | [4.1.2 <Model-Name>](#page18) . . . . . . . . . . . . . . . . . . . . . . . . . . . | | 15 |
|  |  | [4.1.3 Known Issues with View](#page18) . . . . . . . . . . . . . . . . . . . . . . | | 15 |
| [**5**](#page19) | [**Consistency and correspondences**](#page19) | | | **16** |
|  | [5.1](#page19) | [Known inconsistencies](#page19) . . . . . . . . . . . . . . . . . . . . . . . . . . . | | 16 |
|  | [5.2](#page19) | [Correspondences in the AD](#page19) . . . . . . . . . . . . . . . . . . . . . . . . . | | 16 |
|  | [5.3](#page20) | [Correspondence rules](#page20) . . . . . . . . . . . . . . . . . . . . . . . . . . . . | | 17 |
| [**A**](#page21) | [**Architecture decisions and rationale**](#page21) | | | **18** |

22

[A.1 Decisions](#page21) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 18

23