**<Project Name>**

**Software Architecture Document**

**Version <0.5>**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| <15/Juni/14> | <0.5> | Eerste versie document | Ilja fiers |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

[1.       Introduction](file:///D:\Development\Rational\RationalUnifiedProcess\webtmpl\templates\a_and_d\rup_sad.htm#1.                  Introduction)

[1.1     Purpose](file:///D:\Development\Rational\RationalUnifiedProcess\webtmpl\templates\a_and_d\rup_sad.htm#1.1               Purpose)

[1.2     Scope](file:///D:\Development\Rational\RationalUnifiedProcess\webtmpl\templates\a_and_d\rup_sad.htm#1.2               Scope)

[1.3     Definitions, Acronyms and Abbreviations](file:///D:\Development\Rational\RationalUnifiedProcess\webtmpl\templates\a_and_d\rup_sad.htm#1.3               Definitions, Acronyms and Abbreviations)

[1.4     References](file:///D:\Development\Rational\RationalUnifiedProcess\webtmpl\templates\a_and_d\rup_sad.htm#1.4               References)

[1.5     Overview](file:///D:\Development\Rational\RationalUnifiedProcess\webtmpl\templates\a_and_d\rup_sad.htm#1.5               Overview)

[2.       Architectural Representation](file:///D:\Development\Rational\RationalUnifiedProcess\webtmpl\templates\a_and_d\rup_sad.htm#2.                  Architectural Representation)

[3.       Architectural Goals and Constraints](file:///D:\Development\Rational\RationalUnifiedProcess\webtmpl\templates\a_and_d\rup_sad.htm#3.                  Architectural Goals and Constraints)

[4.       Use-Case View](file:///D:\Development\Rational\RationalUnifiedProcess\webtmpl\templates\a_and_d\rup_sad.htm#4.                  Use-Case View)

[4.1     Use-Case Realizations](file:///D:\Development\Rational\RationalUnifiedProcess\webtmpl\templates\a_and_d\rup_sad.htm#4.1               Use-Case Realizations)

[5.       Logical View](file:///D:\Development\Rational\RationalUnifiedProcess\webtmpl\templates\a_and_d\rup_sad.htm#5.                  Logical View)

[5.1     Overview](file:///D:\Development\Rational\RationalUnifiedProcess\webtmpl\templates\a_and_d\rup_sad.htm#5.1               Overview)

[5.2     Architecturally Significant Design Packages](file:///D:\Development\Rational\RationalUnifiedProcess\webtmpl\templates\a_and_d\rup_sad.htm#5.2               Architecturally Significant Design Packages)

[6.       Process View](file:///D:\Development\Rational\RationalUnifiedProcess\webtmpl\templates\a_and_d\rup_sad.htm#6.                  Process View)

[7.       Deployment View](file:///D:\Development\Rational\RationalUnifiedProcess\webtmpl\templates\a_and_d\rup_sad.htm#7.                  Deployment View)

[8.       Implementation View](file:///D:\Development\Rational\RationalUnifiedProcess\webtmpl\templates\a_and_d\rup_sad.htm#8.                  Implementation View)

[8.1     Overview](file:///D:\Development\Rational\RationalUnifiedProcess\webtmpl\templates\a_and_d\rup_sad.htm#8.1               Overview)

[8.2     Layers](file:///D:\Development\Rational\RationalUnifiedProcess\webtmpl\templates\a_and_d\rup_sad.htm#8.2               Layers)

[9.       Data View (optional)](file:///D:\Development\Rational\RationalUnifiedProcess\webtmpl\templates\a_and_d\rup_sad.htm#9.                  Data View (optional))

[10.     Size and Performance](file:///D:\Development\Rational\RationalUnifiedProcess\webtmpl\templates\a_and_d\rup_sad.htm#10.             Size and Performance)

[11.      Quality](file:///D:\Development\Rational\RationalUnifiedProcess\webtmpl\templates\a_and_d\rup_sad.htm#11.             Quality)

**Software Architecture Document**

**1.**                  **Introduction**

Dit document beschrijft de implementatie van de casus “Fotolab” zoals ons aangeleverd. Deze uitwerking is het derde deel van het semester Enterprise Software van de NOH-i ICT opleiding.

In de casus “Fotolab” wordt een werkwijze beschreven van een foto reproductie laboratorium, dat foto’s en bijbehorende producten (denk aan frames, posters) produceert en deze door klanten kan laten bestellen.

De casus beschrijft een huidige werkwijze, die ervan uit gaat dat er veel handmatig wordt gewerkt. Daarna worden er verbetervoorstellen gedaan, waarvan wij de architectuur gaan uitwerken en delen van deze architectuur gaan implementeren.

**1.1**               **Purpose**

Dit document is een onderdeel van de op te leveren producten van deze casus. Het bevat een architectuuroverzicht van het te bouwen systeem, en beschrijft diverse aspecten van de casus. Hierin worden ontwerpbeslissingen vastgelegd en overgebracht.

Het is geschreven als product, en dus bedoeld als aanwijzing voor de werkelijke implementatie van de hier beschreven casus. De doelgroep is allereerst wijzelf, zodat we de beschreven onderdelen ook kunnen bouwen. Daarnaast is dit document bedoeld als eindproduct, ter beoordeling van de casus.

**1.2**               **Scope**

In deze casus gaan wij niet alles zoals beschreven implementeren. Het gaat vooral om een goede beschrijving van de onderliggende architectuur, en niet zozeer de implementatie van het geheel. De architectuur gaan wij volledig uitwerken, met een beschreven databasemodel inclusief DDL en Insert-script. Verder maken we een high-level beschrijving van de omgeving waarin deze database draait.

En deel van de beschreven systemen gaan wij ook werkelijk bouwen. Deze systemen worden specifiek benoemd, waarbij de lezer er van uit kan gaan dat delen die niet specifiek genoemd zijn ook niet gebouwd gaan worden.

Dit document gaat kort in op zaken als performance en beveiliging. Deze zaken zijn in ieder geval geen onderdeel van de bouw.

**1.3**               **Definitions, Acronyms and Abbreviations**

*[This subsection should provide the definitions of all terms, acronyms, and abbreviations required to properly interpret the****Software Architecture Document****.  This information may be provided by reference to the project Glossary.]*

**1.4**               **References**

*[This subsection should provide a complete list of all documents referenced elsewhere in the****Software Architecture Document****.  Each document should be identified by title, report number (if applicable), date, and publishing organization.  Specify the sources from which the references can be obtained. This information may be provided by reference to an appendix or to another document.]*

**1.5**               **Overview**

*[This subsection should describe what the rest of the****Software Architecture Document****contains and explain how the****Software Architecture Document****is organized.]*

**2.**                  **Architectural Representation**

*[This section describes what software architecture is for the current system, and how it is represented. Of the****Use-Case****,****Logical****,****Process****,****Deployment****, and****Implementation Views****, it enumerates the views that are necessary, and for each view, explains what types of model elements it contains.]*

**3.**                  **Architectural Goals and Constraints**

*[This section describes the software requirements and objectives that have some significant impact on the architecture, for example, safety, security, privacy, use of an off-the-shelf product, portability, distribution, and reuse. It also captures the special constraints that may apply: design and implementation strategy, development tools, team structure, schedule, legacy code, and so on.]*

**4.**                  **Use-Case View**

*[This section lists use cases or scenarios from the use-case model if they represent some significant, central functionality of the final system, or if they have a large architectural coverage - they exercise many architectural elements, or if they stress or illustrate a specific, delicate point of the architecture.]*

**4.1**               **Use-Case Realizations**

*[This section illustrates how the software actually works by giving a few selected use-case (or scenario) realizations, and explains how the various design model elements contribute to their functionality.]*

**5.**                  **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.]*

**5.1**               **Overview**

Het fotolab bestaat uit de volgende onderdelen, die aan elkaar verbonden zijn.

Database

Een database met alle gegevens van de onderliggende objecten. Deze database draait op een lokale PC en wordt in ons geval op een SQLEXPRESS installatie gedraaid. Deze database wordt mee-geïnstalleerd met alle versies van Visual Studio en lijkt goed te voldoen; er zijn geen relevante beperkingen gevonden tot nu toe.

WebAPI

Toegang tot de database wordt verzorgd door een WebAPI, deze is de enige die toegang heft tot de onderliggende database. Deze WebAPI is opgezet in ASP.NET. en maakt gebruik van HttpRoutes. Dit is een heel eenvoudig te gebruiken methode om een API te implementeren, zo is gebleken.

Website

Er wordt een website geproduceerd in ASP.NET, hierin worden enkele gegevens uit de database getoont zoals bijvoorbeeld een klantenlijst.

Client applicatie

We hebben een client applicatie gebouwd op basis van C# en Winforms. Deze applicatie is in staat om een serie foto’s te verzamelen, hier een kant aan te koppelen, en deze fotos via de WebAPI te uploaden naar de database server. Het resultaat van deze component is een standaard windows applicatie, gebruik makend van .NET.

**5.2**               **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.]*

**6.**                  **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.]*

**7.**                  **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.]*

**8.**                  **Implementation View**

*[This section describes the overall structure of the implementation model, the decomposition of the software into layers and subsystems in the implementation model, and any architecturally significant components.]*

**8.1**               **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. ]*

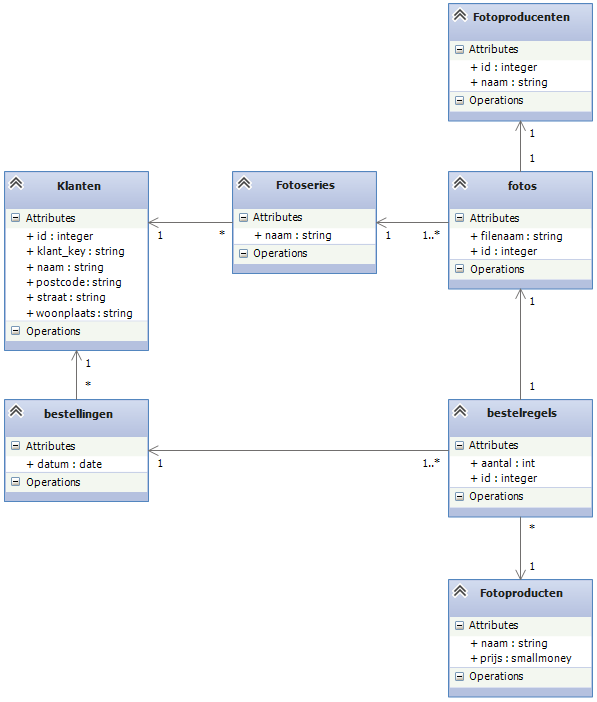
**8.2**               **Layers**

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

**9.**                  **Data View (optional)**

De casus bestaat uit een WebAPI die tegen een redelijk ingewikkelde database aan praat. We hebben deze database gemodelleerd aan de hand van een domeinmodel, wat we opgebouwd hebben aan de hand van de casus zelf. Waar nodig geven we aan waar we nog toevoegingen of optionele veranderingen zouden willen voorstellen. Na het domeinmodel hebben we een DDL en insert script gemaakt, het DDL script is hier ter info aan toegevoegd.

Het domeinmodel ziet er als volgt uit:



Fotoseries:

Een fotoserie heeft een specifieke naam, en een of meerdere foto’s verwijzen er naar. Een fotoserie heeft een klant, maar die is optioneel en mag leeg zijn. Volgens de casus is een groepsfoto in principe te bestellen door iedereen, alle andere foto’s alleen door de klant waarvoor ze zijn gemaakt. We hebben dit gemodelleerd door de klant vermelding mogelijk null te laten zijn, wat dus betekent dat de fotoserie openbaar is.

Bestellingen en bestelregel

Een typische 1 op veel relatie: een bestelling bevat een of meer producten, wat we bestelregels hebben genoemd. Een bestelregel is een bepaalde foto, afgedrukt op een product (poster, foto, mok, etc) met een aantal erbij.

We hebben er voor gekozen om geen afgeleide gegevens in de database op te nemen. We zouden bijvoorbeeld de totaalprijs van een bestelling op kunnen slaan. Dit wordt typisch alleen gedaan als de berekening van deze prijs een duurdere operatie zou zijn dan het opslaan en synchroon houden van dit gegeven. Het is onze inschatting dat dit niet zo is en we de totaalprijs met een SQL query kunnen ophalen indien nodig.

**10.**             **Size and Performance**

Er is in de opdracht geen sprake van constraints voor grootte danwel performance. Wel stellen we vast dat, door te kiezen voor een aanpak van gescheiden componenten (database, webapi, website) dat we door dit ontwerp deze componenten kunnen scheiden en op separate hardware kunnen laten draaien. In de huidige opzet is dit vanwege praktische redenen niet gedaan.

Een verdere verbetering die mogelijk is op het gebied van performance is het inzetten van een broker tussen de website en client application enerzijds, en de WebAPI anderzijds. Hiermee zou een vorm van load balancing bereikt kunnen worden met nog verdere performance verbetering tot gevolg.

**11.**             **Quality**

Drie zaken rond kwaliteit die we hier willen bespreken: uitbreidbaarheid, betrouwbaarheid, en portability.

Zoals eerder besproken is het system door zijn separate componenten goed uit te breiden. Het is mogelijk om separate componenten te vervangen door andere implementaties en/of technieken mocht dit opportuun zijn. De communicatie tussen de componenten wordt vooral uitgevoerd door de WebAPI, deze kan zowel voor zijn clients as voor de database verbinding redelijk makkelijk worden veranderd. De WebAPI is opgezet volgens een model-view-controller opzet, waarbij de view gevormd wordt door de api.

De betrouwbaarheid is wat lastiger te beschrijven: er is geen fallback mechanisme geïmplementeerd aangezien dit geen onderdeel uitmaakt van de casus. De gebruikte technieken zijn industrie standaarden die algemeen bekend zijn. Het is zeker mogelijk om de database engine (MSSQL) te repliceren, ook daar zijn we niet verder op in gegaan.

Portability: de WebAPI is gebouwd met C#, wat een bepaalde vorm van vendor-lockin met zich mee brengt. Dit is een bewuste keuze binnen de casus, maar in theorie is het zonder meer mogelijk de WebAPI te herschrijven naar een andere webtechniek naar keuze (PHP, Node.JS, etc). Hier speelt weer een rol dat de implementatie bestaat uit diverse onafhankeijke en afgeschermde delen.