

**Технически Университет – София**

**Факултет Приложна Математика и Информатика**

**Катедра Информатика**

**АРХИТЕКТУРЕН ПРОЕКТ**

**За**

**Cooking Hub**

**Имена на студентите:** Станислав Бисеров Стоянов, Атанас Иванов Парасков, Димитър Рачев Пашов

**Проектна група:** 16

**Дата:** 26.10.2020г.

**Факултетни номера:** 471218066, 471218069, 471218034

**Съдържание:**

1. Въведение
2. Предназначение на избраната тема

2.1. Обхват на проекта

2.2. Избрани актьори

1. Основни цели на архитектурата - описание на няколко ключови изисквания и ограничения на разработвания софтуерен продукт, които оказват влияние на избраната архитектура
2. Архитектурен обзор

4.1. Описание на основни сценарии (use-case) и актьори (actors), които са важни за разработваната архитектура, чрез use-case диаграми

4.2. Описание на логическия изглед на архитектурата. Клас диаграми за илюстрация връзките между архитектурно значими класове, подсистеми, пакети и слоеве. Диаграми на състоянието за илюстрация конкретни процеси на определени роли.

4.3. Изглед на процесите. Описание на отделните изпълними процеси (подсистеми) и зависимостите между тях.

4.4. Изглед на внедряването (Deployment view)

4.5. Изглед за разработка (Implementation view)

1. Обосновка на това как избраната архитектура осигурява адекватна реализация на поставените нефункционални изисквания
2. Използвани термини и символи
3. Използвани източници

***1. Въведение (Станислав Стоянов и Димитър Пашов)***

С напредването на софтуерните технологии, в наши дни все по-лесно става намирането на информация за различни дейности свързани с бита на модерния човек. Обръщайки внимание на определени необходимости на отделните индивиди, даденият проект цели да представи пример за изготвяне на обществена поръчка за приложение свързано с храната.

Настоящият документ представя софтуерната архитектура служеща за изготвяне на уеб приложение свързано с кулинарията и здравословният начин на живот. Чрез представяне на съществено важните функционални и нефункционални изисквания, представената документация играе ключова роля за правилното разбиране и осмисляне сложността на зададената система.

За изготвянето на подобен проект ще бъде използван архитектурният шаблон за програмиране **MVC** ([Model-View-Controller](https://bg.wikipedia.org/wiki/Model-View-Controller)), като всеки един негов аспект ще бъде подробно описан в настоящия документ.

***2. Предназначение на избраната тема (Станислав Стоянов и Димитър Пашов)***

В дадения документ се представя основната част за изготвянето на проекта, а именно изготвянето и използването на подходяща архитектура за софтуерна разработка. Представяйки определената структура на софтуера, се предоставя началната точка, необходима на софтуерните разработчици да започнат изграждането на продукта. Тази начална точка е с особена тежест и характер, тъй като по определената документация софтуерните разработчици получават опорен план, който след започване на разработката трябва постепенно да бъде надграждан. Също така таргетираната група определя основното предназначение на избраната тема, а именно крайните потребители с интерес към кулинарството и приготвянето на здравословна храна. Освен предоставяне на разнообразни и вдъхновени рецепти от известни готвачи в сферата, **Cooking Hub** ще допринесе за развитието на подобен род системи, осигурявайки на своите потребители необходимата точна информация, лесна достъпност и най-вече интерактивност по отношение приготвянето на храна.

***2.1. Обхват на проекта*** ***(Станислав Стоянов)***

Проектът е насочен към всички крайни потребители с интереси в света на кулиранството. Поради уникалността и разнообразността си, в глобален мащаб проектите много трудно могат да бъдат формализирани. Въпреки това може да се каже, че всеки проект се вписва в един стандартен жизнен цикъл със следните фази:

* Иницииране
* Планиране
* Изпълнение
* Контролиране
* Закриване

Представената архитектура се вписва основно във фазата планиране и служи като опорен стълб за изпълнението на следващата фаза и реалното проектиране на проекта.

***2.2 Избрани актьори (Станислав Стоянов)***

Основните актьори, участващи в описаната архитектура ще бъдат подробно представени в т.4 “Архитектурен обзор” чрез необходимите use-case UML диаграми, клас диаграми и диаграми на последователността и състоянието, тъй като те неизменно участват в бизнес процесите на системата. За повече дейталност и конкретност се представят следните актьори, участващи активно във всички основни функционалности на **Cooking Hub**:

* Администратор
* Модератор
* Обикновен регистрирал се потребител
* Обикновен нерегистрирал се потребител
* Потребител с ограничен достъп
* Потребител с липса на пълен достъп (невъзможност за използване на системата под каквато и да е форма)

***3. Основни цели на архитектурата - описание на няколко ключови изисквания и ограничения на разработвания софтуерен продукт, които оказват влияние на избраната архитектура (Станислав Стоянов)***

Уеб системата за кулинария, за по-кратко навсякъде ще бъде използвана абревиатурата от т.6 “Използвани термини и символи” **CH – Cooking Hub** предоставя възможност за регистрация на крайните потребители, осигурявайки им редица полезни функционалности като:

* Създаване/редакция/изтриване на собствени рецепти
* Преглед на рецептите, написани от останалите потребители
* Създаване на ревю и даване на оценка на рецепта с цел обратна връзка
* Създаване на различни статии, групирани по категории
* Достъп до глобалния чат за обмен на всеобхватна информация по преценка на потребителите

Налице е също така подробна информация за всеки един потребител, изразяваща се под формата на личен профил, където се съхранява профилната снимка на съответния потребител, дата на присъединяване към CH, статистика относно създадените рецепти/ревюта/коментари/статии, информация за използвания провайдър за регистрация и логин било то CH или Facebook. Системата има уеб базиран потребителски интерфейс като избраният архитектурен стил ще бъде MVC ([Model-View-Controller](https://bg.wikipedia.org/wiki/Model-View-Controller)), което ще разреши евентуални бъдещи проблеми свързани с производителността, скалируемостта и ефективността на системата.

* Ключови изисквания

Основните ключови нефункционални изисквания, които трябва да има, са спазване на атрибутите за качество, много добре дефиниран дизайн, ориентиран към потребителя и спазване на различни методи за проектиране и използване на потребителския интерфейс (Usability Heuristics for User Interface Design[[1]](#footnote-1)). На първо място от съществено значение е да бъде взето под внимание времето за реакция на системата (responsiveness), защото това гарантира, че потребителят ще изживее максимално добри чувства ([Emotional design](https://www.interaction-design.org/literature/topics/emotional-design)) докато ползва системата. Дългите закъснения при обработка на заявките трябва да бъдат елиминирани. По-конкретно уеб приложението трябва да отваря бързо в рамките на милисекунди всяка една страница. На следващо място идва сигурността (security), която гарантира, че сензитивните потребителски данни ще бъдат обработвани и използвани само от CH. В това число влиза потребителската парола, пол на потребителя, евентуално предоставен мобилен номер и имейл за връзка с него. Цялата биологическа информация за потребителя не трябва да бъде публично достъпна и използвана без неговото разрешение. Също така не трябва да бъде пренебрегвана надеждността на системата (reliability). Този атрибут за качество гарантира, че средното време между евентуални откази (Mean Time Between Failure MTBF[[2]](#footnote-2)) на системата ще бъде максимално голямо и това може директно да бъде свързано с потребителското изживяване описано по горе.

Добре дефинираният потребителски ориентиран дизайн и спазването на различни методи за проектиране и използване на потребителския интерфейс като: видимост на състоянието на системата, тоест дали в момента тя работи или се извършва някаква поддръжка, съвпадение между системата и реалния свят, постоянство, гъвкавост и ефективност на използването и евентуално възстановяване след грешка, причинена от потребителя са следващите ключови изисквания, които CH трябва да реализира. За да може атрибутите за качество от бизнес гледна точка да бъдат спазени, то трябва много детайлно да бъдат решени въпроси като това дали системата изпраща необходимата обратна връзка на своите потребители под формата на различни известявания (email, контактна форма), дали има съвпадение между очакванията на потребителя за функционалността и реалната такава. На първо място, не е коректно, когато потребителят очаква да получи отговор на изпратено запитване чрез контактната форма и да се получава забавяне и липса на такъв отговор. Особено значим момент е и системата да информира достатъчно добре потребителя за новости по начина на нейното използване, нови статии, както и когато се напише ревю от друг потребител относно дадена рецепта. Постоянството от своя страна представлява потребителят бързо да се ориентира в използването на приложението и да не се чуди кой бутон да натисне или коя страница да отвори, а това се постига единствено със следване на определени конвенции от страна на разработчиците (спазване на КПК – Качество на Програмния Код, SOLID принципи и правилно наименуване на класове и методи). При настъпване на евентуални грешки във CH, потребителят трябва да има възможност за изпращане на контактна форма с конкретния проблем или за по-добро потребителско изживяване Live chat, реализиран с технологията SignalR, където да може конкретно да си зададе въпроса, а от другата страна човек по поддръжка на системата да съдейства максимално бързо. Заложените функционалности при разработката на системата имат за цел да осигурят изброените до тук изисквания.

* Ограничения

Основното ограничение за реализация на системата е времевият фактор. Преди създаването и имплементацията на всички изброени ключови изисквания е необходимо да се разпредели максимално добре времето за имплементация на отделните модули на системата. От атрибутите за качество, тук ясно изразен фактор е съвместимостта на системата (compatiblity), т.е до колко добре системата позволява оперативна съвместимост с по-стара наследена система. Предвид заявения архитектурен стил, системата ще се използва главно в уеб средите и няма да зависи от конкретната операционна система на потребителя. Този атрибут може да бъде пренебрегнат с цел намаляване на разходите и увеличаване на предоставеното време за реализация, като в това число влиза и дизайнът на CH. По-добре е да се отделят повече средства за производителност и надеждност, както и хардуерен ъпгрейд, отколкото за уникално красиво изглеждащо приложение. Друго ограничение е и тестваемостта на системата и до колко добре тя ще бъде покрита с необходимите стрес/load[[3]](#footnote-3) и regression[[4]](#footnote-4) тестове. Трябва да бъдат взети под внимание само тестовете свързани с производителността и надеждността, като целта е да бъде изтествана основната функционалност (при MVC влизат сървисите, отговарящи за бизнес логиката). Освен изброените тестове, ще бъдат реализирани и основните тестове на по-ниско ниво, а именно unit[[5]](#footnote-5) тестове чрез използване на готови външни библиотеки като xUnit/nUnit. И последното съществено ограничение е времето за проектиране и кодиране на цялата система, защото има определени срокове за реализация. Представете си, че определен поръчител има необходимост от създаване на подобна система и се налага бърза реализация на приложение, което да бъде достъпно от всякакви устройства. Ако системата ни бъде бързо проектирана и създадена, то шансовете ни за спечелване овациите и уважението на поръчителя при евентуална продажба на CH са много по-големи.

***4. Архитектурен обзор***

За начало на обзора на съответната архитектура се обръща внимание на нейните изгледи. Водейки се по избрания архитектурен шаблон (MVC), в даденият документ се представят някои от основните изгледи необходими на разработчиците да изградят по-ясна представа за начина на функциониране на системата, както и на участниците, играещи определена роля в нея.

* Use-case изглед
* Логически изглед
* Процесен изглед
* ….

1. Основните методи за проектиране и използване на потребителски интерфейс са създадени през 1990г. от Джакопо Нилсен в колаборация с Ролф Молих. [↑](#footnote-ref-1)
2. MTBF = MTTF (Mean Time To Fail) + MTTR (Mean Time To Repair) [↑](#footnote-ref-2)
3. Стрес/Load тестовете представляват симулация на голям поток от потребителски заявки и гарантират надеждност и липса абнормалност на системата [↑](#footnote-ref-3)
4. Regression тестовете осигуряват това, че поправен бъг в системата няма да се повтори отново в друг компонент [↑](#footnote-ref-4)
5. **Unit testing**-ът е метод за тестване, който проверява дали индивидуалните единици от сорс код работят правилно. Единица или един юнит е най-малката част от едно приложение, която може да се изтества. [↑](#footnote-ref-5)