Applying Patterns to E-Government

Ch. Assist. Prof. PhD. Mariya Armyanova,
University of Economics - Varna, Varna, Bulgaria
armianova@ue-varna.bg

Abstract

This paper present design patterns which simplifies the process of developing, deploying, and managing complex, integrated, and standards-compliant E-government applications. The patterns can be used at different levels and layers of E-government's architecture. Architecture Patterns are the highest layer patterns and have a high level of abstraction. Their aim is to present the different system components and their collaboration through a conceptual model. In the middle layer are design patterns. In e-government, security and object-oriented design patterns are most used. The design patterns application allows to improve the performance of the existing solution, the flexibility and scalability, security and to keep the applications up to date and to incorporate the new requirements. E-government web design patterns allow to create intuitive websites that are suitable for different user categories.

Keywords: Design patterns, E-Government, GoF, Architecture Patterns, Web Design Patterns, Security Design Patterns

JEL Code: C88 *DOI: https://doi.org/10.36997/IJUSV-ESS/2020.9.1.156*

Въведение

Една от основните цели на електронното правителство е да осигури устойчиви услуги за всички и да улесни живота на гражданите. Затова се предлагат различни електронни услуги, като дистанционно подаване, обновяване и издаване на документи, електронно гласуване, поддържане на здравната система и др. Стремежът е електронното правителство да осигури възможно най-разнообразен достъп до услугите така, че да улесни всички категории потребители. Необходим му е интуитивен и лесен за използване уеб интерфейс. Тъй като електронното правителство съхранява данните на всички граждани, то често е обект на хакерски атаки. Това извежда задачата за гарантиране на сигурността на данните, като основна при работата на приложенията на електронното правителство. Трябва да се осигури сигурността на данните в мрежата, тъй като преносът на данните между администрацията и гражданите е през нея. За да се избегне изтичането на поверителна информация, преминаването на данните през глобалната мрежа, трябва да е защитено чрез надеждно криптиране на мрежовия трафик. Друга задача е гарантирането на надеждна автентикация на потребителите. Препоръчително е да се намали излагането на частни данни по време на процедурите по автентикация. Важна задача е подсигуряването срещу пробив в процедурите по обработка и съхранение на данните, в софтуерните приложения на държавната администрация.

Електронното правителство преследва и други цели: ефективност и прозрачност на работата на институциите; по-бързи и качествени услуги за гражданите и събиране на множество разнообразни реални данни с голям обем. Събраните данни могат да се използват за основа на информирано и научно вземане на решенията на всички нива на управление.

Освен това с навлизането на електронното правителство, данните за транзакциите нарастват експоненциално и е трудно да се анализира толкова голям обем данни, съхранявани на множество места с традиционните инструменти за извличане на данни. В световен мащаб обемът на генерираните данни нараства изключително бързо (Sulova, 2020). С увеличаването на използването на мобилните устройства, интелигентните сензори и смарт технологиите се генерира огромно количество данни. Нарастващата цифрова информация се нуждае от по-сложни технологии за съхранение, обработка, защита. Източниците на данни са разнообразни и пряко попълнената информация в системата е от различни публикации, данни от сензори, видео, имейли и друга социална комуникация. Социалните медии

генерират големи обеми данни, свързани с човешкото поведение (Petrov et al., 2020). Социалните медии влияят върху всички области на живота на гражданите и особено на по-младите (Aleksandrova et al., 2019). Затова генерираните от тях големи по-обем данни, следва да се анализират и да се открояят тенденциите.

Използването на шаблоните при разработката на електронното правителство подпомага постигането на целите и задачите му. Те въвеждат принципни постановки при разработването на софтуерните приложения. За да бъдат полезни за разработчиците и да се използват от тях е нужно ясно да се посочи целта им и предимствата от използването им. Шаблоните няма да добият популярност, ако са неясни или в тях има заложено противоречие с някоя от препоръките за електронно правителство.

1. Особености на шаблоните за проектиране за електронното правителство

Шаблоните за проектиране представят доказани решения и известни конструкции, които могат да се използват през целия процес на разработка. Те описват решения, които са утвърдили своята полезност в практиката. Те не са, както завършено софтуерно приложение, което директно да се приложи, така и проектно решение, което може веднага да се трансформира в код. Шаблоните описват популярно решение на даден софтуерен проблем в обобщен вид, за да имат по-голямо приложение. Разработчикът адаптира и прилага шаблона според спецификите на решавания проблем. Затова предложеното в шаблона решение може да има множество реализации, които до голяма степен са независими от езика за програмиране.

Основен критерий за класификация на шаблоните е слоят на архитектурата, чието разработване подпомагат. Има три нива на абстракция – концептуален, логически и физически модел на архитектурата на системата. Шаблоните участват в създаването на различните модели, като нивото на абстракция влияе, както върху засегнатите от тях проблеми, така и върху предложените от тях решения. Шаблоните подпомагат разработването на всеки един архитектурен модел на системата и по този начин поддържат цялостно, комплексно процеса на изграждане на електронното правителство.

Шаблоните, използвани при разработването на концептуалния модел на електронното правителство са архитектурните шаблони. Те определят начина на организация и взаимодействие на елементите му. Подпомагат дефинирането на базовите характеристики и поведение на системата (Richards, 2015). Шаблоните, участващи в разработването на логическия слой на архитектурата или шаблоните за проектиране, имат по-малко влияние – само върху отделен елемент, а не върху цялостното функциониране на системата. Те подпомагат решаването на конкретен проблем на разработката, например за сигурността.

Шаблоните, които подпомагат създаването на технологичния (физическия) архитектурен модел участват в разработването на архитектурата на системата от най-ниско ниво и представят модели, които отразяват особеностите на конкретен елемент като интерфейс.

2. Архитектурни шаблони за електронно правителство

За да се открият шаблоните, поддържащи цялостната архитектура на електронното правителство е необходимо първо да се добие представа за неговата работа. Предложена е обща архитектура на електронното правителство (Figure 1). От едната страна са гражданите с услугите, които използват. От друга страна са държавните учреждения, които осигуряват услугите за гражданите, обединени в електронното правителство. Гражданите и държавната администрация, обменят данни през Интернет, а електронното правителство се реализира чрез облачни услуги, което води до необходимост от решаване на проблемите по сигурността.

За дефинирането на общата архитектура на електронното правителство могат да се

използват архитектурни шаблони. Те представляват множество от дефинирани подсистеми, които постигат определени цели, като предлагат начини за организиране и структуриране на компонентите на системите (Buschmann et al., 1996). Целта им е да представят отделните елементи на системата и взаимодействието им чрез концептуален модел. Те предполагат високо ниво на абстракция и затова показват само компонентите, интерфейсите им и начините за обмен на данни.

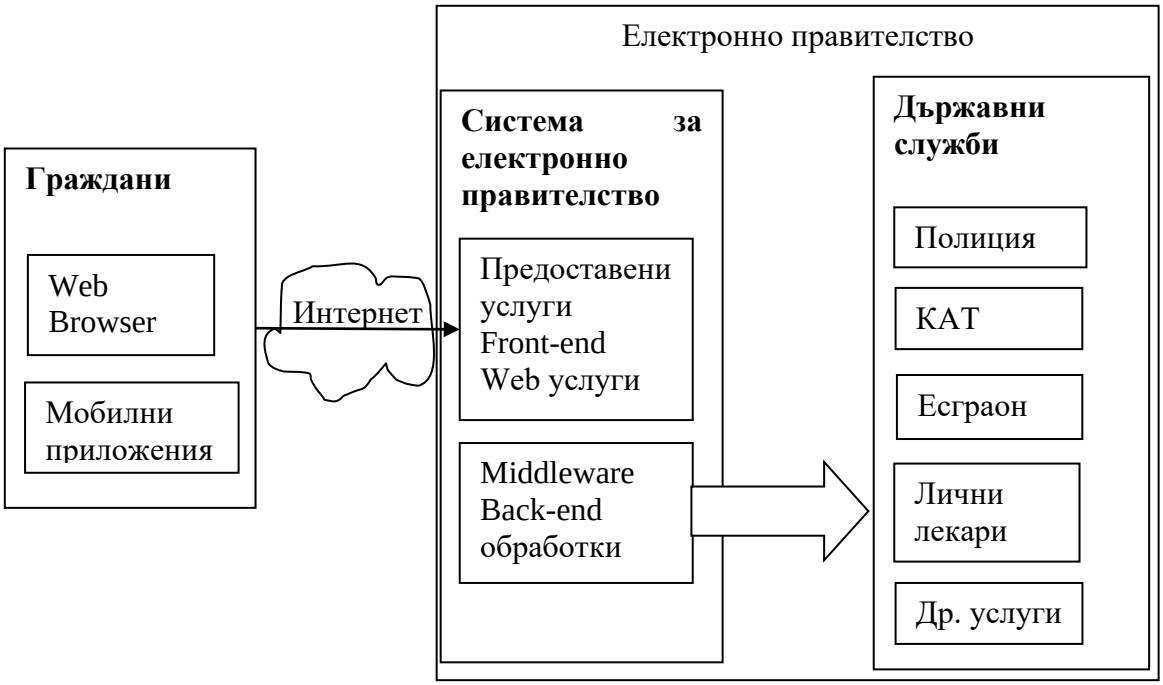

Figure 1. Общ модел на взаимодействието на приложенията на електронно правителство.

Една от основните категории архитектурни шаблони е Layer Architecture (Многослойна архитектура) (Richards, 2015). Затова предложеният вариант за трислойна архитектура на система за електронно правителство може да се тълкува, като архитектурен шаблон (Figure 2). Многослойната архитектура е най-популярния модел, тъй като се поддържа от базите от данни. Системите от този тип се организират така, че винаги в основата на слоевете има отделен слой на данните, обикновено поддържан от базата от данни. Данните преминават през различните слоеве и едва в най-горния слой се използват. Всеки слой има специфична задача, като удостоверяване съгласуваността на данните, или последователна обработка на данните. Основно предимство на многослойната архитектура е възможността всеки слой да се фокусира върху определена ограничена функционалност, което въвежда принципа на декомпозирането при решаването на комплексни проблеми на разработката.

В модела са предложени три слоя: приложен, слой на услугите и слой на данните. Приложният слой се състои от различни мобилни приложения, които потребителите могат да заредят. Той включва и уеб сайтовете, които работят от страна на административните служби. Шаблоните за уеб интерфейс подпомагат реализацията на слоя. В слоя на услугите, работят приложенията на различните административни служби. За тяхната реализация се прилагат различни групи шаблони за проектиране. В слоят на данните се използват предимно шаблони за данни.

Друг архитектурен шаблон, който се използва при разработката на електронното правителство е MVC (Model View Controller). Той е стандартен шаблон за уеб приложенията. Отново функционалността е разпределена по слоеве. Най-горният слой е View (изгледа), който се реализира от CSS, JavaScript и HTML с вграден код. Средният слой е Controller (контролер), определящ правила и методи за трансформиране на данните от вътрешния слой Model (модел) към представянето им в изгледа.

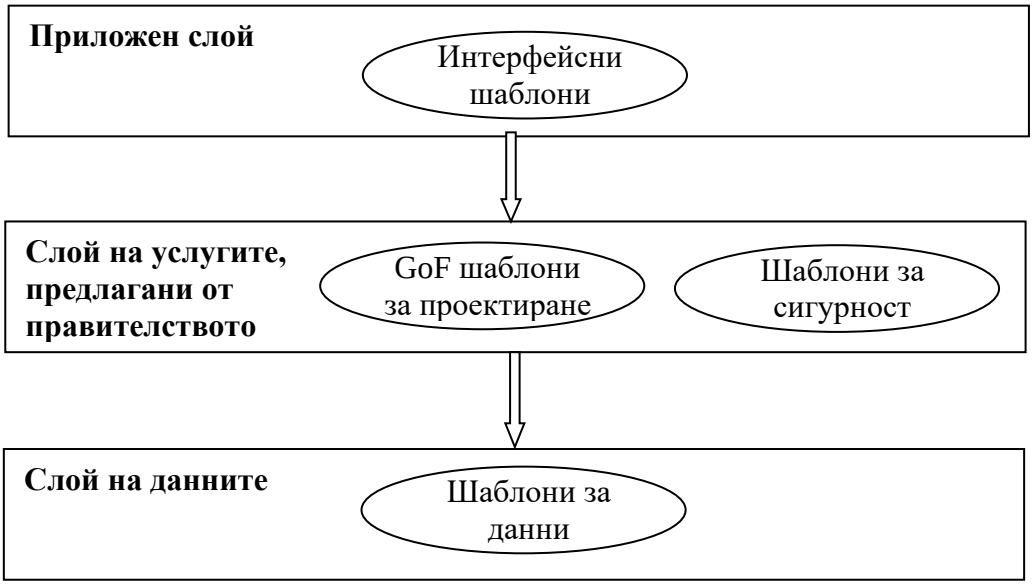

Figure 2. Модел на трислойна архитектура за електронно правителство.

Освен предложеният на Figure 2 има и други архитектурни шаблони. Такъв е шаблонът Ubicio (Medina, 2020). Той съчетава два шаблона за многослойната архитектура и MVC. При него слоевете се формират чрез използването на различни шаблони. В Persistence Layer се използват два шаблона за работа с данни.

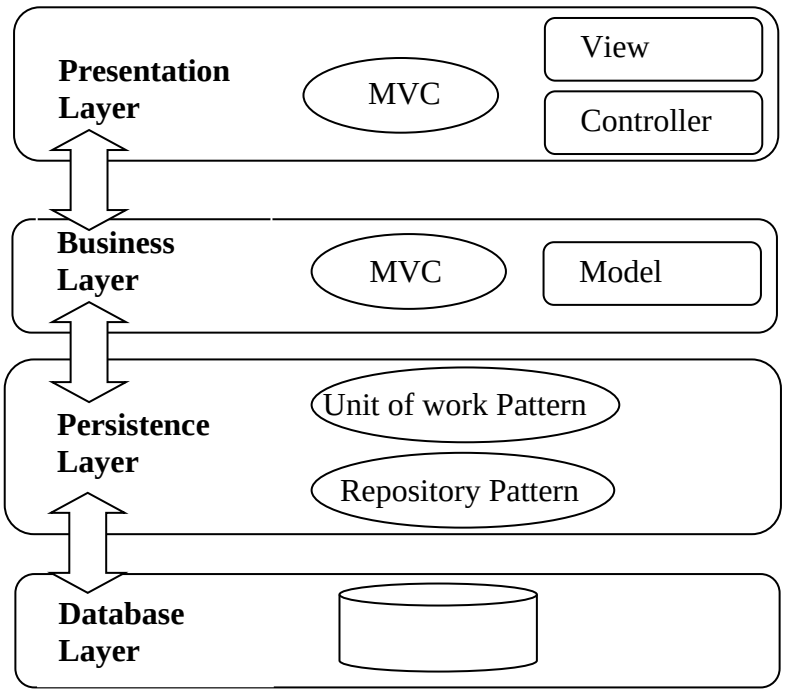

Figure 3. Модел на архитектурния шаблон Ubicio.

При изграждането на електронното правителство може да се използва всеки един от предложените архитектурни шаблони.

3. Шаблони за проектиране за уеб страниците на електронното правителство

Шаблоните за проектиране на уеб страниците на електронно правителство се стремят да подпомогнат създаването на интуитивни уеб сайтове. Те имат две основни цели – да бъдат напълно приложими, без корекции за държавните уеб сайтове и да не са обвързани с временни тенденции в изграждането и структурирането на сайт. Те трябва да са независими от модерните технологии, които също остаряват.

Тези шаблони са от най-ниското ниво на разработката на софтуер и се отнасят до конкретен елемент на едно приложение – интерфейса. Те имат ограничено влияние върху цялостната структура на електронното правителство.

Предполага се, че сайтовете за електронно правителство са устойчиви за по-дълъг период и се придържат към постоянни решения. Въпреки непрекъснатото развитие на технологиите и модата в дизайна на уеб сайтовете, е важно да се открият общите опорни точки. За да бъде едно решение шаблон, то трябва да се отнася за проблем, който се появява често. Освен това решението следва да може да се реализира със средствата на различните езици за програмиране и инструментите на различните технологии. За да се открият шаблоните, следва да се разгледат както добри, така и лоши примери за уеб сайтове (Aquino et al., 2005). Така са предложени много колекции от интерфейсни шаблони. Но Йохансон, Филгуирас, Акуино (Johansson et al., 2015) предлагат 17 шаблона пригодени за електронното правителство. Те са разделени в четири категории според елемента на сайта, които подпомагат – структурни, навигационни, информационни и визуални (Table 1).

Table 1. Видове уеб-интерфейсни шаблони

№	Уеб шаблон	Приложение
1	**Структурни шаблони**	
	Мозък	Подпомага откриването на информация в страницата
	Приоритет	Осигурява бърз достъп до отделни елементи чрез акцентиране върху често използвани пътища
	Справка	Препраща към допълнителни източници на данни, ако е възможно.
2	**Навигационни шаблони**	
	Контакт	Публикува информация за връзка със служител
	Пазител	Осигурява възможност за връщане, ако потребителите въведат заявка за зареждане на липсващ елемент в сайта
	Карта	Показва преминатите страници от началото на сайта до текущото местоположение
	Отбелязка	Отбелязва текущото положение в сайта и позволява по-късното връщане към него
	Дубликат	Дублира функционалните възможности на сайта, така че да са достъпни от всички възможни версии на браузерите
	Уизард	Води потребителя при последователност от действия или попълване на формуляр
3	**Информационни шаблони**	
	Дата	Публикува срокове и дати при въвеждане на промени
	Пазач	Проверява въведените данни
	Ключова дума	Включва пояснения за по-сложни термини или показване на примерни данни

	Пророк	При извършване на неделима последователност от действия са необходими предварителни указания за въвежданите данни или време
	Художник	Замества поясненията с илюстрация или схема
	Учител	При промяна показва промените в сайта и новите му възможности
4	**Визуални шаблони**	
	Известяване	Показва или пояснява предварително, какво следва от даден потребителски избор
	Преформатиране	Представя страница и в друг изглед подходящ за печат

Структурните шаблони акцентират върху подреждането и структурирането на съдържанието и елементите на сайта. Представител на структурните шаблони е Brain (Мозък). За да се ориентират по-лесно потребителите в съдържанието на сайта се предполага подреждането на основните заглавия в него в йерархична структура и създаване на връзки за достъп към тях. Шаблонът Приоритет (Priority) предполага по-бързо откриване на отделни функции и страници, като се акцентира върху най-често използвания маршрут за достигането им. Друг шаблон е Справка (Reference). Той предполага добавяне на връзки за допълнителни справки по темата.

Навигационните шаблони определят маршрутите и последователността от стъпки за зареждане на страниците в сайта. Към навигационните шаблони се отнася шаблонът Контакт (Contact). Той предполага да се публикува в сайта начин за осъществяване на пряко взаимодействие със служител. Друг шаблон от тази група е Пазител (Guardian). Той позволява на потребителите да се върнат към сайта и след заявка за зареждане на несъществуваща в сайта страница. Шаблонът Карта (Map) позволява лесно откриване на пътя до прегледани страници чрез показване на последователността, от стъпките на потребителя до достигане на настоящото му местоположение в сайта. Друг шаблон е Отбелязка (Memorizer). Той позволява на потребителя да постави отбелязка за важни данни, така че след време да се обърне отново към тях. Шаблонът Дубликат (Replicator) предполага създаване на функционалността в сайта в съответствие с различните версии на браузерите, за да се поддържа от тях. Друг шаблон е Уизард (Wizard). При дълга последователност от действия или при попълване на важен документ се определят стъпки, т.е разделят се на групи данните. Представените стъпки трябва да са последователни и логични.

Информационните шаблони въвеждат стандарти и изисквания към съдържанието на сайта. Шаблонът Дата (Date) се отнася към информационните шаблони. Същността му се изразява в публикуване на точни срокове при необходимост от внасяне на промени в сайт на дадена услуга. Друг шаблон от същата група е Пазач (Keeper), който предполага верифициране на въведените данни и връщане на бърз отговор. Шаблонът Ключова дума (Keyword) изисква да се даде възможност за извикване на контекстна помощ за неизвестни термини или при по-сложни ситуации. Друг шаблон отнасящ се за поясненията в сайта е Художник (Painter). Той изисква въвеждането на картинка, като пояснение в случаите, когато едно пояснение ще отвлече вниманието от основната тема. Шаблонът Пророк (Oracle) се използва при необходимост от извършване на дълга последователност от неделими действия или попълването на дълъг неделим формуляр. Той предварително дава точни указания на потребителя за данните, които трябва да попълни или времето, което ще му е необходимо. Така, че да не се налага прекъсване на работата и след това започване отначало на въвеждането. Друг шаблон е Учител (Teacher). Вместо да се предоставя стара версия на сайта при промяна, е желателно да се подпомогнат потребителите с усвояването на новата функционалност с допълнителни пояснения.

Визуалните шаблони се отнасят до възможностите на потребителя за бърз преглед и

представяне в различни изгледи на документ или друго съдържание. Към категорията на визуалните шаблони се отнаса шаблонът Известяване (Notifier). За да се чувстват спокойни потребителите, би следвало да има публикувани пояснения, преди да направят избор за извършване на определени действия. Шаблонът Преформатиране (Reformater) дава възможност на потребителя да разгледа страница в друг вид, като документ или подготвена за отпечатване.

4. Шаблони за проектиране на приложенията за електронното правителство

Средният слой, този на услугите предлагани от правителството, представя специфично за всяко административно подразделение софтуерно приложение с електронни услуги, които са в пряка зависимост от предназначението му. Обаче програмните конструкция, решения и модели на данните са подобни при различните приложения и могат да се използват за множество компоненти на електронното правителство, най-често в различни подразделения. Приложенията имат подобни процеси, като идентификация и удостоверяване на потребителя, проверка на адреса, проверка на медицински сертификати, одит и регистрация. Тези процеси могат да бъдат предоставени чрез шаблони или дори включени в софтуерни рамки и да се използват от всички приложения в правителството.

От друга страна дейностите, които трябва да се реализират от електронното правителство са изключително сложни, многобройни и се променят бързо. Непрекъснато нараства броят на потребителите и предлаганите услуги. Около 85% от процесите в различните административни подразделение са едни и същи (Mittal et al., 2004). Затова повечето разработчици променят вече съществуващото решение, за да отговори на отделната държавна агенция. Този начин на разработка, обаче има редица недостатъци. Проблем е гарантирането на производителността на съществуващото решение, гъвкавостта, разширяемостта и мащабираемостта му. Особено трудно става поддържането на актуалността на приложенията и включването на новите изисквания. Затова по-добър подход е използването на шаблоните за проектиране, които позволяват повторно използване на доказани в практиката решения на често срещани софтуерни проблеми. От същото ниво на декомпозиция (нивото на приложенията) са класическите обектно-ориентирани шаблони на GoF (Gamma et al., 2004). Затова е уместно да се открият подходящи шаблони в рамките на отделните приложения. Друго предимство на тези шаблони е, че са обектно-ориентирани.

Подходящ за използване при разработката на приложения за електронното правителство е шаблонът Наблюдател (Observer) (Parikh et al., 2008). Той дефинира множество класове, които се уведомяват при промяна на даден клас. Той е поведенчески шаблон и като такъв представя алгоритмите и начините за разпределяне на задълженията между обектите. Позволява лесно да се добавят нови обекти, тъй като обвързаността между субектите и наблюдателите се свежда до минимум. Субектът знае единствено интерфейса, използван от наблюдателя. Това осигурява гъвкавост на обектно-ориентирания модел. По всяко време могат да се добавят нови наблюдатели, без да се правят каквито и да е промени в класовете на субектите. Може да се използват повторно субекти или наблюдатели независимо. Промяната на едните не повлиява на другите (Figure 4).

Шаблонът Наблюдател е подходящ, когато има два зависими един от друг аспекта на приложението, които след това подлежат на независима промяна. Или когато обект трябва да уведомява променящ се брой обекти. Тоест, когато промяната на един обект може да предизвика промени в други обекти. Например подходящо приложение на шаблона е приложение за разпространението на правителствени решения, съобщения и резолюции, които засягат много различни административни отдели.

Има и други подходящи шаблони (Parikh, 2010). Шаблонът Декоратор (Decorator) може да се използва, за да се добавят динамично отговорности към обектите. Той е определен, като структурен шаблон и като такъв описва начините, по които класовете и обектите биват композирани за формиране на по-големи структури. Той предоставя гъвкава

алтернатива на наследяването при разширяване функционалността на системата (Figure 5).

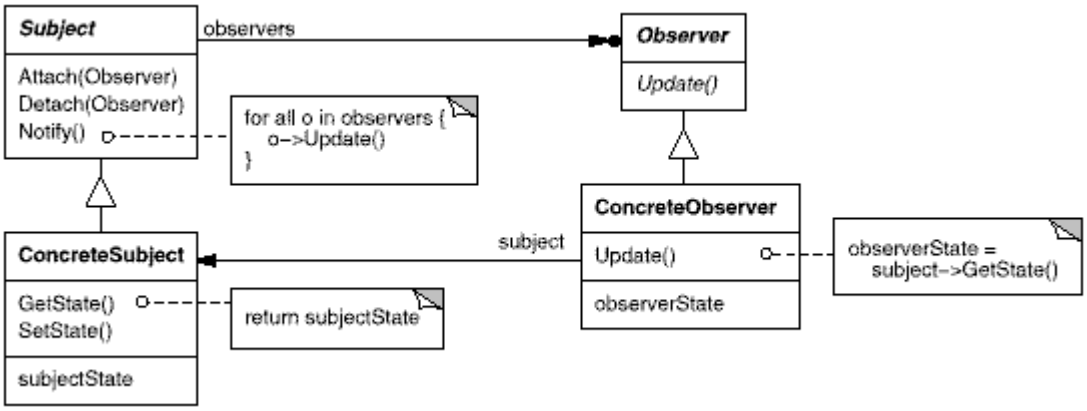

Figure 4. Диаграма на класовете за шаблона Наблюдател.

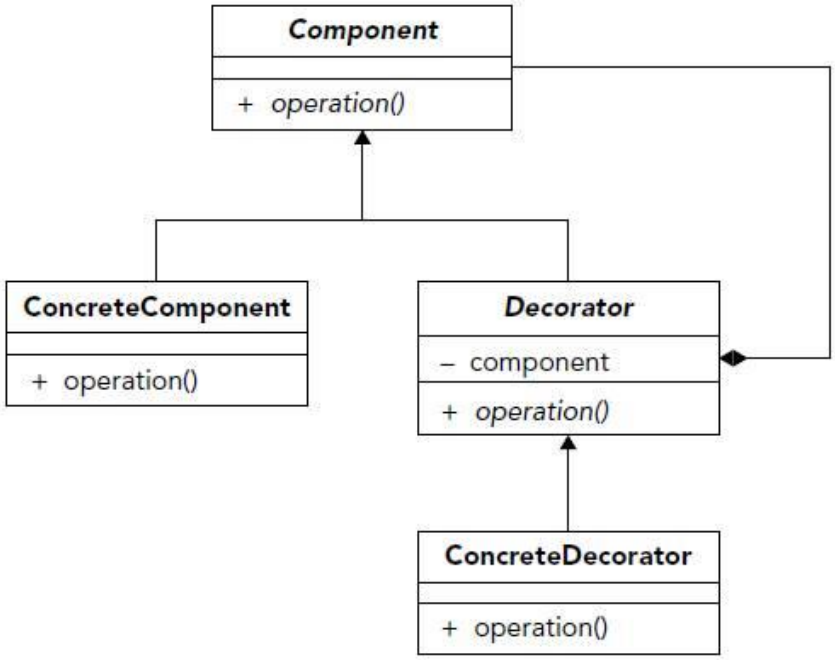

Figure 5. Диаграма на класовете за шаблона Декоратор.

Шаблонът Верига от отговорности (Chain of responsibility) е от типа на шаблоните, които определят поведението на приложенията. Той позволява разделяне на класовете, като избягва обвързването на изпращача на дадена заявка с получателя й. Свързва заедно приемащите обекти и предава заявката по веригата, докато някой я разпознае и обработи. Например може да се използва за предаване на заявка, молба или заявление нагоре по управленската верига. Всяко управленско ниво има собствени правомощия и след като разгледат заявката може да я одобрят, отхвърлят или изпратят на по-горно ниво. Така е възможно много хора да работят по дадена заявка без да имат връзка помежду си (Figure 6).

Друга голяма група шаблони, която се прилага при разработката на почти всички приложения, които са част от електронното правителство са шаблоните за сигурност. Тъй като приложенията функционират в облачна среда, най-голямо приложение имат шаблоните от категорията шаблони за сигурност в облачните системи. След анализ на различни шаблони, са разгледани само тези, които се използват често в приложенията на електронното правителство. Такива са шаблоните на Ърл (Erl, 2015).

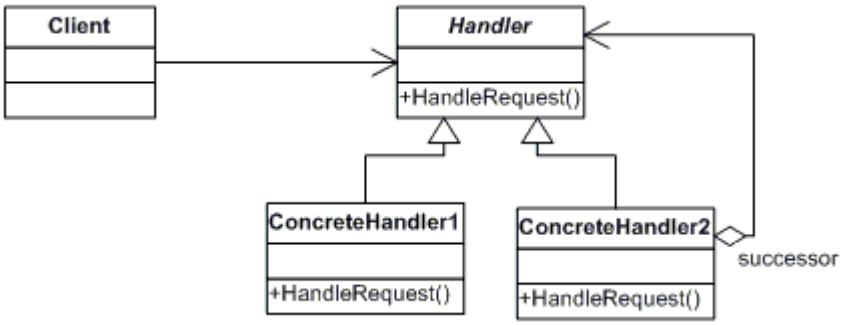

Figure 6. Диаграма на класовете за шаблона Верига от отговорности.

Криптирането е основополагащо за сигурността на приложенията в електронното правителство, но управлението на криптиращите ключове е едно от най-трудните предизвикателства за реализация. Неадекватното управление води до редица административни проблеми и проблеми с поверителността на данните, които се съхраняват или сигурността на данните при преноса им. Шаблон, които решава тези проблеми е шаблонът Управление на ключовете в облак (Cloud Key Management). Той предполага използване на система за управление на ключовете в облак, която се реализира чрез физическо или виртуално устройство, свързано към мрежа. Този модел може да се комбинира с други шаблони, като Система за управление на криптографски ключове (Cryptographic Key Management System), шаблони за модули за хардуерна защита (Hardware Security Module patterns), за да се повиши нивото на защита.

Друг шаблон е Криптиране на данните на Rest[1] (Encryption of Data at Rest). Той позволява реализация на защита на съхраняваните данни чрез защита на достъпа до физическите устройства за съхраняване на данни. Той предлага използване на механизъм за криптиране, поддържан от устройствата за данни, който автоматично да кодира данни при съхранението им на дисковете, и да ги декодира при извличането им. Това е начин да се предпазят данните от неправомерен достъп.

Приложенията, осигуряващи различни електронни услуги, използват ключове, за да гарантират сигурността на достъпа до услугите. Проблемът е управлението на ключовете в съответствие с регулаторните политики. Шаблонът Управление на ключовете за кодиране (Cryptographic Key Management) предполага въвеждане на системата за управление на криптографските ключове, състояща се от правила, процедури, компоненти и устройства за защита, управление и предоставяне на ключове за кодиране във вид на метаданни. Системата се състои от всички устройства или подсистеми, които имат достъп до некриптиран ключ или метаданни. Криптираните ключове и техните криптографски защитени метаданни могат да се обработват от компютри, да се предават чрез комуникационни системи и да се съхраняват и предават през Интернет и устройства, които не са част системата за управление.

Приложно-програмните интерфейси (API) трябва да предоставят гъвкави интерфейси за сигурност. Шаблон Защита на облачните и API интерфейси (Secure Cloud Interfaces and APIs) се базира на създаване на система за идентификация и управление на достъпа. Целта е да се разграничават легитимните потребители от нарушителите. Предполага се достъп до системата само през точки за идентифициране на достъпа (authentication gateway service AGS), в които следва да се вложат механизми за разпознаване на потребителите. Те също могат да се реализират на база концепциите на други шаблони. Предоставянето на малко и защитени точки за проникване увеличава надеждността на процедурите по автентикация на

1 Representational State Transfer

потребителите.

Незащитените данни са уязвими на голямо разнообразие от нарушения, които могат да имат значителни последствия за сигурността на клиента на електронната услуга. Решение на проблема е създаване на система, която осигурява кодиране на важните данни, така че дори и данните да станат достояние на лица без право на достъп, те да не могат да ги разкодират без цялостната система и така данните да не са разбираеми. Шаблонът Защита на данните в облака (Cloud Data Breach Protection) дава такава възможност, като обединява криптирането, управлението и политиките за сигурност.

Шаблонът Федеративна идентичност (Federated Identity) разчита на делегиране на права на външен доставчик за удостоверяване на самоличността на потребителите. По този начин шаблонът постига сигурност на идентифицирането на потребителите, като същевременно опростява разработката. Свежда до минимум необходимостта от администрирането им и същевременно подобрява практическата работа на приложението.

Шаблонът Федеративна идентичност предлага решение, при което идентифицирането на потребителя, вече не е задача на приложението. Разчита се на делегиране на отговорността по автентикацията на потребителите на специализиран, сигурен доставчик на услугата по идентифициране. Използването на шаблона Федеративна идентичност е подходящо решение за клиентското приложение, което предлага достъп до услуга, изискваща идентифициране на ползвателите. Автентикацията се извършва от доставчик на услугата по идентифициране, който разпознава цифрови сертификати, издадени от нарочен сертификационен орган (Certificate Authority (CA)), като част от инфраструктура с публичен ключ. Цифровите сертификати предоставят информация за идентифициране на потребителя. Те съдържат информация за самоличността на потребителя, но могат да съдържат друга информация, като роля му, оторизациите и права за достъп.

5. Шаблони за работата с данни

Някои от шаблоните за работа с данни са част от слоя на данните, предложен от архитектурните шаблони в т. 2. Други, като шаблоните подпомагащи обработката на големи по обем данни, обхващат няколко слоя на архитектурата.

Шаблонът Generic Repository Pattern (Общ модел на хранилището) се използва в системи, при които функционалността е зависима от данните (Dykstra, 2013). Затова е подходящ за използване при реализиране на слоя, осигуряващ достъпа до данните. Целта на шаблона е да осигури съхранение на общите данни в споделеното хранилище, с които работят няколко компонента на приложенията. Състоянието на данните в хранилището оказва влияние върху конкретен поток на компонентите. Прекият достъп до данните крие рискове, като дублиране на кода, грешки в програмите, трудности при създаване на кеш за данни. Обектно-ориентираните приложения и релационните бази от данни използват различни механизми за структуриране на данните. Затова шаблонът се вмъква между слоя на приложенията и слоя, който препраща данните. Той позволява да се установи стандартен единен метод за достъп до данните, който да кореспондира с SQL server database engine.

Друг шаблон, който се използва в съчетание с него е Unit of work (Единица работа). Той представя идеята за създаване на указател, показващ свързването на всеки обект на приложението с необходимите данни от базата от данни. Шаблонът се явява посредник при преноса на данни и извършва всички необходими техни преобразувания и гарантира съгласуваност между представянето на едни и същи данни. В същото време позволява независимото развитие на релационния модел на данните и обектно-ориентирания модел на приложението.

Събраните данни от електронното управление могат да се обработват с различни методи и системи за подкрепа на вземането на решения, но изискват подходяща инфраструктура за събиране и манипулиране на големи по обем данни. Използват се

складове от данни, за да се съхраняват данни от минали периоди, нужни за проследяване на тенденции и вземане на решения. Могат да се използват в редица сектори, като земеделие, здравеопазване, социално осигуряване, образование и др. Изграждането на складове от данни има някои недостатъци, сред които и високата стойност на разходите по създаването им. За да се улесни обработката се въвежда идеята за използване на шаблонен модел на данните за обмена им и обработка на данните при източника им.

Архитектурният шаблон Pattern Based Data Sharing (Споделяне на данни по определен модел) подпомага работа с големи данни (Suganya, 2017). При повечето системи, работещи с големи данни, необработените данни се събират в централни хранилища и в последствие се анализират. Идеята на шаблона е събраните данни да се подлагат на локална обработка и след това да се изпращат. При мобилните приложения може да се използва мощността на мобилното устройство, като данните да се обработят и приведат се към определен модел и така се изпратят за анализ (Figure 7).

Figure 7. Модел на шаблон Pattern Based Data Sharing (Habib et al.,2015).

Шаблонът определя пет стъпки за обработка на данните, четири от които се извършват от мобилното устройство. Те са: събиране на данни (data acquisition), обработката им (data processing), извличане на знания (knowledge discovery), изпращането им по шаблон (pattern sharing) и дистанционно управление на данни (remote data management).

Друг шаблон е Garbage collection from big data (Събиране на боклука в системите за големи данни). Съществуващите възможности на езиците от високо ниво, като .Net или Java не са достатъчни за работа в разпределени системи. Използват се различни алгоритми за реализация: преброяване на референции, маркиране и почистване, спиране и копиране на Garbage collection. Обаче използването на шаблона Pattern Based Data Sharing, реализира и модел за Garbage collection за големи данни. Предварително обработените данни се анализират с помощта на различни алгоритми за извличане на данни, за класификация, клъстериране и асоцииране. Извлечените данни се представят с модели, синхронизирани с отдалечени хранилища на данни в облачни среди, споделените модели се анализират допълнително в среди за големи данни, използвайки различни техники за извличане на големи данни. При дистанционното управление на данните споделените бутони се управляват в облачна среда, като се използват различни структурирани и неструктурирани системи за управление на данни. И няма нужда от последваща обработка с Garbage collection.

Заключение

Прилагането на шаблоните при разработването на електронното правителство подпомага решаването на редица проблеми. Отделните приложения, влизащи в състава на електронното правителство се изменят непрекъснато. Чрез архитектурните шаблони и шаблоните за проектиране би могло да се създаде гъвкава архитектура, която позволява непрекъснатото добавяне на нови приложения към електронното правителство и лесното въвеждане на новите правителствени изисквания в отделните приложения. За целта е достатъчно да се идентифицира подходящ шаблон, в които да се въведат индивидуалните особености на приложението. Шаблоните за сигурност сравнително лесно могат да се използват, така че електронното правителство да отговори на изискванията за сигурност. Чрез тях могат да се намалят както разходите, така и рисковете за сигурността. Шаблоните за уеб дизайн позволяват страниците на електронното правителство да се направят максимално удобни за гражданите. Шаблоните за данните подпомагат работа с големите по обем данни.

References

1. Aleksandrova, Y., Parusheva, S., (2019) Social Media Usage Patterns in Higher Education Institutions – An Empirical Study. International Journal of Emerging Technologies in Learning (iJET) – Vol. 14, No. 5, 2019: pp. 108-121.
2. Aquino, P., Filgueiras, L., Oliveira, E., Bello, T. (2005) Patterns de interface em PDA: aplicação em coleta de dados de usabilidade. *CLIHC '05: Proceedings of the 2005 Latin American conference on Human-computer interaction*. October 2005, pp. 341. Available from: https://doi.org/10.1145/1111360.1111401 [Accessed 20/10/2020].
3. Buschmann, F., Sommerland, P., Stal, M., Meunier, R., Rohnert, H. (1996) *Pattern-Oriented Software Architecture: A System of Patterns*, vol. 1, Wiley, Hoboken.
4. Dykstra, T., (2013) *Implementing the Repository and Unit of Work Patterns in an ASP.NET MVC Application.* Available from: https://docs.microsoft.com/en-us/aspnet/mvc/overview/older-versions/getting-started-with-ef-5-using-mvc-4/implementing-the-repository-and-unit-of-work-patterns-in-an-asp-net-mvc-application [Accessed 20/10/2020].
5. Erl, T. & Cope, R. & Naserpour, A. (2015) *Service-Oriented Architecture: Concepts, Technology, and Design.* н.м.: Prentice Hall.
6. Gamma, E., Helm, R., Johnson, R., Vlissides, J. (2004) Design Patterns: *Elements of Reusable ObjectOriented Software*. Reading, Addison-Wesley Professional Computing Series.
7. Habib ur Rehman, M., Batool, A. (2015) The Concept of Pattern based Data Sharing in Big Data Environments. *International Journal of Database Theory and Applications.* Vol.8, No.4, pp.11-18.
8. Johansson, J.F., Filgueiras, L., Aquino, P. (2015) *Web design patterns for E-Government websites*. [Online] Available from: https://www.ip.do/wp-content/uploads/2015/04/SIGCHI_paper_draft_v4.pdf [Accessed 20/10/2020].
9. Medina, O., Romero, M., Marciszack, M.M. (2020) Using Architecture Patterns in the Conceptual Model of an eGov Software. *ICITS 2020, AISC 1137*. pp. 54-63.
10. Mittal, P.A., Kumar, M., Mohania Nair, M. (2004) A framework for eGovernance solutions *IBM Journal of Research and Development*. volume 48, issue: 5.6. pp. 717-733.
11. Petrov, P., Nacheva, R. (2020) *Informacionni sistemi za socialna biznes analitichnost w realno vreme.* Science and Economic – Varna.
12. Richards, M. (2015) *Software Architecture Patterns.* O'Reilly, Newton.
13. Parikh, A. Buddhdev, B. (2008) E-Governance Solution Based on Observer Design Pattern. *National Conference on architecture Future IT Systems* (NCAFIS'2008). Leeds: School of Computer Science & Information Technology, Devi Ahilya University, Indore.
14. Parikh, A. (2010) Software Design Patterns For E-governance Solution Framework. *Proceedings of the 4th National Conference; INDIACom-2010,* Computing For Nation Development, February 25-26. Leeds: Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi.
15. Suganya, M.J., Deepamalar, M. (2017) Implementation of E-Government Using Pattern Based Data Sharing and Garbage Collection in Big Data Environments. *International Journal of Innovative Research in Science, Engineering and Technology.* Vol. 6, Issue 5, May 2017. pp. 9212-9220.
16. Sulova, S. (2020) Predizvikatelstvata pred upravlenieto na dannite v usloviata na digitalna transformaciq na biznesa. *Jubilee International Scientific Conference dedicated to the 100th anniversary of the University of Economics – Varna*, vol.1(1), May 2020. Leeds: University of Economics – Varna.