|  |
| --- |
| EPAM Systems |
| Booking Management System  Software Architecture Document |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| REVISION HISTORY | | | | | |
| Ver. | Description of Change | Author | Date | Approved | |
| Name | Effective Date |
| 0.1 | Initial version | Sergey Krasnov | 09.11.2017 |  |  |
|  |  |  |  |  |  |

|  |  |
| --- | --- |
| Related Artifacts | |
| Ref. | Name |
|  |  |
|  |  |

|  |  |
| --- | --- |
| Abbreviations and Acronyms | |
|  |  |
|  |  |

Contents

[1. Introduction 3](#_Toc498030431)

[1.1. Purpose 3](#_Toc498030432)

[1.2. Scope 3](#_Toc498030433)

[1.3. Definitions, Acronyms, and Abbreviations 3](#_Toc498030434)

[1.4. Overview 3](#_Toc498030435)

[2. Architectural Representation 4](#_Toc498030436)

[3. Architectural Goals and Constraints 4](#_Toc498030437)

[4. Use-Case View 5](#_Toc498030438)

[4.1. Use-Case Realizations 5](#_Toc498030439)

[5. Logical View 5](#_Toc498030440)

[5.1. Overview 5](#_Toc498030441)

[5.2. Architecturally Significant Design Packages 5](#_Toc498030442)

[6. Process View 5](#_Toc498030443)

[7. Deployment View 6](#_Toc498030444)

[8. Implementation View 6](#_Toc498030445)

[8.1. Overview 6](#_Toc498030446)

[8.2. Layers 6](#_Toc498030447)

[9. Security 6](#_Toc498030448)

[9.1. Authentication and Authorization 6](#_Toc498030449)

[9.2. Encryption 6](#_Toc498030450)

[9.3. User Profiles 6](#_Toc498030451)

[9.4. Permissions and Security Management 6](#_Toc498030452)

[10. Size and Performance 6](#_Toc498030453)

[11. Quality 6](#_Toc498030454)

# Introduction

This document provides a high level overview and explains the whole architecture of EPAM Booking Management System (BMS). It describes how would end users (EPAM employees), administrators (Travel Department) and IT Support be involved into booking management process and it also specifies BMS underlying architecture. The document provides a high-level description of the goals of the architecture, the use cases support by the system and architectural styles and components that have been selected to best achieve the use cases.

## Purpose

This document provides a comprehensive architectural overview of the Booking Management System architecture, using a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions which have been made on the system.

In order to depict the software as accurately as possible, the structure of this document is based on the “4+1” model view of architecture. The “4+1” View Model allows for various stakeholders to find what they need in the software architecture.

## Scope

The scope of this SAD is to depict the architecture of the EPAM Booking Management System (BMS).

It is written for partners, developers, or solution architects who are looking for guidance on architecting and designing an implementation with the product and provides different views of the architecture of the system.

This document acts as a guide for the functional and operational architecture of the solution. It brings a conceptual understanding of the solution architecture.

## Definitions, Acronyms, and Abbreviations

* **SAD** – Software Architecture Document
* **BMS** – Booking Management System
* **QA** – Quality Attribute
* **UML** – Unified Modeling Language

## Overview

In order to fully document all the aspects of the architecture, the Software Architecture Document contains the following subsections:

[Section 2](#_Architectural_Representation): describes the use of each view

[Section 3](#_Architectural_Goals_and): describes the architectural constraints of the system

[Section 4](#_Use-Case_View): describes the most important use-case scenarios

[Section 5](#_Logical_View): describes the architecturally significant parts of the design model

[Section 6](#_Process_View): describes the system's decomposition into lightweight processes

[Section 7](#_Deployment_View): describes how the system will be deployed

[Section 8](#_Implementation_View): describes the overall structure of the implementation model

[Section 9](#_Security): describes the security aspects of the system

[Section 10](#_Size_and_Performance): describes major dimensioning characteristics of the system

[Section 11](#_Quality): describes the quality attributes of the system

# Architectural Representation

The views used to document the EPAM Booking Management system in current phase are:

**Use Case view**

**Audience**: all the stakeholders of the system, including the end-users.

**Area**: describes the set of scenarios and/or use cases that represent some significant, central functionality of the system.

**Related Artifacts**: Use-Case Model, Use-Case documents.

**Logical view**

**Audience**: Designers.

**Area**: Functional Requirements: describes the design's object model. Also describes the most important use-case realizations and business requirements of the system.

**Related Artifacts**: Design model.

**Process view**

**Audience**: Integrators.

**Area**: Non-functional requirements: describes the design's concurrency and synchronization aspects.

**Related Artifacts**: Activity diagram.

**Module Decomposition view**

**Audience**: Programmers.

**Area**: Software components: describes the modules and subsystems of the application.

**Related Artifacts**: Implementation model, components.

**Deployment view**

**Audience**: Infrastructure engineers and managers.

**Area**: Topology: describes the mapping of the software onto the hardware and shows the system's distributed aspects.

**Related Artifacts**: Deployment model.

# Architectural Goals and Constraints

This section describes the software requirements and objectives that have some significant impact on the architecture.

**Technical Platform**

Booking management system will be deployed onto EPAM Cloud clusters with several availability zones related to users work regions.

**Transaction**

Spring Transaction Model will be used.

**Security**

The system must be secured, so that users could have a distinct number of access grants depending on their role in a system (employee / travel managers / administrators). Unauthorized access is prohibited.

**Basic security behavior**:

Authentication: EPAM internal account with SSO authentication capability

Authorization: according to their profile, online user must be granted or not allowed to receive some specific services (self-search options, choose booking options from several available, etc.)

Confidentiality: sensitive data must be encrypted.

Safety: Credit card data must not be kept at a local database.

Data integrity: Data sent across the network cannot be modified by a tier.

Auditing: Every sensitive action can be logged.

Non-repudiation: gives evidence a specific action occurred.

EPAM security model will be reused.

**Persistence**

Data persistence will be addressed using a relational database.

**Reliability/Availability**

High availability is required since there are monetary issues for the EPAM related to the systems availability. The system’s high availability will also ensure user satisfaction and loyalty.

**Targeted availability**: 23 hours a day, 7 days a week (allowed for a maintenance at night hours).

**Performance**

Search queries should be performed less than in 5 seconds for 95% of all requests. Other long running operations must take not more than 10 seconds.

**Internationalization**

Initially the system will support English and Russian. It should be easily modifiable to extend language support.

# Use-Case View

This view exposes the set of use-cases that defines the core functionality of EPAM Booking Management System.

## Use-Case Realizations

Use case functionality diagram below describes how design elements provide the functionalities identified in the significant use-cases. Use cases are displayed as functionalities for the system. Functionality may enclose more than one use-case.

# Logical View

[This section describes the architecturally significant parts of the design model, such as its decomposition into subsystems and packages. And for each significant package, its decomposition into classes and class utilities. You should introduce architecturally significant classes and describe their responsibilities, as well as a few very important relationships, operations, and attributes.]

## Overview

The EPAM Booking Management system is divided into the following layers

* **Presentation layer** deals with the presentation logic and the booking details rendering.
* **Application layer** deals with the core system functionality (create booking request, search, add booking options for supplier, integration interfaces implementation).
* **Persistence layer** is responsible for storing booking available options, details, statistics and a history for the user’s previous orders.

## Architecturally Significant Design Packages

[For each significant package, include a subsection with its name, its brief description, and a diagram with all significant classes and packages contained within the package.

For each significant class in the package, include its name, brief description, and, optionally, a description of some of its major responsibilities, operations, and attributes.]

# Process View

[This section describes the system's decomposition into lightweight processes (single threads of control) and heavyweight processes (groupings of lightweight processes). Organize the section by groups of processes that communicate or interact. Describe the main modes of communication between processes, such as message passing, interrupts, and rendezvous.]

# Deployment View

[This section describes one or more physical network (hardware) configurations on which the software is deployed and run. It is a view of the Deployment Model. At a minimum for each configuration it should indicate the physical nodes (computers, CPUs) that execute the software and their interconnections (bus, LAN, point-to-point, and so on.) Also include a mapping of the processes of the **Process View** onto the physical nodes.]

# Implementation View

## Overview

[This subsection names and defines the various layers and their contents, the rules that govern the inclusion to a given layer, and the boundaries between layers. Include a component diagram that shows the relations between layers. ]

## Layers

[For each layer, include a subsection with its name, an enumeration of the subsystems located in the layer, and a component diagram.]

# Security

## Authentication and Authorization

[Describe the used actor authentication and authorization techniques]

## Encryption

[Describe the applied data encryption techniques, where the keys are stored, how they are accessed, maintained, secured, etc.]

## User Profiles

[User profiles should be described here.]

## Permissions and Security Management

[Management of permissions and security should be described here.]

# Size and Performance

Volumes:

* Targeted member count: 30000
* Daily user access: %5 = 1500
* Estimated member count in 1 month: 4000

# Quality

**Modifiability**

|  |  |
| --- | --- |
| **Tactic** | **Pattern Description** |
| Reduce size of a module | Split Module |
| Increase cohesion | Increase Semantic Coherence |
| Reduce coupling | * Encapsulate * Use and Intermediary * Restrict Dependencies * Abstract Common Services |
| Defer Binding | Try to bind values as late as possible providing more general solutions on each stage.  Compile/Build Stage   * Component replacement (e.g. in a buildscript) * Compile-time parameterization * Aspects   Deployment Stage   * Configuration-time binding   Startup/Initialization Stage   * Resource files   Runtime Stage   * Dynamic lookup * Shared repositories * Publish-Subscribe model * Polymorphism |

**Performance**

|  |  |
| --- | --- |
| **Tactic** | **Pattern Description** |
| Control Resource Demand | * Manage sampling rate (reduce frequency) * Limit Event Response (make queues) * Prioritize events * Reduce overhead * Bound execution times |
| Manage Resources | * Increase resources * Introduce concurrency * Use load balancer * Use caching and data replication * Bound queue size * Schedule resources |

**Usability**

|  |  |
| --- | --- |
| **Tactic** | **Pattern Description** |
| Support User Initiative | * Cancel (free resources, return state to initial) * Undo (return state to previous * Pause/Resume (for long-running tasks) * Aggregate (group operations over single) |
| Support System Initiative | * Maintain task model * Maintain system model * Maintain user model |