T.R.

GEBZE TECHNICAL UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

GTU GRAPHICS MODULE

ERVA AKSU

SUPERVISOR
ASST. PROF. BURCU YILMAZ

GEBZE
2024

T.R.
GEBZE TECHNICAL UNIVERSITY
FACULTY OF ENGINEERING
COMPUTER ENGINEERING DEPARTMENT

GTU GRAPHICS MODULE

ERVA AKSU

SUPERVISOR
ASST. PROF. BURCU YILMAZ

2024
GEBZE

GEBZE TECHNICAL UNIVERSITY	GRADUATION PROJECT JURY APPROVAL FORM

This study has been accepted as an Undergraduate Graduation Project in the Department of Computer Engineering on 25/01/2024 by the following jury.

JURY

Member (Supervisor)	:	Asst. Prof. Burcu Yılmaz
Member	:	Assoc. Prof. Habil Kalkan
Member	:	Prof. Didem Gözüpek
Member	:	Asst. Prof. Tülay Ayyıldız Akoğlu

ABSTRACT

The GTU Graphics Module is a comprehensive tool designed to generate detailed graphics for performance indicators within a system. Focused on facilitating year-based and unit-based comparisons, the module provides a nuanced analysis of system performance. The depth of analysis extends to process-based and resource-based perspectives, enabling users to gain insights into specific operational aspects.

This module serves as an instrument for decision-makers seeking to evaluate and enhance system efficiency. Through visually compelling graphics, it allows for the identification of trends, patterns, and potential areas of improvement over time. The comparison features offer a dynamic view of performance metrics on both macro and micro levels, enabling users to make informed decisions based on a comprehensive understanding of the system's dynamics.

Keywords: GTU Graphics Module, performance indicators, year-based comparison, unit-based comparison, process-based analysis, resource-based analysis, system efficiency.

ÖZET

GTÜ Grafik Modülü, bir sistem içindeki performans göstergeleri için detaylı grafikler üretmek amacıyla tasarlanmış kapsamlı bir araçtır. Yıl bazlı ve birim bazlı karşılaştırmaları kolaylaştırmaya odaklanan modül, sistem performansının ayrıntılı analizini sağlar. Analiz derinliği, süreç tabanlı ve kaynak tabanlı perspektiflere kadar uzanır, kullanıcılara belirli operasyonel yönler hakkında içgörü kazanma imkanı sunar.

Bu modül, sistem verimliliğini değerlendirmek ve artırmak isteyen karar vericiler için bir araç olarak hizmet eder. Görsel olarak etkileyici grafikler aracılığıyla, zaman içinde trendleri, desenleri ve potansiyel iyileştirme alanlarını belirleme imkanı tanır. Karşılaştırma özellikleri, performans metriklerinin makro ve mikro düzeylerde dinamik bir görünümünü sunarak, kullanıcılara sistem dinamiklerinin kapsamlı bir anlayışına dayalı bilinçli kararlar alma imkanı sağlar.

Anahtar Kelimeler: GTÜ Grafik Modülü, performans göstergeleri, yıl bazlı karşılaştırma, birim bazlı karşılaştırma, süreç tabanlı analiz, kaynak tabanlı analiz, sistem verimliliği.

ACKNOWLEDGEMENT

I would like to express my sincere thanks to my advisor Assistant Professor Burcu Yılmaz for keeping track of my work and advising during the semester. Besides, I am also thankful to Dilara Akıncı for her support and guidance throughout the project.

Erva Aksu

LIST OF SYMBOLS AND ABBREVIATIONS

Symbol or Abbreviation	:	**Explanation**
MS	:	Microsoft
MVC	:	Model-View-Controller

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. BACK-END STRUCTURE

The back-end structure of the GTU Graphics Module is implemented using the .NET Core framework and Entity Framework Core as an Object Relational Mapper. To enforce a strict dependency flow, where the inner layers (core and domain services) have no dependencies on the outer layers, Onion Architecture is used as a software architectural pattern that provides a way to organize and structure the components of this software application.

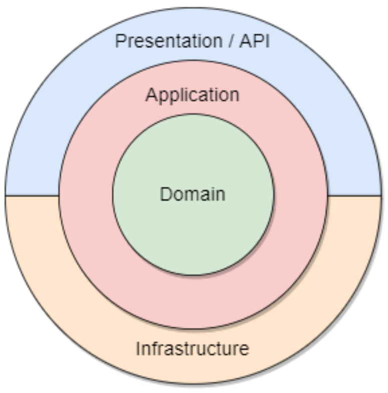


Figure 1.1: Onion architecture that represent the back-end structure.

1.1. Database Structure

The database structure of the project is created using the existing tables to enhance the adaptability to the system. A separate model that represents the entities and their relationships is created for each existing table within the model layer of the back-end structure. A context class is implemented that represents the database context and includes set properties for each model, representing the tables in the database. These tables are created in the MS SQL database using the code first method of Entity Framework Core that uses migrations to create and update the database schema based on changes in the models.

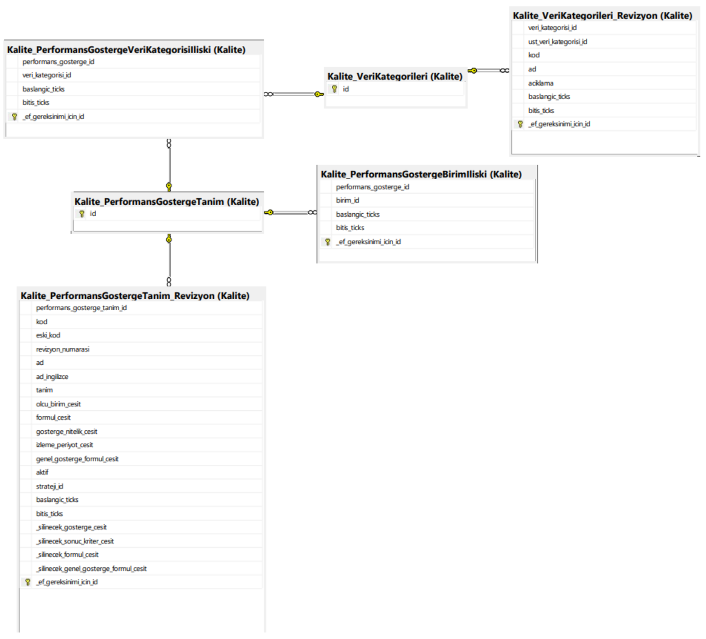


Figure 1.2: Database structure as tables.

1.2. Core Layer

The core layer undertakes the execution of operations related to the creation, retrieval, updating, and deletion of entities, which are independent of specific data models. Within this repository, a structured approach is employed wherein a context object is used to reach the database, and the operations are implemented using this object's functions.

1.3. API Implementation

1.3.1. Business Layer

In the architectural design of the API, a distinct business layer has been to separate the application logic from the controller components. Within this business layer, a modularized structure is established, featuring dedicated interfaces and classes for each model. Business classes use create, read, update, and delete operations in the core layer by customizing them for models. Furthermore, the business layer encapsulates model-specific methods, meticulously implemented in isolation for each class.

1.3.2. Controller Layer

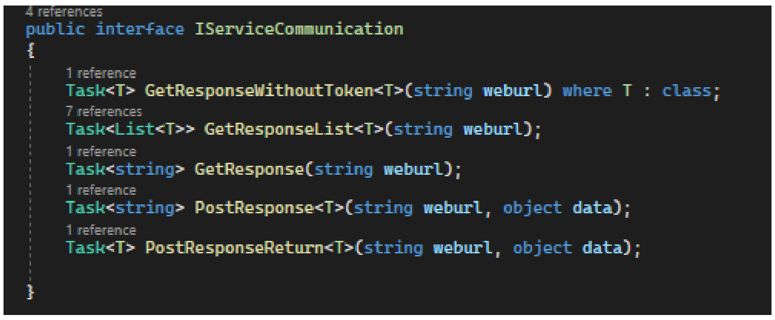
Controllers are responsible for managing incoming requests, processing them, and generating appropriate responses. Each controller is designed to operate independently for specific models within the application. Controllers do not implement the methods but call the appropriate methods using the business layer, which has been pre-implemented.

2. FRONT-END STRUCTURE

The front-end structure of the GTU Graphics Module is implemented using C Sharp, JavaScript, CSS, and HTML languages. This structure comprises two key components: the communication section and the MVC (Model-View-Controller) section. Additionally, the visualization of graphics is facilitated by leveraging the Chart.js library, a powerful JavaScript library for creating interactive and visually appealing charts.

2.1. Service Communication Module

Service Communication Module, a vital component responsible for communication between the MVC controllers and the underlying APIs. The module contains several methods for making HTTP requests and handling responses. These methods are implemented specifically for different responses and requests like an object of type T or lists.



```
4 references
public interface IServiceCommunication
{
    1 reference
    Task<T> GetResponseWithoutToken<T>(string weburl) where T : class;
    7 references
    Task<List<T>> GetResponseList<T>(string weburl);
    1 reference
    Task<string> GetResponse(string weburl);
    1 reference
    Task<string> PostResponse<T>(string weburl, object data);
    1 reference
    Task<T> PostResponseReturn<T>(string weburl, object data);

}
```

Figure 2.1: Interface of the Service Communication Module.

2.2. MVC Structure

Model-View-Controller architecture, a design pattern that has proven instrumental in structuring interactive and maintainable front-end applications. This part of the project uses the models that have been defined to create the database. Therefore, it includes only the controller and view sections.

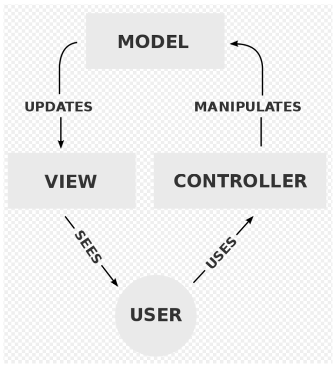


Figure 2.2: Model-View-Controller structure.

2.2.1. MVC Controller Implementation

The controller part of the MVC structure handles data flow between the back-end structure and the views that the user is involved. There is a controller method for each view in the system. These methods get the specific data that is used in that view from the API using the service communication module and return the views with the data.

2.2.2. MVC View Implementations

The system is separated into three parts which include two sub-sections as a user interface. These three view layers go from general to detailed when displaying the data on the system. The two sub-sections separate the data as university-base and unit-base. For visualization, the newest version of the JavaScript library Chart.js is used.

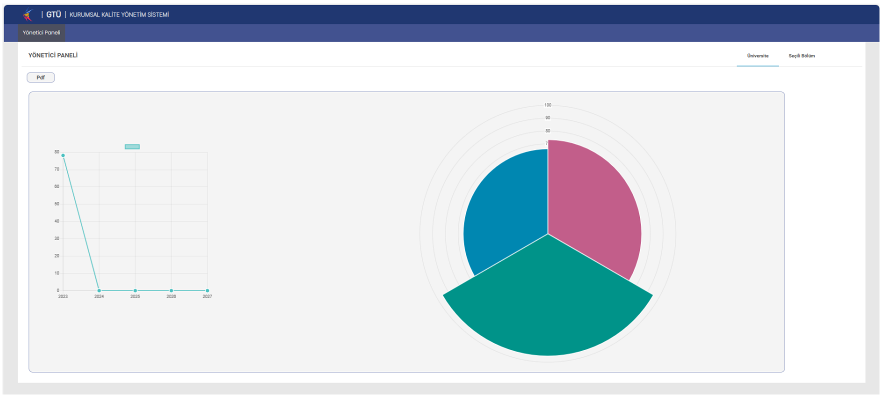


Figure 2.3: The main page that displays the data as general.

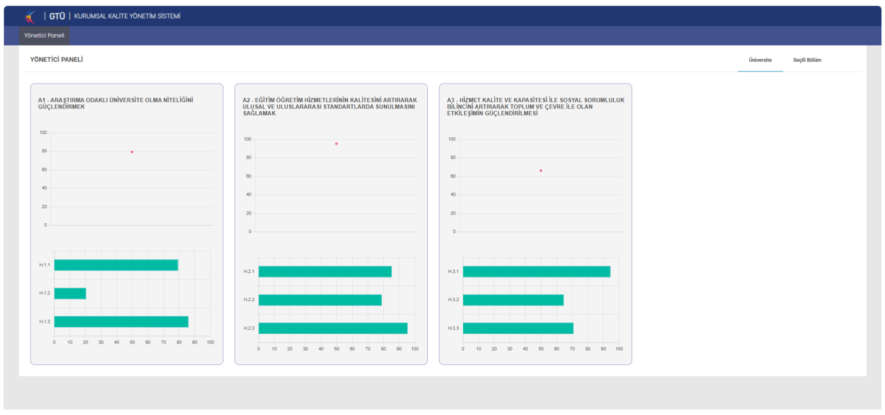
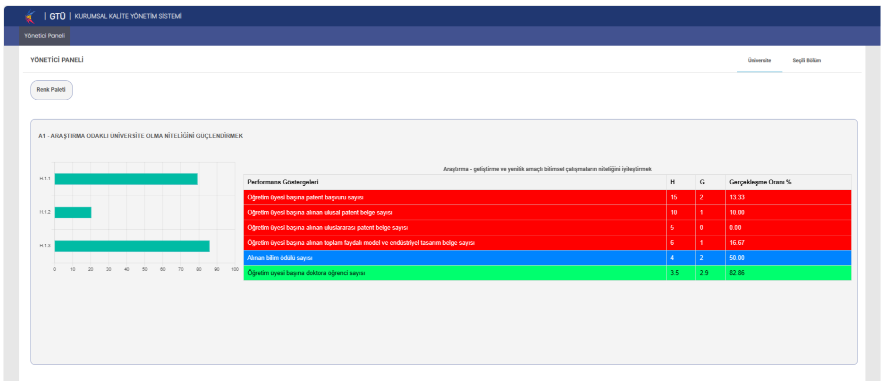

| GTÜ | KURUMSAL KALİTE YÖNETİM SİSTEMİ
Yönetici Paneli
YÖNETİCİ PANELİ
Üniversite
Seçili Bölüm
A1 - ARAŞTIRMA ODAKLI ÜNİVERSİTE OLMA NİTELİĞİNİ GÜÇLENDİRMEK
A2 - EĞİTİM ÖĞRETİM HİZMETLERİNİN KALİTESİNİ ARTIRARAK ULUSAL VE ULUSLARARASI STANDARTLARDA SUNULMASINI SAĞLAMAK
A3 - HİZMET KALİTE VE KAPASİTESİ İLE SOSYAL SORUMLULUK BİLİNCİNİ ARTIRARAK TOPLUM VE ÇEVRE İLE OLAN ETKİLEŞİMİN GÜÇLENDİRİLMESİ
H.1.1
H.1.2
H.1.3
H.2.1
H.2.2
H.2.3
H.3.1
H.3.2
H.3.3

Figure 2.4: The page that displays the performance indicators as general.


| GTÜ | KURUMSAL KALİTE YÖNETİM SİSTEMİ
Yönetici Paneli
YÖNETİCİ PANELİ
Üniversite
Seçili Bölüm
Renk Paleti
A1 - ARAŞTIRMA ODAKLI ÜNİVERSİTE OLMA NİTELİĞİNİ GÜÇLENDİRMEK
H.1.1
H.1.2
H.1.3

Araştırma - geliştirme ve yenilik amaçlı bilimsel çalışmaların niteliğini iyileştirmek

Performans Göstergeleri	H	G	Gerçekleşme Oranı %
Öğretim üyesi başına patent başvuru sayısı	15	2	13.33
Öğretim üyesi başına alınan ulusal patent belge sayısı	10	1	10.00
Öğretim üyesi başına alınan uluslararası patent belge sayısı	5	0	0.00
Öğretim üyesi başına alınan toplam faydalı model ve endüstriyel tasarım belge sayısı	6	1	16.67
Alınan bilim ödülü sayısı	4	2	50.00
Öğretim üyesi başına doktora öğrenci sayısı	3.5	2.9	82.86

Figure 2.5: The page that displays the data in detailed.

3. CONCLUSIONS

In conclusion, the GTU Graphics Module emerges as a valuable ally for decision-makers navigating the complexities of system management. With its user-friendly interface and dynamic visualization capabilities, the module facilitates a deeper understanding of system performance. By offering insights into trends, patterns, and potential improvements, it empowers users to make informed decisions aimed at enhancing overall efficiency.

BIBLIOGRAPHY

https://github.com/NilavPatel/dotnet-onion-architecture https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller https://www.chartjs.org/