ПЛОВДИВСКИ УНИВЕРСИТЕТ „ПАИСИЙ ХИЛЕНДАРСКИ“

ФАКУЛТЕТ ПО МАТЕМАТИКА И ИНФОРМАТИКА

КАТЕДРА „СОФТУЕРНИ ТЕХНОЛОГИИ“

Веселина Руменова Нанева

Уеб базирани инструменти за визуализация на данни

АВТОРЕФЕРАТ

на ДИСЕРТАЦИОНЕН ТРУД

за присъждане на образователна и научна степен „*доктор*“
по област на висше образование:
4. Природни науки, математика и информатика,
професионално направление: *4.6. Информатика и компютърни науки*,
докторска програма: *Информатика*

Научни ръководители:

проф. д-р Ангел Голев

проф. д-р Николай Павлов

Пловдив, 2023 г.

Дисертационният труд е обсъден и насрочен за защита на разширено заседание на катедра „Софтуерни технологии“ при Факултета по математика и информатика на Пловдивския университет „Паисий Хилендарски“ на 18.10.2023 г.

Дисертационният труд **„Уеб базирани инструменти за визуализация на данни“** съдържа 121 страници, от които 106 основен текст. В допълнение е включено Приложение в обем от 19 страници с прилежащи кодови фрагменти. Цитираната литература включва 86 източника. Списъкът с авторските публикации по темата съдържа 4 заглавия, всички от които на английски език.

Защитата на дисертационния труд ще се състои на 22.01.2024 г. от 12:00 ч. в Заседателната зала на Нова сграда на Пловдивски университет „Паисий Хилендарски“, гр. Пловдив.

Материалите по защитата са на разположение на интересуващите се в секретариата на ФМИ, Нова сграда на ПУ, каб. 330, всеки работен ден от 8:30 до 17:00 ч.

НАУЧНО ЖУРИ:

Председател:

Проф. д-р Асен Кънчев Рахнев (Пловдивски университет „Паисий Хилендарски“, гр. Пловдив)

Членове:

1. Проф. д.т.н. инж. Тодор Атанасов Стоилов (Институт по информационни и комуникационни технологии на БАН, гр. София);
2. Проф. д-р Георги Теохаров Тупаров (Нов български университет, гр. София);
3. Проф. д-р Кольо Златанов Онков (Аграрен университет, гр. Пловдив);
4. Проф. д-р Ася Георгиева Стоянова-Дойчева (Пловдивски университет „Паисий Хилендарски“, гр. Пловдив).

Автор: Веселина Руменова Нанева

Заглавие: „Уеб базирани инструменти за визуализация на данни“

Пловдив, 2023 г.

Съдържание

Използвани съкращения

BI – Business Intelligence или Бизнес разузнаване

ETL – Extract, Transform, Load или Извличане, Трансформиране, Зареждане

Power BI – Microsoft Power BI

ППИ – Приложно-програмен интерфейс

Триъгълник – „Триъгълник на застрахователните претенции“

Увод

Актуалност на темата

Голяма част от бизнес единиците в днешно време имат пристрастие към софтуерните продукти, които използват, базирано на фактори като цена, качество, достъпност, съгласуваност на колекция от услуги и други. От своя страна, доставчиците на Business Intelligence решения непрекъснато се конкурират в обновяването на предоставените функционалности и възможности на своите продукти. Тази непрестанна „борба“ за превес и конкуренция извежда основополагащ проблем, който адресира целевата насоченост на настоящия дисертационен труд. Миграцията от една BI среда в друга, погледната от перспективата на персонализираните визуализации, реализирани до момента, би коствала допълнителни средства и време за разучаване на новата технология. Внедрените корпоративни визуализации следва да отговарят на същите клиентски изисквания по отношение на функционалност, стилизация и достъпност на базата данни, както тези в първоначално избрания продукт.

След проучване от моя страна както на различни научни трудове в разпознаваеми бази от данни със затворен достъп като Web of Science, Scopus и други, така и на периодични издания, книги, дисертации, доклади от конференции и други трудове в бази с отворен достъп, не успях да открия библиотека, която да предоставя възможност за изграждане на платформено независими персонализирани визуални елементи.

Цел и задачи на дисертационния труд

Целта на дисертационния труд е да се моделира архитектура за унифицирано създаване на персонализирана визуализация на данни за среди Microsoft Power BI и Tableau с възможност за интеграция със специфичните им приложно-програмни интерфейси.

За постигане на заложената цел е необходимо да се решат следните задачи:

1. Да се изготви сравнителна характеристика на подбраните среди за интерактивно представяне на информация – Microsoft Power BI и Tableau, като се постави фокус не само върху техните особености като инструменти, но и върху наличните видове визуализации, които предоставят;
2. Да се анализират и обобщят особеностите в приложно-програмните интерфейси за разработка на нови визуализации, които всяка една система налага. Подзадачите обхващат:
   - 2.1. Да се отразят специфичните технологии за разработка на персонализирани изгледи, които Microsoft Power BI и Tableau поддържат, като се посочат предимствата и недостатъците им;
   - 2.2. Да се упоменат стъпките по конфигуриране на компоненти и съответните зависимости от външни библиотеки и да се посочат разликите в конфигурационните файлове, съдържащи метаданните за вида на визуализацията, версията и лицензa за използване;
   - 2.3. Да се изведат основните общи за двете среди интерфейси и константи, предоставящи успешната инициализация и работа на компонентите.

3. Да се моделира архитектура за унифицирано създаване на допълнителни визуализации към Business Intelligence инструментите Microsoft Power BI и Tableau.
4. Да се разработят прототипи на визуализации с помощта на новата архитектура за унифицирано създаване на персонализирана визуализация на данни за платформите Microsoft Power BI и Tableau.

Структура и обем на дисертационния труд

Дисертационният труд се състои от увод, четири глави, заключение, авторска справка за приносите, апробация на резултатите, перспективи за бъдещо развитие, библиография и приложение.

В **първа глава** е разгледана спецификата на визуализацията на данни по отношение на сферата Business Intelligence. Акцентирани са характеристиките на двете базисни BI среди за приложение на модела на архитектура, заложен в дисертационния труд, а именно Microsoft Power BI и Tableau. Посочени са техните същности при оформянето на дигитална справка.

Втора глава дефинира особеностите при работата с приложно-програмните интерфейси на софтуерните продукти Microsoft Power BI и Tableau спрямо стилизацията и достъпа до данните. Създадени са примерни визуализации с всеки един специализиран инструмент.

В **трета глава** е демонстриран разработеният модел на архитектура за унифицирано създаване на персонализирани визуализации, наречен TabWerBI. Поставен е фокус върху абстракциите на начините за достъп до данните и способите за оформяне на стилови изменения в изгледа от страна на потребителя. Чрез прилежащи схеми и клас диаграми са илюстрирани основните интерфейси, класове, обекти и константи, които следва да бъдат използвани при реализацията на нови визуални елементи.

Четвърта глава представя примерно приложение на архитектурата за унифицирано разработване на визуализации за Microsoft Power BI и Tableau посредством изграждането на „Триъгълник на застрахователните претенции“, етикет за данни с единично поле, „Lollipop“ и „Tree“ и е обърнато внимание на възможностите им за работа в двете BI системи.

В **заключението** се покрива изпълнението на формулираните задачи и се анализира постигането на поставената цел. Изведени са възможностите за дообогатяване и развитие на архитектурата в бъдеще, както и са отличени приносите по дисертационния труд.

В **Приложение 1** са включени всички фрагменти от кодове, използвани за илюстриране на реалната реализация на компонентите, разгледани в дисертационния труд.

Глава 1. Уеб базирана визуализация на данни

1.1. Business Intelligence (Бизнес разузнаване)

По дефиниция, Business Intelligence в актуално време представлява процесът, при който дадено предприятие използва алгоритъм и технология за анализ на текущи и исторически данни с цел подобряване на стратегическото вземане на решения и предоставяне на конкурентно предимство благодарение на него [50]. Налични са разнообразни технологии, които обхващат всеки един аспект, характерен за сферата. В

състава на BI се откроява утвърденият процес по извличане, обработка и зареждане на данни, наречен ETL, който се характеризира с приложение на разнородни технологични средства за работа и администриране на неструктурирани данни. В параграф 1.1. от дисертационния труд обстойно са описани задачите, които решава формираният алгоритъм за управление на сурови данни, съхранение и достъпност на релационна и свързана база от данни. Важен аспект на крайния BI продукт е обособяването на принципи при изграждането на динамичен отчет. Две от най-използваните BI среди за визуализация на такъв са Microsoft Power BI и Tableau.

1.2. Microsoft Power BI

Microsoft Power BI е приложение за анализиране и споделяне на бизнес знание посредством разработката на функционални интерактивни отчети. Това е инструмент, реализиран от Microsoft, за трансформация и представяне на данни чрез готови визуализации, които могат да бъдат достъпни от Power BI Service – софтуер като услуга (SaaS) в категорията Power Apps. В допълнение, интерактивните визуални справки могат да бъдат създадени и от работното пространство чрез локалната версия, наречена Power BI Desktop. Всяко формирано табло за управление (dashboard) представлява комбинация от добре изграден модел от данни и подходящото им представяне. Визуализациите, включени в съответния панел, изискват различен тип полета и позиция. Могат да бъдат добавени нови колони и изчисления с помощта на специализиран изразен език за калкулация. Даден dashboard може да комбинира локална и облачно-базирана информация, което осигурява консолидиран изглед, независимо къде се намира [3, 4].

Въпреки разнообразието от готови визуализации, средата позволява добавяне на допълнителни такива. Ако желаният изглед не е намерен в наличния „магазин“ за това, средата предоставя възможност интерактивната справка да работи с добавен персонализиран визуален Power BI компонент, който да се използва от индивидуалния бизнес или цялата Power BI общност [60]. Визуалните елементи на Power BI са налични чрез файлови пакети **`.pbiviz`**, които включват код за достигане до данните и визуалното им изграждане, които системата изисква.

1.3. Tableau

Сферата на Business Intelligence и анализът на данни прилага Tableau като представителна платформа в процеса на вземане на решения, базирана на разнообразни методи за обработка на информация. В екосистемата на Tableau съществуват много подсистеми, всяка от които има специализирано предназначение [31]. Той предлага различни функционалности по отношение на източници на данни, като успешно работи с Excel файлове, релационни бази данни или инструменти от външни облачни услуги. Друг важен аспект на визуализацията на данни е структурата на динамичен отчет. Съществува стриктно разпределение на графиките, комбинацията на визуални компоненти и бизнес историята. Започвайки от гледна точка на една визуализация, тя трябва да бъде съхранена в така наречената „работна тетрадка“. Всяка от тези части на Tableau изисква да има полета за добавяне на колони и редове. В зависимост от типа данни, съответно текст, цяло число, дата и други, различни изгледи биват достъпни в раздела „Show me“.

В Tableau могат да се заредят както готови изгледи, така и външни такива, реализирани от оторизирани разработчици, или елементи, локално добавени спрямо персонализираните изисквания на дадената компания. Новият компонент се внедрява в BI

средата аналогично на уеб приложение и може да притежава конфигурационен модален прозорец, с помощта на който потребителят да приложи определени стилови настройки [38, 39, 42]. За целите на правилната интеграция на ново външно разширение следва да се зареди манифест файл с разширение `.trex`, за да се достъпят името, номера на версията, лиценза и съставните файлове на визуализацията.

Глава 2. Изграждане на персонализирани решения за представяне на информация

Разгледаните BI инструменти Microsoft Power BI и Tableau разполагат със самостоятелни приложно-програмни интерфейси (ППИ) за изграждане на допълнителни корпоративни визуализации. Тези колекции от класове, методи, интерфейси и константи, подпомагат разработчиците по отношение на достъпа до данни, интегрирани в базите на съответните среди, както и до конфигурационните техники за индивидуална потребителска настройка. В настоящата глава ще се постави фокус върху възможностите за реализация на нови визуализации, зависимостите и ограниченията, които всяко ППИ налага.

2.1. Power BI Visual API

Приложно-програмният интерфейс към софтуерния продукт Power BI спомага разработването на персонализирани визуализации, които да функционират съгласувано с интегрираната база от данни в инструмента, както и да комуникират с останалите графики и панели. Визуалните елементи на Power BI са налични във файлови пакети `.pbiviz`. Приложението на персонализираната визуализация, интегрирана в Power BI, формира начина, по който средата обработва комуникацията между потребителя, визуалното изображение и хоста. Действия и последващи актуализации в Power BI могат да бъдат инициирани ръчно или автоматично [66-68]. Типове обновявания включват взаимодействия от различен характер, подробно описани в дисертационния труд. Последователността от действия между потребителя, визуализацията и ППИ, зависи изцяло от спецификата на самото персонализирано разширение, обема от данни, който ще бъде обработван, и възможностите, които то ще предостави на своите ползватели [12, 63, 67, 80, 81].

Съществуват специализирани методи и класове в ППИ, които следва да бъдат използвани при утвърждаването на вида достъп до данните. Специфика се откроява в това, че при Power BI се налага конкретно картографиране на данните спрямо изгледа на крайната визуализация не само при достъпа до самата информация, но и при моделиране на конфигурационните файлове, базисни за разширението. Визуализацията, реализирана чрез ППИ на Power BI, притежава достъп до опциите на средата, касаещи потребителски селекции от панел за форматиране.

2.1.1. Технологии за реализация, зависимости и ограничения

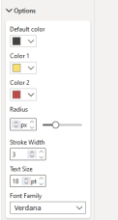
По подразбиране разработката на персонализирани разширения с помощта на ППИ става чрез TypeScript, тъй като при инсталацията на зависимостите, файловата структура се оформя като `.ts` ориентиран проект. По отношение на външни услуги и библиотеки, ППИ поддържа добра съгласуваност с всички JavaScript базирани рамки и техните особености могат да бъдат свободно приложени. По-детайлен обзор на технологиите е заложен в подпараграф 2.1.3 от дисертационния труд.

2.1.2. Примерна визуализация, реализирана с помощта на Power BI Visual API

С оглед представянето на спецификите при употребата на ППИ за Power BI, ще се разгледа разработката и внедряването на персонализиран визуален елемент Lollipop. Той обикновено е сравнение между категории по малко по-атрактивен начин. Важно е да се посочи конфигурационният компонент, който е първият основен файл, включващ множество мета информация, а именно **`pbiviz.json`**. Освен него, подходящо конфигуриран е и **`capabilities.json`**, който трябва да бъде коригиран според съответните изисквания на визуалния елемент. Измененията включват подходящо картографиране на данните, стиловите модификации, които ще бъдат приложени, и видът на полетата, които ще се визуализират в средата. Самият графичен елемент бива изчертан чрез D3.JS библиотека за изграждане на визуални диаграми и графики. Главният метод, предоставен от съответния ППИ при опресняването на състоянието, е демонстриран чрез Код 8 в Приложение 1 на дисертационния труд.

Крайният резултат е цветна диаграма Lollipop, която получава три полета: категория, стойности и целева информация, показана от дясната страна на графиката. Визуализацията е илюстрирана на Фигура 1. Тя взаимодейства с другите двустранно и може да бъде персонализирана от индивидуално подготвения панел за форматиране (виж Фигура 2).

Фигура 1. Изглед от персоналната визуализация Lollipop в среда Power BI



Фигура 2. Форматиращ панел в среда Power BI

2.2. Tableau Extensions API

Приложно-програмният интерфейс за създаване на персонализирани разширения за Tableau позволява на разработчиците да реализират уеб приложения, които могат да взаимодействат и комуникират със средата Tableau [53, 55-58]. Поради спецификата на софтуерния продукт, както е описано в Глава 1, от съществено значение е да се разграничи степента на детайлност на дадената визуализация, тъй като ако при разработката се оформи досег до съдържание на цялото табло за управление, то разширението ще демонстрира контакт с всички елементи в него, включително работните листове, маркировките, филтрите, параметрите и източниците на данни. В допълнение, обръща се внимание на пространствата от имена на ППИ, представляващи контейнери, които съдържат класове и методи за комуникация с компонентите на Tableau. Именуваният контейнер от най-високо ниво специфицира глобалната дефиниция на средата, в която ще се внедри разширението, а именно **`tableau`**. Съществуват още няколко именувани колекции с тясно приложение, които са набор от отговорности и списъци, управляващи функционалността на визуалния компонент.

По отношение на разработката на персонализирано разширение, трябва да се отдаде значение на това, че ППИ е пригоден да работи с чист JavaScript или TypeScript като средства за изграждане на функционалността на визуализацията. Въпреки че ППИ се оформя като JavaScript библиотека, средата Tableau предоставя и типизирани дефиниции,

благодарение на които разработчиците могат да се възползват от предимствата на анотациите и приложението на обектно-ориентираните концепции.

В случай на необходимост от оформяне на персонализирана стилова настройка за допълнителния визуален елемент, при ППИ на Tableau е предвидено реализирането на външен конфигурационен прозорец. Той е своеобразно допълнително уеб приложение, което се свързва с методите за опресняване на състоянието на елементите в основната визуализация, имайки идентичен достъп до данните. За настройката на подобна функционалност следва да се зададе **`callback`** функция, когато разширението бива инициализирано при таблото за управление [36-38, 60]. Функцията за обратно извикване създава опция за конфигурация, позволяваща отварянето на изскачащ прозорец (модален диалогов екран) за съответното разширение. Възможно е да бъде отразена специфична особеност за моментално зареждане при стартиране на визуализацията или при поискване.

2.2.1. Технологии за реализация, зависимости и ограничения

Разширението, пригодено за внедряване в Tableau, може да работи в комбинация с ППИ технологии за изграждане на функционалностите, базирани на JavaScript. Подбрана е и разработка чрез TypeScript, която налага употребата на прекомпилиране и пакетиране на функционалния код с разширение **`.ts`** до синтезиран **`.js`** изглед. Това, от своя страна, води до използването на няколко зависимости като Webpack [26] и JQuery, които имат отношение към изходния код на разширението. Подробното описание на техните особености и предназначение е в подпараграф 2.2.3 от дисертационния труд.

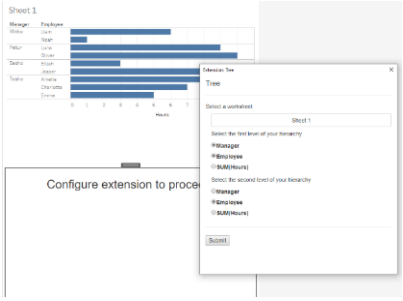
2.2.2. Примерна визуализация, реализирана с помощта на Tableau Extensions API

За да се онагледи примерна визуализация, разработена като персонализирано разширение за Tableau, ще се разгледат стъпките по оформяне на графичен елемент „Tree“. Такъв тип диаграма се отнася до йерархично представяне на набор от данни, когато възлите са връзки тип родител–дете, а не са таблични. Базата данни за визуализация е в **`.xls`** формат и е интегрирана в Tableau Public. Необходимо е наличието на поне един активен графичен елемент в таблото за управление, за да се зареди новото разширение. В секцията Dashborard от потребителския интерфейс на Tableau Public е отразена връзка за добавяне на визуален елемент с достъп до манифест **`.trex`** файл от локален компютър.

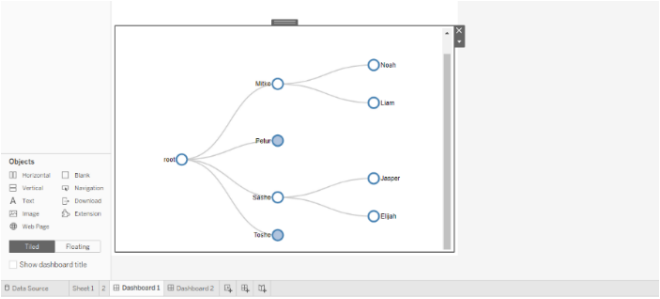
При инициализирането на изгледа се извежда модален прозорец с настройки. За реализиране на конфигурацията се борави с допълнителен HTML, който ще зареди диалогов екран за достъп до определени данни от работната книга. Също така, в ППИ е специфициран метод за затваряне на модалния прозорец и връщане на обновеното състояние на разширението. На Фигура 3 е демонстрирано как разширението се зарежда в таблото за управление със заглавие, идващо от респективния HTML файл. Въпреки успешната инициализация, тя не е обобщение на данни, тъй като първо трябва да се вземат предвид някои стъпки за конфигуриране. Потребителят следва да избере всяко от йерархичните нива от съответния работен лист [56]. Това е индивидуален подход при даденото новодобавено разширение и не е наложителна практика при всички внедрени графични елементи. Крайният резултат от визуализацията може да бъде видян на Фигура 4.

Процесът по разработване на персонализирана диаграма обхваща почти същите стъпки, както при създаването на уеб компонент. Разлика може да се намери във функциите,

които са свързани с Tableau ППИ за изграждане на разширения и осигуряване на коректната инициализация в съответната софтуерна среда.



Фигура 3. Внедрено разширение с диалогов прозорец в Tableau



Фигура 4. Изглед от персоналната визуализация Tree в среда Tableau

Глава 3. Модел на архитектура за унифицирано създаване на допълнителни визуализации към BI инструменти

В настоящата глава ще се разгледат етапите по оформяне на модел на архитектура за унифицирано създаване на допълнителни визуализации към BI инструментите Microsoft Power BI и Tableau. Ще се проследи реализацията на уеднаквяване на зависимостите от външни услуги чрез дефинирането на конфигурационни шаблони за наследяване, както и ще се специфицират абстракции в модела за достъп до данните в базата, цялостната промяна във визуализацията и способите за нейното стилизиране, обобщени в архитектурата TabWerBI.

3.1. Технологии за разработка, зависимости и работен процес

С оглед на оформянето на максимално уеднаквени технологични особености, ще се представят подбраните способи за разработка на модела на архитектурата TabWerBI и зависимостите, които трябва да се предвидят при неговата употреба. При стартирането на нов проект, разработчикът първоначално подготвя както инициализацията на архитектура за унифицирано създаване на разширение, така и набора от външни услуги, който ще бъде посочен в следващия подпараграф.

3.1.1. Предварителна подготовка

Node.js и TypeScript

Двете BI системи изискват визуализациите им да бъдат заредени като всяко едно уеб базирано приложение и трябва да се дефинира необходимостта от инсталиране на Node.js. Подробно е описано значението на услугата за всеки един от подходите за разработка на персонални изгледи спрямо съответните ППИ. Тъй като TabWerBI ги обединява, следва Node.js да бъде предварително инсталирана въз основа на успешната инициализация на новата визуализация.

Имайки предвид това, че Power BI Visual API по подразбиране използва TypeScript, а Tableau предоставя възможност за типизиран подход на разработка, TypeScript е подбран и за базово средство при TabWerBI. Избирайки този програмен език, трябва да се обърне внимание на необходимостта от `tsconfig.json` модификация, която да бъде уеднаквена за целите на унифицираното създаване на нови изгледи. Именно поради тази причина е разработен шаблон, чието наследяване в главната директория на нова визуализация прави

това възможно. Употребата се отразява като наследяване на специфичния шаблон в **`tsconfig.json`** в чрез **`"extends":"./node_modules/tabwerbi/tsconfigtableau.json"`** или **`"extends":"./node_modules/tabwerbi/tsconfigpbi.json"`** в зависимост от средата на внедряване.

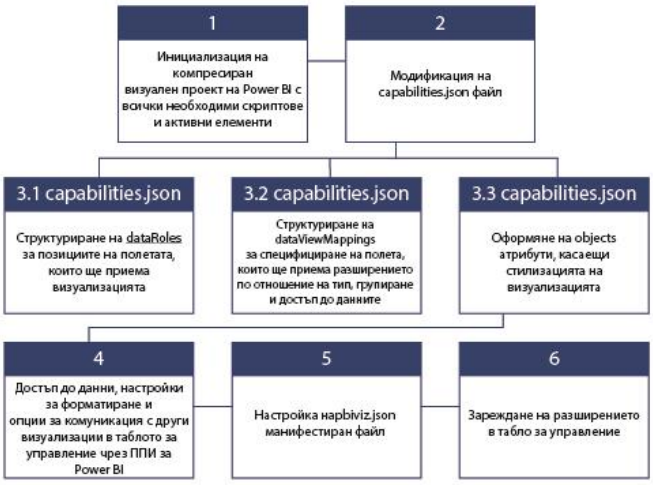
В допълнение, в подпараграф 3.1.1. от дисертационния труд обстойно е засегната важността от употребата на пакета **`@types/jquery`**, наличието на **`webpack-cli & webpack`**, **`ts-loader`** и **`Semistandard`**, които имат тясно приложение при реализирането на функционалната логика за новата визуализация.

3.1.2. Power BI работен процес

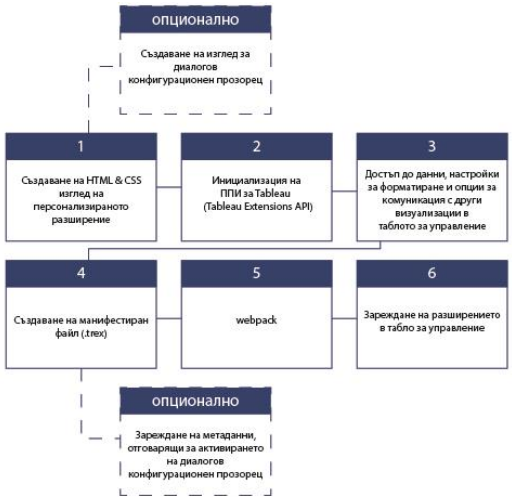
По отношение на работния процес при създаване на допълнителни визуални графични компоненти, трябва да се разгледа алгоритъмът за изграждане на визуализация чрез ППИ на Power BI. Както е показано на Фигура 5, той се състои от шест основни стъпки. Чрез командна функция, по-подробно описана в Глава 2, се инициализира компресираният визуален проект на Power BI. Това оформя цялостен проект, готов за внедряване в съответната среда. След успешна реализация се модифицира манифест файл, идващ от инициализацията на ППИ.

3.1.3. Tableau работен процес

Разработката на проект чрез ППИ на Tableau стартира с направата на визуалната концепция на графичния елемент. Преминава се през няколко базови за средата стъпки, подробно описани в подпараграф 3.1.3. от дисертационния труд. Наличието на по-комплексна функционалност в допълнителните конфигурационни прозорци изисква те да бъдат специфицирани и в **`webpack.config.js`** като модули за прекомпилиране и пакетиране от TypeScript в четим JavaScript код. Чрез Фигура 6 са илюстрирани стъпките, наложителни за формирането и зареждането на ново визуално приложение в дадената BI среда.



Фигура 5. Алгоритъм за разработване и зареждане на визуализация чрез ППИ на Power BI

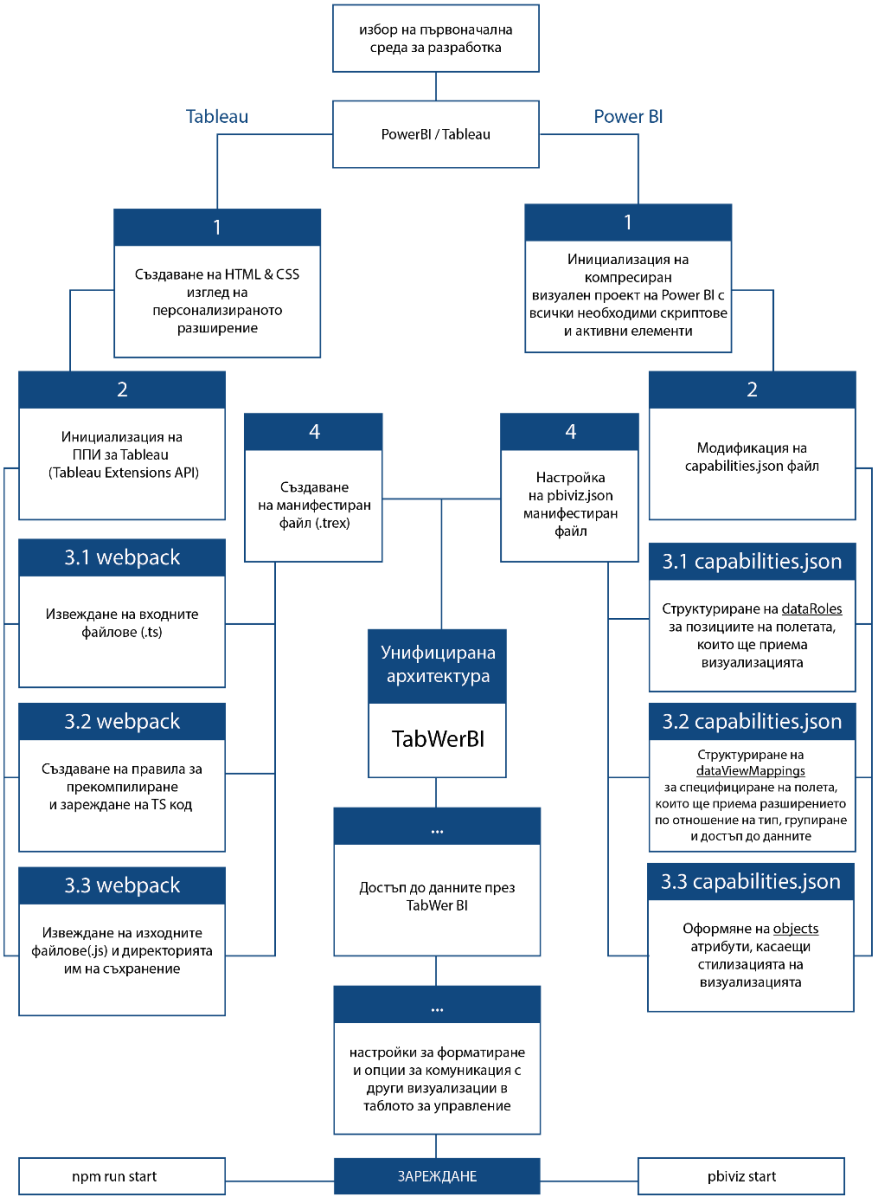


Фигура 6. Алгоритъм за разработване и зареждане на визуализация чрез ППИ на Tableau

3.1.4. TabWerBI работен процес

Спрямо формирането на TabWerBI работен процес е изведен комбиниращ алгоритъм, който е основополагащ за функционирането на модела на разработената

архитектура. В схемата на Фигура 7 са представени четирите базови и специфични за средата конфигурации. Те включват инициализация на разширението, достъп до базата данни и генериране на специализиран манифест файл, съгласно персоналните за системата изисквания. Илюстрирани са три стъпки за зареждане и употреба на TabWerBI, които отразяват инсталирането на самата архитектура, включването на референции към изискуемото пространство за достъп до данните и базовата стилизация. Освен това, включени са и двете персонализирани команди за стартиране на визуализацията в процеса на разработка, които изцяло функционират с архитектурата.



Фигура 7. Схема на алгоритъм за разработване и зареждане на персонализирана визуализация чрез TabWerBI

3.2. Абстракция на модела за достъп до данните в базата

Независимо от избора на първоначален инструмент, преминавайки постъпково през посочените в предходния подпараграф етапи, може да се достигне до работа с

новосъздадената архитектура TabWerBI. Тя бива инсталирана като стандартна външна услуга и включена в базовия файл, съдържащ логиката на разширението. Разработени са шест метода за достъп до информация, необходима за визуализация, която работи с единично поле, графичен елемент, презентиращ категорийни данни, и разширение в табличен вид. Те са обобщени в две именувани пространства **`TabWerBITableau`** и **`TabWerBIPowerBI`**. Достъпът до данните за трите типа визуализации бива уеднаквен с помощта на нова инстанция на методите **`getSingleData()`**, **`getCategoricalData()`** и **`getTableData()`**. При промяна на средата, в която съответният персонализиран продукт бива внедрен, следва да се смени името на именуваното пространство и изискуемите аргументи в инстанцирания обект.

3.2.1. Достъп до данните чрез ППИ на Power BI

Създаването на обобщен достъп до данните при оформянето на категориен изглед, единична визуализация и такава, подходяща за изграждането на таблица, се подпомага чрез реализирането на три специфични функции, чрез които данните да се разпределят по акуратния начин. За целта са избрани методи **`getSingleData()`**, **`getCategoricalData()`** и **`getTableData()`**, чийто алгоритъм на работа ще бъде обстойно прегледан в следващите редове.

CustomSingleView

По отношение на наличието на визуализация, работеща с единствено поле, се пристъпва към картографиране на едно такова за мярка, връщащо определена избрана агрегация. В конкретиката на средата Power BI, трябва да се отрази ролята на данните в **`capabilities.json`**. При дефинирането на метод за работа с единични данни се използва обект **`DataView`**. По този начин става възможно обработването на свойството **`single.value`** на обекта **`dataView`**. В дисертационния труд подробно са описани постъпковите валидации, които са необходими за достигането до реалната реализация на метод **`getSingleData()`** за достъп до самостоятелно поле от базата данни. Методът е демонстриран чрез Код 19 в Приложение 1 от дисертационния труд.

CustomCategoricalView

За изготвянето на визуализации, които наподобяват осева диаграма, разпределяща едно поле по оста X и стойност по Y, следва да се подходи първоначално с промяна в **`capabilities.json`**. Чрез категорийното картографиране на данни се получават независими групи или такива, разпределени по категории. След като успешно е модифициран конфигурационният файл, се обработват полетата от данни чрез подходящ метод, а именно **`getCategoricalData()`**, който може да бъде разгледан чрез фрагмент от Код 20.

CustomTableMatrix

Матричният изглед по отношение на персонализираните презентативни елементи дава възможност за направа на многоприложни визуализации. Такъв тип достъп до данни формира колони, редове и стойности в подходящ за крайния потребител вид. Освен това, матричното картографиране представя редовете йерархично. Модификацията на полетата отново се отразява в **`capabilities.json`**. С оглед на това да се оформи структурата за достъп до данни, трябва да се разработи метод **`getTableData()`**. Негова логика е представена подробно в подпараграф 3.2.1. от дисертационния труд и Код 22.

3.2.2. Достъп до данните чрез ППИ на Tableau

Предвид факта, че няма точно образувани класове, отговарящи за оформянето на трите конкретни типа визуализации, подобно на предходно описания ППИ, се налага специфицирането на отделен способ за достъп до данните. Всяко персонализирано разширение, разработено чрез ППИ на Tableau, наследява основния интерфейс `Extensions`. Базисната функция за всяка визуализация е `initializeAsync()`. За картографирането на данните за среда Tableau, в настоящия подпараграф са разпределени в три категории, съответно `CustomSingleView` за единични визуализации, `CustomCategoricalView` за категорийни диаграми и графики, и `CustomTableMatrix` при оформяне на таблици/матрици.

CustomSingleView

Спрямо изискването `CustomSingleView` да оформя единично изчисляемо поле, трябва да се отрази начин за извличане на данните от текущия работен лист в таблото за управление на Tableau, касаещо изграждането на визуализация с една стойност. Чрез фрагмент от Код 23 в Приложение 1 на дисертационния труд е демонстрирана асинхронната функция `getSingleData()`, която достъпва обобщени данни от работен лист и изчислява броя на отделните стойности в първата колона.

CustomCategoricalView

Базирайки се на изискването визуализация от типа на линейните диаграми да притежава категориен достъп до информацията, се налага създаването на отделна функция в Tableau, която да оформя данните по подходящ начин, съответно по ос `X` и `Y`. Код 24 съдържа реализацията на метод `getCategoricalData()`. Сходно на предходния подход, оформя се асинхронно извличане на данни с помощта на `getSummaryDataAsync()`, като получените такива се запазват в `sumdata`, и чрез `sumdata.data.map(row => ({ ... }))` се картографира всеки ред данни от обобщените в обект с две свойства: `x` и `y`.

CustomTableMatrix

По отношение на `CustomTableMatrix` трябва да се специфицира начин за извличане на данните от текущия работен лист в таблото за управление на Tableau, касаещо оформянето на матричен изглед. Методът `getSummaryDataAsync()` връща обобщените данни. След успешното им извличане, те ще бъдат обработени чрез `.then((sumdata) => { ... })`. При работата с данните е необходимо те да се разпределят по редове и колони, и за целта се създава нов обект `dataTable`. Той съдържа `columns: sumdata.columns.map(...)` и `data: sumdata.data.map(...)`, подробно разгледани в дисертационния труд и в Код 25.

3.2.3. Абстракция на модела в TabWerBI

Архитектурата за унифицирано създаване на персонализирани изгледи притежава достъп до двете основни ППИ за разработване на разширения за Tableau и Power BI. За реализирането на абстракция на модела за достъп до данните, в основната директория е създаден TypeScript файл, който включва зависимостите от съответните работни библиотеки. Обособени са две именувани пространства – `namespace TabWerBITableau {…}` и `namespace TabWerBIPowerBI {…}`, които ще обобщават методите и класовете за двете системи. Предвид достъпването на специализираните методи и класове при обработката на данните във визуализациите, трябва да се добави релационна свързаност с именуваното пространство. Това става възможно чрез един от следните редове в зависимост от средата за интеграция: `import {TabWerBITableau} from "tabwerbi/tabwerbi-pack/DataAccess/TableauTableDataAccess"` или `import {TabWerBIPowerBI} from "tabwerbi/tabwerbi-pack/`

`DataAccess/PowerBITable DataAccess"`. Фигура 25 на страница 75 от дисертационния труд демонстрира чрез А, Б и В сечението между двата ППИ, както следва:

А → Достъп до табличен изглед

TabWerBITableau

За извличане на данните в Tableau се употребява интерфейс `GetTableData`, който декларира свойство `worksheet`. В интерфейса се отразява и метод, който носи името `getTableData()`. Реализацията е описана подробно в дисертацията в подпараграф 3.2.3. С оглед на приложението на персонализираната архитектура, по отношение на формирането на матрица или таблица, трябва да се имплементира интерфейсът `GetTableData` в базисния клас. При успешната инсталация и включване на TabWerBI в нов проект е необходимо да бъде създаден един нов обект от този клас `GetTableData`, за да може да се оформи логиката за извличане на данните и поставянето им на желаните позиции.

TabWerBIPowerBI

При оформянето на алгоритъма за достъп до таблични данни за среда на визуализация Power BI, се дефинира интерфейс с метод `getTableData()`. Декларираният метод разполага със структура `columnHeaders: string[]` и `rows: any[][]`, която трябва да бъде върната като резултат. Обзор на свойствата, които притежава, е направен в съответния подпараграф от дисертационния труд. За използването на `GetTableData` в основния проект за среда Power BI, класът трябва да имплементира интерфейс `GetTableData`.

Б → Достъп до единично поле

TabWerBITableau

Подобно на предходно описания модел, за коректната имплементация на класа, който ще обработи единичното поле за среда Tableau, да се изгради интерфейс със свойство `worksheet` и респективния метод. В настоящия случай това е `getSingleData()`, който трябва да върне `Promise` при тип на данните `any`. Класът `GetSingleData` имплементира `GetSingleData` интерфейс, както може да бъде разгледано в Код 31. За да работи коректно е достатъчно да се създаде променлива, която да извиква нов обект от именуваното пространство `TabWerBITableau`, използващо класа `GetSingleData`.

TabWerBIPowerBI

За приложението на `getSingleData()` метод в среда Power BI се реализират два интерфейса `GetSingleDataResult` (`(1)`, Код 32) и `GetSingleData` (`(2)`, Код 32). При `GetSingleDataResult` се посочва единично свойство, което е именувано `dataValue` и следва да бъде от тип `string`. За основния интерфейс `GetSingleData` се извежда метод `getSingleData` с параметър `dataView`, който е необходимо да върне обект, отговарящ на интерфейса `GetSingleDataResult`. За точната употреба на този метод се изисква променлива, която да инициализира нов обект от клас `GetSingleData` и да извика полето, което потребителят е поставил в съответния компонент от потребителския интерфейс на средата Power BI.

В → Достъп до категория

Категорийната информация се извлича за създаването на визуализации, които обикновено имат оси – `X` и `Y`, базирани на различни типове данни.

TabWerBITableau

Отразяването на достъп до данните през ППИ на Tableau се подпомага чрез интерфейс `GetCategoricalData`, който има свойство `worksheet`, сходно на всички споменати

до тук, и метод **getCategoricalData()**, отговарящ за извличането на категорийните данни. Специфичното при този асинхронен метод е, че очаква да върне масив от обекти от описаните типове в Код 34 от Приложение 1. Начинът, по който работи методът **getCategoricalData()**, е подробно представен в подпараграф 3.2.2 и поради тази причина в Код 35 е само имплементацията на базовия клас. За да се възползва от абстракцията на модела за достъп до категорийни данни в среда Tableau, разработчикът трябва да създаде нов обект от клас **GetCategoricalData** и аналогично на останалите – да подаде необходимия аргумент.

TabWerBIPowerBI

Коректната имплементация на TabWerBI за работа с ППИ на Power BI налага да се създаде интерфейс **GetCategoricalData**. Той има единствен метод **getCategoricalData()**, който приема като атрибут **categoricalDataView**, очакващ обект от тип **DataViewCategorical**. Чрез Код 36 е интерфейсът **GetCategoricalData**. Алгоритъмът на извличане на данните за **getCategoricalData** изцяло съвпада с този, разгледан в подпараграф 3.2.1. Употребата на абстракцията е аналогична с предходно описаните.

Въпреки съществуването на голям набор от картографиране на данни за среди Power BI и Tableau, в модела на архитектурата за унифицирано създаване на персонализирани разширения се покриват трите основни вида достъп до информация.

3.3. Абстракция на модела за извеждане на промяна при форматиране на персоналното визуално разширение

Друг важен аспект, който следва да се разгледа, засяга модификацията на стилово оформление на дадено персонализирано разширение. Двете среди за визуализация на данни притежават разлики при конфигурацията на форматирането на изгледа, което е предпоставка за особености при реализирането на абстракцията.

3.3.1. Стилизация на персоналното разширение чрез ППИ на Power BI

ППИ на Power BI изисква за специфицирането на достъп до форматиране на визуализацията да се създаде нов обект от клас **VisualSettings**. Той наследява **DataObjectsParser**. За точната му работа трябва да се заредят интерфейсите, работещи с „Format your visual“ секцията на средата. Този основен интерфейс поддържа няколко функции и статични обекти (виж подпараграф 3.3.1. от дисертационния труд). Във връзка с прилагането на дадена стилистика в персонализираното разширение, разработено с ППИ на Power BI, е подбрано да се демонстрира начинът, по който се променя цвят на фон, шрифт и размер на текста спрямо потребителска селекция. Предварително са изведени обектите, които могат да се обособят в стиловото изграждане като **backgroundColor**, **headerColor**, **fontSize** и **fontFamily**. Първоначалното им визуализиране в панела за корекция е с помощта на обекти в **capabilities.json**. Примерно разпределение на всички реализирани стилизации в архитектурата за унифицирано създаване на персонализирани разширения е посочено в Код 38.

Принципът на работа на персоналната визуализация при нейното стартиране, разгледан подробно в Глава 2, параграф 2.1, задължава дефинирането на всички промени в метода **update()**. В следствие на това, той се специфицира в посочения метод **this.settings** за инициализацията на форматиращите възможности. Чрез фрагментите от Код 41 и Код 42 може да се отбележи по какъв начин се достъпват **VisualUpdateOptions**, **parseSettings(1)** и

`enumerateObjectInstances(2)`, отговарящи за цялостната промяна на размера на визуализацията, данните, с които работи, нейният тип и други.

3.3.2. Стилизация на персоналното разширение чрез ППИ на Tableau

Особеностите при оформянето на достъп до стилизацията на персонализираното разширение покриват изискването да бъде установен изглед за нов модален прозорец и да бъдат включени функционални настройки за промяна на форматирането. Това, от своя страна, означава, че базисният клас на разширението трябва да притежава метод `configure()`, който включва следните зависимости `dialogOptions`, `ui.displayDialogAsync()`, `settings.set()` и `settings.saveAsync()`. Необходимо е да се отрази размерът на модалния прозорец чрез `dialogOptions`. Основният метод, който е специфичен при зареждането на диалогов такъв, е `displayDialogAsync()`. Функцията `initializeDialogAsync()` е входната точка за визуализация, която се изпълнява в даден конфигурационен екран, създаден от именуваното пространство `UI`. В подпараграф 3.3.2. подробно е разгледан способът за формиране на модален изглед. При реализацията на модела на архитектурата са предвидени специфични `id` атрибути, които разработчикът трябва да добави в своята уеб визия, за да бъде възможно изпращането на данни от диалоговия прозорец към базисното разширение.

При изграждането на конкретен примерен диалогов прозорец с определен дизайн, наложително е да се премине през няколко основни стъпки, които са демонстрирани чрез фрагмент от Код 43. Той включва и реализация на метод `closeDialog()` за затваряне на модалния екран след направен избор. В базовия клас на допълнителното разширение се достъпва метод `configure()` (виж Код 45 от Приложение 1 на дисертационния труд), който съдържа достъп до `.html` на диалоговия екран и логиката, която да комуникира с него. Предвид факта, че разработчикът може да измени наименованието на `.html` файла, както и размерите на изскачащия модален прозорец, би следвало в архитектурата тези компоненти да не се унифицират.

3.3.3. Абстракция на модела в TabWerBI

Формирането на абстрактен модел за извеждане на промяна при форматиране на персоналното визуално разширение цели да унифицира начина, по който разработчикът ще достъпи ППИ алгоритъм с тази насоченост. Поради разликите в двете среди и принципът им на работа не могат да се прескочат конфигурационните особености.

TabWerBITableau

Абстракцията на стилизация в среда Tableau предопределя създаването на интерфейс, който да специфицира типа на полетата, които се очаква да бъдат тези, отразяващи определените стилове за форматиране. В конкретния случай те са низове и затова всяко едно приема тип `string`. Посредством клас `VisualSettingDialogTableau` се извежда абстракция за формирането на диалогов прозорец към ППИ на Tableau. Тъй като `displayDialogAsync()` налага като аргументи достъп до `.html` изглед на форматиращия панел и съответен негов размер при инициализация, се формира асинхронен метод, който изисква аргументи `popupUrl` и `DialogSize` при извикването му.

В подпараграф 3.3.3. се описват подробно видовете стилови настройки, които методът ще трябва да обработи, и се записват в съответната променлива. Основната абстракция на панела за стилизация борави с `initializeDialogAsync()`, чиято функционална логика се намира в допълнителния конфигурационен файл към базовото разширение. С оглед на това, че средата различава основната визуализация от тази, създадена за

форматиращи настройки, следва да се именува подходящо методът, отговарящ за инициализацията на модален прозорец в TabWerBI. При необходимост от персонализирани настройки, разработчикът използва `DialogLogic()` от именуваното пространство за достъп до `initializeDialogAsync()` за ППИ на Tableau.

В модела на архитектурата TabWerBI за среда Tableau са предвидени да се обработят `color1`, `color2`, `fontFamily` и `textSize`, които трябва да удовлетворяват необходимите стилови модификации при формирането на индивидуална корпоративна идентичност за персонализираното разширение.

TabWerBIPowerBI

По отношение на работата с ППИ за Power BI, в именуваното пространство `TabWerBIPowerBI` са добавени първостепенните зависимости `import { dataViewObjectsParser } from "powerbi-visuals-utils-dataviewutils";` и `import DataViewObjectsParser = dataViewObjectsParser.DataViewObjectsParser;`, за да се достъпи `VisualSettings`, който наследява `DataViewObjectsParser`. Оформени са четирите форматиращи настройки, като са изведени подходящи имена за отразяване на `color1`, `color2`, `textSize` и `fontFamily`. За дефиниране на абстракцията за среда на внедряване на графичния елемент Power BI, са създадени четири метода, които служат за достъп до съответните стилови настройки. Всеки един от тях има единствен параметър, който извиква тип `VisualSettings`. По този начин, в основната функционалност на базовия файл на изгледа може да се включи класът `VisualSettings` от `TabWerBIPowerBI`. Методите са подробно анализирани в подпараграф 3.3.3.

Глава 4. Персонализирани визуализации, базирани на архитектурата TabWerBI

В настоящата глава ще бъде разгледан етапът по разработка на персонализирано разширение посредством TabWerBI и TypeScript, като за примерни приложения са избрани визуален компонент, насочен към представяне на застрахователните претенции, и единично поле за демонстриране на етикет за данни. В допълнение са илюстрирани още две специализирани визуализации, внедрени в Power BI и Tableau.

4.1. Същност и приложение на „Триъгълник на застрахователните претенции“

Визуализациите за застрахователните претенции са сред едни най-важните и специфични форми на докладване в застрахователната индустрия. Съответните експерти имат за цел да използват своеобразни триъгълници на исковете, които демонстрират натрупаните изплатени обезщетения в годините до края на периода по дадена полица. Обстоен преглед на спецификата на визуализацията и нейното приложение е направен в началото на Глава 4.

4.1.1. Извеждане на необходимите зависимости и конфигурационни файлове

Първоначално избрана среда Microsoft Power BI (Power BI First)

При стартиране на разработката от среда Power BI следва да се приложат базовите стъпки при работния процес в TabWerBI от Глава 3. Инициализира се нов пакет на ППИ на Power BI, който ще бъде прочетен от софтуерния продукт при внедряването на инструмента, преминава се през специфична настройка както на `pbiviz.json` манифест

файл, така и на `capabilities.json` за отразяване на възможностите за достъп до полета и стилови настройки.

Първоначално избрана среда Tableau (Tableau First)

При стартиране на изграждането от среда Tableau или след миграцията от Power BI в настоящата среда, трябва да се премине през специфичните елементи за системата (виж Глава 3). В стандартен `html` файл е необходимо да се опише структурата на разширението, като се включат базисни тагове с идентификатор и класове, които в последствие се използват при достъп и обработка на данните. Подробна конфигурация на webpack пакета е демонстрирана в подпараграф 4.1.1. на дисертационния труд. Допълнително е модифициран и конфигурационният файл `.trex`, съхраняващ възможностите на визуализацията за стилизиране чрез съответните мета данни.

4.1.2. Изграждане на основната визия на графичния елемент

Визуализация „Триъгълник на застрахователните претенции“ представлява таблица. В конкретния случай е избрана библиотеката D3.JS за създаване на своеобразната таблична структура, тъй като предоставя набор от функции за обвързване на данни към DOM и прилагане на управлявани трансформации към документа [71-73].

4.1.3. Достъп до данните, необходими за визуализацията

При разработката на визуализации за съответните среди, инициализацията не е уеднаквена в архитектурата и се налага в базовите им методи да се използват специфичните операции от прилежащите ППИ. В Код 56 от Приложение 1 в дисертационния труд са достъпни двата основни метода за зареждането на нов изглед и промяната на неговото състояние, като чрез `(1)` и `(2)` са белязани отделните имплементации.

Спрямо софтуерния продукт за интерактивни справки, в който ще се внедри новата визуализация, е дефинирано подходящо именувано пространство за достъп до разработените методи. То би било активно, ако архитектурата е инсталирана успешно и е включена зависимостта от нея чрез `import {TabWerBITableau} from "tabwerbi/tabwerbi-pack/DataAccess/TableauTableDataAccess"` за среда на интегриране Tableau или `import {TabWerBIPowerBI} from "tabwerbi/tabwerbi-pack/DataAccess/PowerBITableDataAccess"` за PowerBI.

Както е илюстрирано на фрагмент от Код 57, създава се нова инстанция на клас `GetTableData`, който налага два аргумента `this.dataViewMatrix` и `this.target`. Идентична е логиката на оформление при среда на внедряване Tableau. Благодарение на това, че е единна рамката за изграждане на визуалната идентичност D3.JS, достъпът до данните става чрез редовете, посочени в Код 58. Достатъчно е да се подаде като аргумент `this.worksheet` в инстанцията на класа `GetTableData` и да се спази асинхронния механизъм, изискуем от съответната среда, за да се достигнат коректно данните.

4.1.4. Настройки за форматиране на визуалния елемент

Както подробно е описано в Глава 3 от дисертационния труд, ППИ на двете среди се различават освен по формиране на стилови методи и класове, и по това, че при Power BI е наличен директен достъп до панела за конфигурация, докато при Tableau следва да се изгради такъв като диалогов прозорец.

При **Power BI First** подход, трябва да се добавят зависимостта `import VisualSettings = TabWerBIPowerBI.VisualSettings;`, за да се заредят основните методи, служещи за промяна

на стиловите настройки, и още няколко зависимости в началото на базовия файл на визуализацията, като тяхната структура и специфика са изложени в подпараграф 3.3.1. С оглед на коректната функционалност, в **`capabilities.json`** са включени обектите **`backgroundColor`**, **`headerColor`**, **`fontSize`** и **`fontFamily`** (виж Код 59). Методът, обобщаващ изграждането на таблицата, е **`displayTable()`**.

Спрямо **Tableau First** подход, трябва да се създаде **`config.html`** файл, който да съдържа в себе си необходимата структура, за да се симулира конфигурационен прозорец за избор на цветове, шрифт и размер на текста. Това е постигнато чрез логика на манипулация в **`config.ts`** за идентификаторите на **`html`** атрибути. Код 65 представя имплементация на метод за обновяване на настройките, чрез който конкретно **`id`** със съдържанието на персонализираното разширение се изтрива и в последствие се изчертава отново чрез **`displayTable()`**.

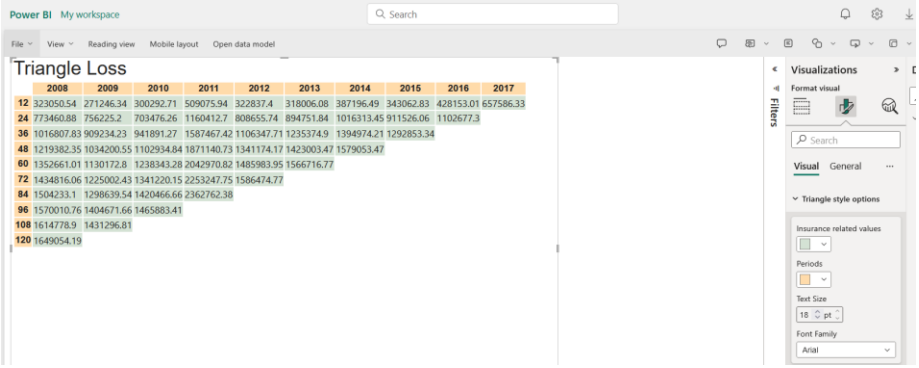
Независимо от подхода за реализация на новата визуализация, следва да се премине през допълнителни стъпки по отношение на стилизацията. Както вече бе споменато, при разработката на триъгълника бе избрана първоначалната среда да бъде Power BI, което налага допълнително да се създават **`config.html`** и **`config.ts`** за внедряването му в Tableau. Освен това, трябва да се обърне внимание на настройките, които са индивидуални при **`webpack.config.js`** и манифест **`trex`** файл, тъй като той служи за средство на достъп до бизнес логиката на разширението в среда Tableau Public.

4.1.5. Внедряване на визуализацията в Power BI и Tableau

След като основното визуално изграждане е изготвено с помощта на D3.JS и TypeScript, всички зависимости от други външни услуги са инсталирани и е добавена архитектурата TabWerBI чрез **`npm install tabwerbi`**, трябва да се специфицират някои базови конфигурации в отделната BI среда. Те са изискуеми за коректното инициализиране на персоналното разширение.

Зареждане на разширението в среда Microsoft Power BI

Стартирането на визуализацията в посочената платформа става посредством **`pbiviz start`**. В дисертационния труд е демонстриран изгледът по подразбиране на средата, а именно триъгълник с примерна база от данни. Информацията е разпределена по редове, колони и клетки, без приложени стилови настройки. Полетата са провлачени в съответните позиции, които са именувани и изведени благодарение на настройката в **`capabilities.json`**. Стилизирането на персонализирания изглед е илюстрирано на Фигура 8. Активиран е „Format visual“ екран, който предоставя възможност за форматиране на визуализациите.

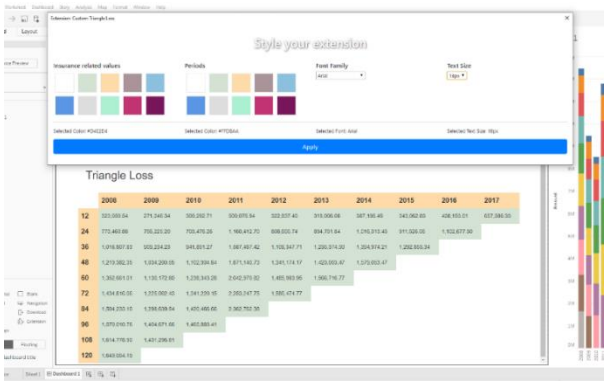


Фигура 8. Изглед на стилизирана визуализация чрез настройки от форматиращ панел

Зареждане на разширението в среда Tableau

При внедряването на визуализацията в другата BI среда трябва да се преминат всички инсталационни зависимости, които са надлежно описани в съответния подпараграф. Тъй като персонализираното разширение ще притежава допълнение за стилизация, то следва да се използва формулираният механизъм за визуализация с модален екран. Достъпваме среда Tableau чрез команден ред `npm run start.`

Изгледът на триъгълника бива отразен по подразбиране без стилови настройки. При избор на опция configure за даденото разширение, може да бъде видян как изглежда конфигурационният диалогов прозорец, който в момента на първоначално зареждане, не поддържа потребителска селекция. При подбор на дадени цветове, шрифт и размер на текста, потребителят вижда своят модален екран с предпочитания като изписване под всяка настройка и след отразяване на бутон `Apply`, диалоговият прозорец се затваря, прилагайки стиловите модификации.



Фигура 9. Изглед на стилизирана визуализация чрез селекция на настройки от диалогов прозорец

4.2. Същност и приложение на „Data Label“

Изграждането на визуализации във вид на етикети за данни илюстрира динамични изгледи, които работят с единични полета. Тяхното предназначение се характеризира с това, че могат да представят информация, която да служи за заглавия, насочващи етикети и други.

4.2.1. Извеждане на необходимите зависимости и конфигурационни файлове

По отношение на оформянето на „Data Label“, стъпките, зависимостите и спецификата на конфигурационните файлове са идентични с тези, които са разгледани в подпараграф 4.1.1. Визуалният елемент изисква формирането на панел с настройки за стилизация, което означава, че трябва да бъдат спазени последователностите на действията при Power BI First и Tableau First, касаещи отразяването на специфичните атрибути в `capabilities.json` и метаданни в `DataLabel.trex` (манифест файл за новата визуализация).

4.2.2. Изграждане на основната визия на графичния елемент

Оформянето на изгледа на персонализираната визуализация е постигнато отново чрез D3.JS библиотека. Специфицирайки основната структура на `svg` елемент, могат да бъдат модифицирани всички негови стилови компоненти. Създаден е динамичен правоъгълник, който се преоразмерява съгласно потребителската настройка за размер и шрифт на текста, която трябва да се достъпи от средата за внедряване.

4.2.3. Достъп до данните, необходими за визуализацията

Осъществяването на унифициран достъп до единичните полета налага да се използва клас **`GetSingleData`**, имплементиращ **`GetSingleData`** интерфейс, който е валиден и за двете именувани пространства. Независимо от формирането им в TabWerBI, за зареждането на стойността, необходима за представяне на етикет за данни с една стойност, по подразбиране се достъпва бройката на уникалните записи при извиканото поле чрез метода **`getSingleData()`**, разгледан по-подробно в дисертационния труд.

4.2.4. Настройки за форматиране на визуалния елемент

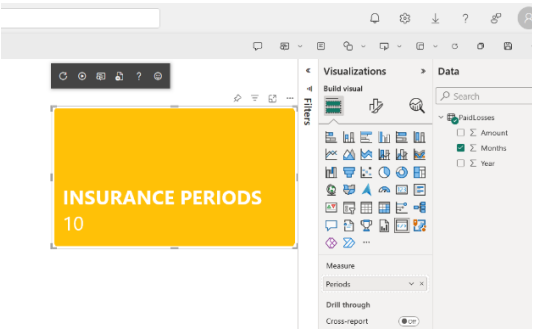
По отношение на изграждането на стилово оформление за персонализираната визуализация на данни, следва да се премине през идентични стъпки на тези при стилизацията на Триъгълника. Разлика се откроява в специфичните елементи, които всеки форматиращ панел ще притежава. При промяната на изгледа, трябва да се вземе предвид, че ще се измени фонът на Data Label елемента, както и поотделно размерът и шрифтът на двата текстови елемента. Това, от своя страна, означава, че съответните стилови елементи могат да бъдат дефинирани като **`backgroundColor`**, **`headerFontSize`**, **`headerFontFamily`**, **`AmountFontSize`** и **`AmountFontFamily`**.

4.2.5. Внедряване на визуализацията в Power BI и Tableau

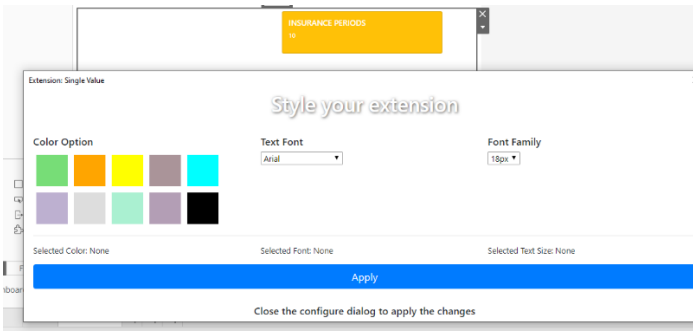
След финализиране на изграждането на визуалното оформление с D3.JS и TypeScript, се покриват сходни стъпки на тези, посочени при внедряването на Триъгълника. TabWerBI аналогично се добавя чрез **`npm install tabwerbi`**.

Зареждане на разширението в среда Microsoft Power BI

Информацията е насочена за използване като единична стойност на дадено поле, което може да бъде от различни типове. За визуализацията на етикета на данни е провлачено полето в съответната позиция, която е именувана и изведена благодарение на настройката в **`capabilities.json`**. Базовият метод за инициализация, чрез който средата Power BI комуникира с предоставения от нея ППИ, е **`update()`**. В тялото му е наличен методът **`displaySingle()`**. Стартирането на визуализацията аналогично се извършва чрез **`pbiviz start`** и е внедрена отново в облачната среда на съответната BI система. Визуализацията в Power BI е илюстрирана на Фигура 10.



Фигура 10. „Data Label“ визуализация в среда Power BI



Фигура 11. „Data Label“ визуализация в среда Tableau

Зареждане на разширението в среда Tableau

Тъй като персонализираният изглед би следвало да може да се модифицира спрямо индивидуалните предпочитания на бизнеса, базовият метод, който трябва да

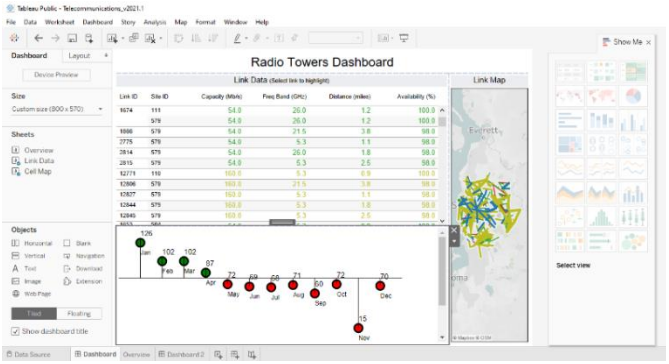
присъства първи в асинхронния `initialize()`, е `initializeAsync({ 'configure': async () => this.configure() }`. В тялото на инициализацията се извиква метод `displaySingle()`, който подобно на внедряването в среда Power BI, служи за обработката на данните, изчертаването на графиката и обособяването на стиловите настройки, съгласно условията на D3.JS. Достъпването до среда Tableau става с команден ред `npm run start`, който аналогично на Триъгълника активира необходимите скриптове. Крайният вече стилизиран елемент в среда Tableau е на Фигура 11.

4.3. Визуализации „Lollipop“ и „Tree“

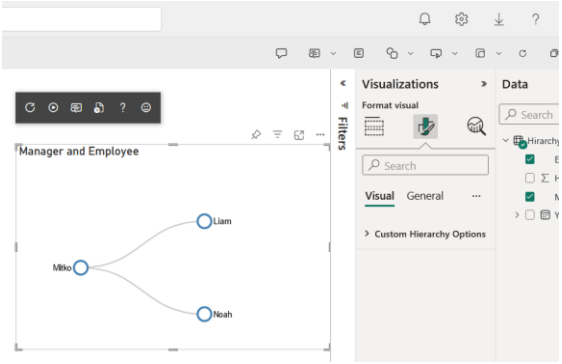
Формирането на процес по разработка на нови визуализации, включващи директна употреба на TabWerBI, до голяма степен би улеснило евентуалната последвала миграция в среда, различна от първоначалната такава. Разгледани са случаите, при които моделът на архитектура бива внедрен във вече изграден изглед на компонент чрез визуализации „Lollipop“ и „Tree“.

На Фигура 12 е демонстрирана визуализация от тип „Lollipop“, внедрена в среда Tableau, която представя данните за наличните абонати на радио честотни кули в област Вашингтон. Справката е готово генерирана и в нея е внедрен персонализираният изглед [86]. Данните във визуалния елемент са в хиляди спрямо 2021 година и предоставят бройката на потребителите, достъпващи услугата на радио кулата, разпределена по месеци. С оглед на получаването на ясно сведение относно достигането на оптималното ниво на потребление, към визуализацията е включена target линия, спрямо която са позиционирани точковите данни. Достъпът до данните е осъществен чрез TabWerBI, като типът на картографирането на данните съответства на визуализация с категориен модел. Използван е метод `getCategoricalData()` и за двете среди на внедряване. Поради особеността, характерна за взаимовръзката на данни при Power BI, следва да се отрази специфичното им разпределение в конфигурационния файл `capabilities.json`.

Стилизацията на визуалния елемент аналогично, както при останалите имплементации, предопределя да се достъпи `const MainBackgroundColor = TabWerBIPowerBI.getMainBackgroundColor(this.settings)` при необходимост от смяна на цвета за среда на внедряване, работеща с именувано пространство `TabWerBIPowerBI`. Моделирането на стиловите настройки за платформа Tableau налага отразяване на нов модален прозорец с опции, отговарящи на `TabWerBITableau`.



Фигура 12. Визуализация „Lollipop“



Фигура 13. Визуализация „Tree“

Фигура 13 илюстрира приложението на дървовидна структура в среда Power BI. „Tree“ визуализация е предназначена да представя йерархично дадена информация, като

разпределя стойностите стъпаловидно спрямо най-високия „клон“ на дървото. Визуалният елемент е реализиран така, че при зареждането на данните да се моделират всички нива на йерархията. В примера се вижда такава в малка компания, като директните подчинени служители са обвързани графично със своя мениджър. С оглед на по-изчистената визия, останалите мениджъри са филтрирани от характерния за това панел в средата Power BI. За достъпа до данните, въпреки че концептуално и визуално се различава от категорийните изгледи, при дървовидната структура може да се употреби клас **`GetCategoricalData`** за картографиране на информацията и при двете среди. Това би станало възможно чрез инициализация на нов обект от настоящо разгледания клас и дефиниране на метод **`getCategoricalData()`** с аргументи **`dataView.categorical`** при платформа Power BI или **`this.worksheet`** за Tableau.

По отношение на стилизацията, следва да се оформи персонален конфигурационен прозорец с настройки, включващи смяна на цвета на точковите данни, размера и шрифта на текстовите етикети. Базирайки се на клас **`VisualSettingDialogTableau`**, се извежда абстракция за формирането на диалогов прозорец към Tableau, който би отразявал селектираните от потребителя настройки. При среда за внедряване Power BI трябва да се достъпи форматиращият панел, генериран от **`capabilities.json`** и клас **`VisualSettings`** от **`TabWerBIPowerBI.`**

Заключение

Изпълнена е основната цел на дисертационния труд – да се разработи модел на архитектура за унифицирано изграждане на персонализирани изгледи за двете разгледани BI системи. Той покрива два модела на абстракция – достъп до данни и стилизация. На база архитектурата са разработени визуализации „Триъгълник на застрахователните претенции“, „Data Label“, „Lollipop“ и „Tree“, които биват внедрени в посочените среди.

Авторска справка на приносите

Резултатите от дисертационния труд могат да бъдат обобщени в следните научно-приложни и приложни приноси:

1. Изведени са както необходимите зависимости и специфики, така и пресечените точки на приложно-програмните интерфейси за Power BI и Tableau, с оглед на оформянето на уеднаквена структура.
2. Създадена е архитектура за унифицирано разработване на персонализирани визуализации TabWerBI, която покрива два аспекта на абстракция – на ниво достъп до данни и базова стилизация на новото разширение.
3. Изследвани са специфичните изисквания на конкретна предметна област – застраховането, и е изградена разпознаваема от индустрията визуализация на развитието на качеството на застрахователно портфолио.
4. Разработени са няколко прототипи на визуализации с конкретна насоченост чрез архитектура за унифицирано разработване на персонализирани визуализации TabWerBI.

Връзката между всеки принос, задачата, поставена в дисертационния труд, позицията на описание, както и съответната включена публикация, е изведена в Таблица 1.

Таблица 1. Таблично разпределение на приносите въз основа на задачи, параграфи в дисертационен труд и научни публикации

Принос	Вид принос	Задачи	Параграф	Публикации
1	научно-приложен	1, 2	1.1, 1.2, 1.3, 2.1, 2.2	1, 2, 3, 4
2	научно-приложен	3	3.1, 3.2, 3.3	-
3	приложен	4	4.1	-
4	приложен	4	4.1, 4.2, 4.3	-

Перспективи за бъдещо развитие

Бъдещо развитие на модела на архитектура TabWerBI, работеща с текущите ППИ на посочените инструменти, може да покрие няколко основни аспекта по отношение на:

- достъпа до базата данни, като се предложи обогатяване на създадената колекция от картографиране на данните;
- налични възможности за форматиране на елементите на новия допълнителен графичен продукт;
- стилизацията на персонализираните информативни диаграми, която може да разширена с повече опции;
- изграждане на способ за генериране на необходимите манифест файлове, съдържащи специфичната за всяка среда мета информация.

Въпреки че едни от най-разпространените BI системи са Microsoft Power BI и Tableau, обект на следващо изследване могат да бъдат спецификите при създаването на персонализирани визуализации и за други актуални системи за оформяне на интерактивни справки като QlikSense, Dundas BI и други.

Апробация на резултатите

Част от резултатите, получени в дисертационния труд, са:

- **внедрени като прототипни визуализации в информационна система на застрахователна компания в Нидерландия:**

1. „Триъгълник на застрахователните претенции“;
2. „Data Label“.

- **докладвани на следните международни конференции:**

1. **Naneva, V**., K. Stefanova, One Approach For Developing A Custom Extension For Tableau, *International Conference on Technology (IConTech’2022),* 16-19 November, 2022, Antalya, Turkey;
2. **Naneva, V**., K. Stefanova, Implementation Of A Custom Lollipop Visual In Power BI, *International Conference on Technology (IConTech’2022),* 16-19 November, 2022, Antalya, Turkey.

- **включени в лабораторните упражнения на дисциплини, водени от автора:**

1. „Интернет технологии“ за специалност „Софтуерно инженерство“, III курс, редовно и задочно обучение;

2. „Информационни технологии в интернет“ за специалност „Софтуерни технологии и дизайн“, IV курс, редовно и задочно обучение.

Участие на автора в научни проекти:

1. Научен проект МУ19-ФМИ009 „Моделиране чрез математика и информатика и симбиозата им с ИКТ“ към НПД на ПУ „Паисий Хилендарски“, 2019 – 2020, участник;
2. Национална научна програма Д01-205 „Информационни и комуникационни технологии за единен цифров пазар в науката, образованието и сигурността“ (ИКТ в науката, образованието и сигурността – ИКТвНОС), 2018 – 2021, участник;
3. Научен проект МУ21-ФМИ-007 „Симбиоза между математиката и информатиката“ (СМИ във ФМИ) към НПД на ПУ „Паисий Хилендарски“, 2021 – 2022, участник;
4. Научен проект МУ21-ФМИ-009 „Систематични иновационни изследвания в математиката и информатиката“ към НПД на ПУ „Паисий Хилендарски“, 2021 – 2022, участник;
5. Национален проект КП-06Н62/1 „Математическо и информационно моделиране на динамични процеси – нови теоретични резултати, методи за изследвания и приложения“ при ФИ към ФНИ, с ръководител проф. д.м.н. С. Христева-Краева, 2022 – 2025, участник;
6. Научен проект МУПД23-ФМИ-009 „Развитие на ИКТ чрез нови изследвания и технологични решения“ към НПД на ПУ „Паисий Хилендарски“, 2023 – 2024, участник.

Публикации по дисертационния труд

Публикации в списания и сборници на конференции, индексирани в Scopus, Web of Science и други бази данни:

1. **Naneva, V.**, K. Stefanova, Implementation of a custom visual in BI tools, *International Journal of Differential Equations and Applications*, 2021, Vol. 20, No. 2, pp. 225–234, ISSN (Print): 1311-2872; ISSN (Online): 1314-6084; doi: 10.12732/ijdea.v20i2.9, (SJR 2021: 0.214, Q4, Indexed in Scopus))

2. **Naneva, V.**, K. Stefanova, A Comparative Analysis of Popular Technologies for Developing Custom Visuals, 46th International Conference Applications of Mathematics in Engineering and Economics, Sozopol, Bulgaria, 7-13 June 2020, *AIP Conference Proceedings*, 2021, 2333, 070009, ISBN: 978-0-7354-4077-7, https://doi.org/10.1063/5.0042201, (SJR 2021: 0.189, Indexed in Scopus & WOS)

Публикации в сборници с доклади на международни конференции:

1. **Naneva, V.**, K. Stefanova, Implementation of a Custom Lollipop Visual in Power BI, *Proceedings of International Conference on Technology (IConTech)*, 2022, pp. 164–171, ISBN: 978-605-72832-5-2, https://www.isres.org/conferences2022_Antalya/ICONTECH 2022_Proceedings.pdf

2. **Naneva, V.**, K. Stefanova, One Approach for Developing a Custom Extension for Tableau, *Proceedings of International Conference on Technology (IConTech)*, 2022, pp. 172–179, ISBN: 978-605-72832-5-2, https://www.isres.org/conferences/2022_Antalya/ICONTECH 2022_Proceedings.pdf

Благодарности

Резултатите, получени в дисертационния труд, са частично финансирани по Проект № КП-06 Н62/1 „Математическо и информационно моделиране на динамични процеси – нови теоретични резултати, методи за изследвания и приложения“ при ФИ към ФНИ, с ръководител проф. д.м.н С. Христева-Краева.

Бих искала да изкажа искрените си благодарности на проф. д-р Ангел Голев и проф. д-р Николай Павлов, които като мои научни ръководители ме напътстваха и насърчаваха по време на работа ми върху дисертационния труд. Признателна съм, че имах възможността да бъда техен докторант. В допълнение, сърдечно благодаря на доц. д-р Кремена Стефанова за съвместната научна работа и за непрестанната ѝ подкрепа в и извън академичната среда. Освен това, благодаря за ценните насоки и забележки, предоставени от членовете на разширения катедрен съвет на катедра „Софтуерни технологии“ по време на предварителното обсъждане на дисертацията.

Библиография

Научни публикации на английски и български език:

[1] R. Galici, L. Ordile, M. Marchesi, A. Pinna, R. Tonelli, Applying the ETL Process to Blockchain Data. Prospect and Findings, *Information*, 2020, 11 (4), 204, https://doi.org/10.3390/info11040204

[2] M. Patel, D. Patel, Progressive Growth of ETL Tools: A Literature Review of Past to Equip Future, *Rising Threats in Expert Applications and Solutions. Advances in Intelligent Systems and Computing*, 2021, Vol. 1187, Springer, Singapore, https://doi.org/10.1007/978-981-15-6014-9_45

[3] M. Dobreva, N. Pavlov, A. Rahnev, Integrate Power Bi With Wpf Desktop Applications, Proc. of the Scientific Conference “Innovative ICT in Research and Education: Mathematics, Informatics and Information Technologies”, 2019, pp. 65–72, ISBN: 978-619-202-439-0

[4] N. Pavlov, M. Dobreva, Pivot Reporting Tool, Сборник с доклади на научна конференция „Иновационни ИКТ: Изследвания, разработка и приложения в бизнеса и обучението“, гр. Хисар, 11–12 ноември 2015 г., стр. 21–30, ISBN: 978-954-8852-56-7

[5] В. Хаджиев, А. Рашидов, Обзор и анализ на методи и модели за структуриране, съхраняване и обработка на данни в интернет, *Автоматика и Информатика 2*, 2019, стр. 27–33, ISSN: 0861-7562

[6] И. Велкова, Обработка и анализ на неструктурирани данни с помощта на изкуствен интелект, *Автоматика и Информатика*, 2022, стр. 27–30, ISSN: 0861-7562

[7] V. Lucie, Factors, Affecting the Adoption of Electronic Data Interchange, *Acta Universitatis Agriculturae et Silviculturae Mendelianae Brunensis*, 2017, 65(6), pp. 2123–2130, ISSN: 12118516

[8] S. Narayanan, A. Marucheck, R. Handfield, Electronic Data Interchange: Research Review and Future Directions, *Decision Science Journal compilation*, Decision Sciences Institute, 2009, 40 (1), pp. 121–163, doi: 10.1111/j.1540-5915.2008.00218.x

[9] N. Elgendy, A. Elragal, Big Data Analytics in Support of the Decision-Making Process, *Procedia Computer Science*, 2016, Vol. 100, pp. 1071–1084

[10] Ajah, H. Nweke, Big Data and Business Analytics: Trends, Platforms, Success Factors and Applications, *Big Data Cogn. Comput*., 2019, 3, 32, https://doi.org/10.3390/bdcc3020032

[11] J. Bharadiya, A Comparative Study of Business Intelligence and Artificial Intelligence with Big Data Analytics, *American Journal of Artificial Intelligence*, 2023, Vol. 7, No. 1, pp. 24–30. doi: 10.11648/j.ajai.20230701.14

[12] J. Bharadiya, Machine Learning and AI in Business Intelligence: Trendsand Opportunities, *International Journal of Computer (IJC)*, 2023, pp. 123–134, ISSN: 2307-4523 (Print & Online)

[13] S. Fahad, A. Yahya, Big Data Visualization: Allotting by R and Python with GUI Tools, 2018 *International Conference on Smart Computing and Electronic Enterprise (ICSCEE)*, Shah Alam, 2018, pp. 1–8, https://doi.org/10.1109/ICSCEE.2018.8538413

[14] M. Muntean, D. Dănăiaţă, L. Hurbean, C. Jude, A Business Intelligence & Analytics Framework for Clean and Affordable Energy Data Analysis, *Sustainability*, 2021, 13, 638, https://doi.org/10.3390/su13020638

[15] C. Smatt, R. Pratt, T. Abegaz, A. Dobbs, Towards effective customer data visualization: data-driven documents (D3.js) vs. Google charts, *A Journal Of International Academy Of Business Disciplines*, 2020, pp. 207–222, ISSN: 2334-0169

[16] S. Prakash, M. Suhal, M. Mithesh, R. Santhiya, The Power of D3.js – Data Visualization, *International Research Journal of Engineering and Technology (IRJET)*, 2020, Vol. 07, Issue 07, pp. 5750–5755, e-ISSN: 2395-0056

[17] Bansal, A. Upadhyay, Microsoft Power BI, *International Journal of Soft Computing and Engineering (IJSCE)*, July 2017, Vol. 7, Issue 3, pp. 14–20, ISSN: 2231-2307

[18] H. Mei, H. Guan, Ch. Xin, X. Wen, W. Chen, DataV: Data Visualization on large high-resolution displays, *Visual Informatics*, 2020, Vol. 4, Issue 3, pp. 12–23, ISSN: 2468-502X, https://doi.org/10.1016/ j.visinf.2020.07.001

[19] S. Karabtsev, R. Kotov, I. Davzit, E. Gurov, Building data marts to analyze university faculty activities using Power BI, *E3S Web of Conferences 419*, 2023, 02014, WFCES 2023, https://doi.org/10.1051/ e3sconf/202341902014

[20] Z. Sun, L. Sun, K. Strang, Big Data Analytics Services for Enhancing Business Intelligence, *Journal of Computer Information Systems*, 2016, pp. 162–169, doi: 10.1080/08874417.2016.1220239

[21] V. Naneva, K. Stefanova, Implementation of a custom visual in BI tools, *International Journal of Differential Equations and Applications*, 2021, Vol. 20, No. 2, pp. 225–234, ISSN (Print): 1311-2872, ISSN (Online): 1314-6084, doi: 10.12732/ijdea.v20i2.9, (SJR 2021: 0.214, Q4, Indexed in Scopus)

[22] V. Naneva, K. Stefanova, A Comparative Analysis of Popular Technologies for Developing Custom Visuals, 46th International Conference Applications of Mathematics in Engineering and Economics, Sozopol, Bulgaria, 7-13 June 2020, *AIP Conference Proceedings*, 2021, 2333, 070009, ISBN: 978-0-7354-4077-7, https://doi.org/10.1063/5.0042201, (SJR 2021: 0.189, Indexed in Scopus & WOS)

[23] V. Naneva, K. Stefanova, Integration of a Custom Visual in a Web Application, 47th International Conference Applications of Mathematics in Engineering and Economics, Sozopol, Bulgaria, 7-13 June 2021, *AIP Conference Proceedings*, 2022, 2505, 060010, ISBN: 978-0-7354-4396-9, https://doi.org/10.1063/5.0102163, (SJR 2022: 0.164, Indexed in Scopus & WOS)

[24] V. Naneva, K. Stefanova, Implementation of a Custom Lollipop Visual in Power BI, *Proc. of International Conference on Technology (IConTech)*, 2022, pp. 164–171, ISBN: 978-605-72832-5-2, https://www.isres.org/conferences2022_Antalya/ICONTECH2022_Proceedings.pdf

[25] V. Naneva, K. Stefanova, One Approach for Developing a Custom Extension for Tableau, *Proc. of International Conference on Technology (IConTech)*, 2022, pp. 172–179, ISBN: 978-605-72832-5-2, https://www.isres.org/conferences/2022_Antalya/ICONTECH2022_Proceedings.pdf

[26] S. Batt, T. Grealis, O. Harmon, P. Tomolonis, Learning Tableau: A data visualization tool, *The Journal of Economic Education*, 2020, 51: 3-4, pp. 317–328, ISSN: 317-328, doi: 10.1080/00220485.2020.1804503

Книги:

[27] F. Zammetti, *Modern Full-Stack Development*, USA: Press Media, LLC is a California, 2020, ISBN: 9781484257371

[28] D. Vanderkam, *Effective TypeScript: 62 Specific Ways to Improve Your TypeScript*, 1st Edition, O'Reilly Media Inc., 2019, ISBN: 9781492053712

[29] D. Choi, *Full-Stack React, TypeScript, and Node*, Packt Publishing, 2020, ISBN: 9781839219931

[30] T. Runkler, *Data Analytics*, Springer Fachmedien Wiesbaden, 2012, ISBN: 978-3-8348-2588-9

[31] P. Zikopoulos, C. Eaton, D. de Roos, T. Deutch, G. Lapis, *Understanding Big Data, Analytics for Enterprise Class Hadoop and Streaming Data*, New York: McGraw-Hill, 2012, ISBN: 978-0-07-179053-6

[32] B. Devlin, Business Intelligence: Insight and Innovation beyond Analytics and Big Data, Technics Publications, 2013, ISBN: 978-163-462-032-1

[33] F. Provost, T. Fawcett, *Data Science for Business*, O'Reilly Media Inc., 2013, ISBN: 978-144-936-132-7

[34] L. Favero, P. Belfiore, *Data Science for Business and Decision Making*, Academic Press, 2019, ISBN: 978-012-811-217-5

[35] C. Sinha, Mastering Power BI Build Business Intelligence Applications Powered with DAX Calculations, Insightful Visualizations, Advanced BI Techniques, and Loads of Data Sources, BPB Publications, 2021, ISBN 978-939-103-072-8

[36] Haldorai, A. Ramu, S. Khan, *Business Intelligence for Enterprise Internet of Things*, Springer International Publishing, 2020, ISBN: 978-303-044-407-5

[37] G. Shmueli, N. Patel, P. Bruce, Data Mining for Busiess Intelligence: Concepts, Techniques, and Applications in Microsoft Office Excel with XLMiner, John Wiley and Sons, 2011, ISBN: 111-812-604-1

[38] M. Meier, D. Baldwin, K. Strachnyi, Mastering Tableau 2021: Implement advanced business intelligence techniques and analytics with Tableau, Packt Publishing Ltd., 2021, ISBN: 978-180-056-074-1

[39] R. Skyrius, *Business Intelligence: A Comprehensive Approach to Information Needs, Technologies and Culture Progress in IS, Springer International Publishing*, 2022, ISBN: 978-303-067-034-4

[40] R. Sherman, Business Intelligence Guidebook: From Data Integration to Analytics, Newnes, 2014, ISBN: 978-012-411-528-6

[41] Business Science Reference, Business Intelligence: Concepts, Methodologies, Tools, and Applications: Concepts, Methodologies, Tools, and Applications, IGI Global, 2015, ISBN: 978-146-669-563-4

[42] R. Sleeper, *Practical Tableau*, O'Reilly Media Inc., 2018, ISBN: 978-149-197-728-6

[43] R. Cherny, Programming TypeScript: Making Your JavaScript Applications Scale, O'Reilly Media Inc., 2019, ISBN: 978-149-203-762-0

[44] D. Vanderkam, *Effective TypeScript: 62 Specific Ways to Improve Your TypeScript*, O'Reilly Media Inc., 2019, ISBN: 978-149-205-369-9

[45] H. Rocha, Learn D3.js: Create interactive data-driven visualizations for the web with the D3.js library, Packt Publishing Ltd, 2019, ISBN: 978-183-864-960-9

[46] D. Knight, M. Pearson, B. Schacht, E. Ostrowsky, Microsoft Power BI Quick Start Guide: Bring your data to life through data modeling, visualization, digital storytelling, and more, Packt Publishing Ltd., 2020, ISBN: 978-180-056-994-2

[47] E. Brown, Learning JavaScript: JavaScript Essentials for Modern Application Development, O'Reilly Media Inc., 2016, ISBN: 978-149-191-493-9

[48] D. Sarka, J. Rihar, K. Vončina, Advanced Analytics with Power BI and Excel: Learn powerful visualization and data analysis techniques using Microsoft BI tools along with Python and R (English Edition), Orange Education Pvt Ltd., 2023, ISBN: 978-9391246709

[49] В. Спасова, *Интелигентни системи за вземане на решения в условията на индустрия 4.0*, Варненски свободен университет „Черноризец Храбър“ – Университетско издателство, 2019, ISBN: 978-954-715-675-3

Интернет източници:

[50] Business Intelligence Definition, https://www.heavy.ai/technical-glossary/business-intelligence (последно достъпен на 18.08.2023 г.)

[51] Ключови показатели за ефективност, https://www.novavizia.com/klyuchovi-pokazateli-za-efektivnost/ (последно достъпен на 18.08.2023 г.)

[52] Mastering the Art of Creating Effective and Intuitive Data Views: Unveiling the Power of Extra Mile Efforts, https://www.linkedin.com/pulse/mastering-art-creating-effective-intuitive-data-views-nikher-verma/ (последно достъпен на 18.08.2023 г.)

[53] Tableau Extensions API Basics, https://tableau.github.io/extensions-api/docs/trex_api_about.html (последно достъпен на 18.08.2023 г.)

[54] Get Started with Dashboard Extensions, https://tableau.github.io/extensions-api/docs/trex_getstarted.html (последно достъпен на 18.08.2023 г.)

[55] What Happens When you Reload an Extension, https://tableau.github.io/extensions-api/docs/trex_reload.html (последно достъпен на 18.08.2023 г.)

[56] Get Data from the Dashboard, https://tableau.github.io/extensions-api/docs/trex_getdata.html (последно достъпен на 18.08.2023 г.)

[57] GetSummaryDataAsync() method, https://tableau.github.io/extensions-api/docs/interfaces/worksheet.html#getsummarydataasync (последно достъпен на 18.08.2023 г.)

[58] HTTPS and Tableau Extensions, https://tableau.github.io/extensions-api/docs/trex_security.html (последно достъпен на 18.08.2023 г.)

[59] Get Data from the Dashboard, https://tableau.github.io/extensions-api/docs/trex_getdata.html#get-the-worksheet-object (последно достъпен на 18.08.2023 г.)

[60] Events and Event Handling, https://tableau.github.io/extensions-api/docs/trex_events.html (последно достъпен на 18.08.2023 г.)

[61] Power BI visuals system integration, https://learn.microsoft.com/en-us/power-bi /developer/visuals/power-bi-visuals-concept (последно достъпен на 18.08.2023 г.)

[62] Power BI visual project structure, https://learn.microsoft.com/en-us/power-bi/developer/visuals/visual-project-structure (последно достъпен на 18.08.2023 г.)

[63] Objects and properties of Power BI visuals, https://learn.microsoft.com/en-us/power-bi/developer/visuals/objects-properties?tabs=getFormattingModel (последно достъпен на 18.08.2023 г.)

[64] Import a Power BI visual from AppSource into your workspace, https://learn.microsoft.com/en-us/power-bi/developer/visuals/import-visual (последно достъпен на 18.08.2023 г.)

[65] Guidelines for publishing Power BI custom visuals, https://learn.microsoft.com/en-us/power-bi/developer/visuals/guidelines-powerbi-visuals (последно достъпен на 18.08.2023 г.)

[66] Test a Power BI custom visual before submitting it for publication, https://learn.microsoft.com/en-us/power-bi/developer/visuals/submission-testing (последно достъпен на 18.08.2023 г.)

[67] Create Power BI visuals with Python, https://learn.microsoft.com/en-us/power-bi/connect-data/desktop-python-visuals (последно достъпен на 18.08.2023 г.)

[68] Understand data view mapping in Power BI visuals, https://learn.microsoft.com/en-us/power-bi/developer/visuals/dataview-mappings (последно достъпен на 18.08.2023 г.)

[69] Library QuickStart, https://basarat.gitbook.io/typescript/library (последно достъпен на 18.08.2023 г.)

[70] TypeScript's Type System, https://basarat.gitbook.io/typescript/type-system (последно достъпен на 18.08.2023 г.)

[71] How to develop a Typescript Library, https://medium.com/@cameronadams1225/how-to-develop-a-typescript-library-ade8d329636 (последно достъпен на 18.08.2023 г.)

[72] Creating Awesome TypeScript NPM Packages, https://levelup.gitconnected.com/creating-awesome-typescript-npm-packages-5d97ac342361 (последно достъпен на 03.09.2023 г.)

[73] Step by step: Building and publishing an NPM Typescript package, https://itnext.io/step-by-step-building-and-publishing-an-npm-typescript-package-44fe7164964c, (последно достъпен на 03.09.2023 г.)

[74] Why Javascript everywhere? And what does it mean for software development?, https://www.linkedin.com/pulse/why-javascript-everywhere-what-does-mean-software-gulshan-sharma/ (последно достъпен на 03.09.2023 г.)

[75] Typescript Webpack, https://webpack.js.org/guides/typescript/ (последно достъпен на 03.09.2023 г.)

[76] Using webpack with TypeScript, https://blog.logrocket.com/using-webpack-typescript/ (последно достъпен на 03.09.2023 г.)

[77] JavaScript Standard Style, https://standardjs.com/ (последно достъпен на 03.09.2023 г.)

[78] Interface Worksheet, https://tableau.github.io/extensions-api/docs/interfaces/worksheet.html (последно достъпен на 03.09.2023 г.)

[79] getSelectedMarksAsync, https://tableau.github.io/extensions-api/docs/interfaces/worksheet.html #getselectedmarksasync (последно достъпен на 03.09.2023 г.)

[80] addEventListener, https://tableau.github.io/extensions-api/docs/interfaces/worksheet.html# addeventlistener (последно достъпен на 03.09.2023 г.)

[81] How To Create A TypeScript Library, https://marketsplash.com/tutorials/typescript/how-to-create-a-typescript-library/ (последно достъпен на 03.09.2023 г.)

[82] WHAT ARE THE TYPES OF GUIDS?, https://www.webopedia.com/definitions/guid/ (последно достъпен на 03.09.2023 г.)

[83] Color picker tool, https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_colors/Color_picker_tool (последно достъпен на 03.09.2023 г.)

[84] Loss Triangles: How to Find Ultimate Loss for Financial Statements, https://www.waterstreetcompany.com/loss-triangle-analytics/, https://www.waterstreetcompany.com/loss-triangle-analytics/ (последно достъпен на 03.09.2023 г.)

[85] Insurance Claims triangle – A jab at SQLScripting, https://blogs.sap.com/2015/06/04/insurance-claims-triangle-a-jab-at-sqlscripting/ (последно достъпен на 03.09.2023 г.)

[86] Telecommunications Analytics, https://www.tableau.com/data-insights/dashboard-showcase/telecommunications-analytics (последно достъпен на 29.09.2023 г.)