

SUNAN: AKIN İNCECİK

SOFTWARE PROJECT *Portfolio*



Presentation Content

1	Crm Demo Project
2	Database Listener Service (RabbitMQ)
3	RabbitMQ Listener Service
4	MVC With AdminLTE (IIS)
5	Angular Web Project

6	React Web Project
7	Vue Web Project
8	Wordpress Project
9	Android Notepad App
10	Python Web Project

SUNAN: AKIN İNCECİK

Crm Demo Project

The project I developed for managing student registrations is designed to collect not only basic user information but also the languages the student knows and wants to learn. After registration, email addresses are saved in the database and used for automated email delivery. The user interface and database integration work seamlessly together.



Welcome

User Name

Personal Records

Personal Informations

ID Number	<input type="text" value="111111111111"/>
Name	<input type="text" value="aaaa"/>
Surname	<input type="text" value="bbbb"/>
Date of Birth	<input type="text" value="29-02-2024"/> <input type="button" value="..."/>
Gender	<input type="radio"/> Not Given <input checked="" type="radio"/> Male <input type="radio"/> Female
Main Language	<input type="button" value="Spanish"/>
Foreign Language	<input type="button" value="Seçiniz"/> <input type="button" value="English"/> <input type="button" value="Spanish"/>
Description	<input type="text" value="cccccccc"/>

Contact Informations

Phone	<input type="text" value="05344509536"/> <input type="button" value="Work"/>
E-mail	<input type="text" value="akinincecik@gmail.com"/> <input type="button" value="Seçiniz"/>
Adress	<input type="text" value="eeee"/> <input type="button" value="Seçiniz"/>

Contact **Type** **Value**

Phone	Work	05344509536
-------	------	-------------

Send EMail

Crm Demo Codes

CRMDemo is a dynamic CRM demo application that manages student registrations and email delivery processes. After logging in, users can register student information; validation rules are applied, and the data can be stored in either MSSQL or PostgreSQL databases.

Project Features:

- Secure Data Connection:** Secure data communication is ensured through encrypted connection strings.
- Asynchronous Email Sending:** Emails are queued and sent asynchronously using RabbitMQ and a background service.
- Email Content:** Email bodies can be stored in both JSON and HTML formats.
- Modular Structure:** The modular SQLUtils structure simplifies and separates database operations.
- Flexible CRM Environment:** This architecture provides a flexible and extensible CRM demo environment.

```
public class SQLUtils
{
    public static class DatabaseHelper
    {
        private static readonly Dictionary<string, Func<string, IDbConnection>> _connectionFactories =
            new Dictionary<string, Func<string, IDbConnection>>
            {
                { "MsSqlConnection", connStr => new SqlConnection(connStr) },
                { "PostgreSqlConnection", connStr => new NpgsqlConnection(connStr) }
            };

        public static string GetConnectionString(string databaseType)
        {
            if (string.IsNullOrWhiteSpace(databaseType))
                throw new ArgumentException("Veritabanı tipi boş olamaz.", nameof(databaseType));

            // Bağlantı dizesini al
            string encryptedConnectionString = ConfigurationManager.ConnectionStrings[databaseType].ConnectionString;
            if (string.IsNullOrEmpty(encryptedConnectionString))
                throw new Exception($"Bağlantı dizesi bulunamadı: {databaseType}");

            // Şifre çözme işlemi
            try
            {
                return SecurityHelper.Decrypt(encryptedConnectionString);
            }
            catch (Exception ex)
            {
                throw new Exception($"Bağlantı dizesi çözülemedi: {ex.Message}");
            }
        }

        public static IDbConnection CreateConnection(string databaseType)
        {
            if (!_connectionFactories.TryGetValue(databaseType, out var connectionFactory))
                throw new ArgumentException($"Geçersiz veritabanı tipi: {databaseType}");

            return connectionFactory(GetConnectionString(databaseType));
        }

        public static void ManageConnectionState(IDbConnection connection, bool open)
        {
            if (connection == null) throw new ArgumentNullException(nameof(connection));

            try
            {
                if (open && connection.State != ConnectionState.Open)
                    connection.Open();
                else if (!open && connection.State != ConnectionState.Closed)
                    connection.Close();
            }
            catch (Exception ex)
            {
                throw new Exception($"Bağlantı yönetimi hatası: {ex.Message}");
            }
        }
    }
}
```

```
namespace CRMDemo
{
    public class SQLUtils
    {
        public static class DatabaseHelper
        {
            public static Dictionary<string, ValidationRule> GetValidationRulesFromDb(string databaseType)
            {
                // Implementation
            }
        }

        public static class QueryHelper
        {
            // Implementation
        }

        public static class AccountRepository
        {
            // Implementation
        }

        public static class DatabaseExecutor
        {
            // Implementation
        }

        private void SaveStudentInformation()
        {
            if (!ValidationUtils.ValidateForm(this, cb_SendMail, skipInfoMailCheck: true))
                return;

            if (!ValidateAndPrepareForm(out var studentParams))
                return;

            if (!cb_SendMail.Checked && !MessageUtils.ConfirmNoInfoMail())
            {
                cb_SendMail.Checked = true;
                tb_InfoMail.Enabled = true;
                tb_InfoMail.Focus();
                return;
            }

            List<string> databases = new List<string> { "MsSqlConnection" /*, "PostgreSqlConnection" */ };
            int lastInsertedId = -1;

            foreach (var db in databases)
            {
                // 📸 Öğrenci verisini fotoğraf ile birlikte her iki veritabanına ekle
                int insertedId = StudentRepository.InsertStudentInformation(db, studentParams);

                if (db == "MsSqlConnection")
                {
                    lastInsertedId = insertedId; // MSSQL'deki ID'yi referans al
                }

                Console.WriteLine($"※ Kaydedilen StudentID ({db}): {insertedId}");
            }

            lastInsertedStudentId = lastInsertedId; // Global değişkeni güncelle
            Console.WriteLine($"※ Global StudentID: {lastInsertedStudentId}");

            if (cb_SendMail.Checked)
            {
                if (!ValidateAndPrepareEmail(out var emailParams))
                    return;

                foreach (var db in databases)
                {
                    EmailRepository.InsertEmailInformation(db, emailParams);
                }
            }

            MessageBox.Show("※ Kayıt işlemi her iki veritabanı için başarılı.");
        }
    }
}
```

CRM Demo Simultaneous Data Recording and Mail Preparation

Data received from the form is saved simultaneously to both MSSQL and PostgreSQL databases. This process ensures data integrity across both systems and enables synchronization between them.

id	state	recipient	subject	body	created_at	sent_at	errormessage	jsondata
117	145	2 akineendmail@gmail.co	Öğrenci kayıt bilgi formu	Test	2025-02-24 19:27:14.116563	2025-02-24 19:27:14.116563	[null]	{"PersonInfo": {"Name": "aaaa", "Level": 0, "Gender": "Not Given", "Surname": "bbbb", "IdNumber": "111111111111", "DateOfBirth": "2024-02-25 01:10:14.207696"}, "Email": "akineendmail@gmail.com", "Subject": "Öğrenci kayıt bilgi formu", "Body": "Test", "Created": "2025-02-24 19:27:14.116563", "Sent": "2025-02-24 19:27:14.116563", "Error": null}
118	146	2 akineendmail@gmail.co	Öğrenci kayıt bilgi formu	Test	2025-02-25 00:49:38.686784	2025-02-25 00:49:38.686784	[null]	{"PersonInfo": {"Name": "aaaa", "Level": 0, "Gender": "Not Given", "Surname": "bbbb", "IdNumber": "111111111111", "DateOfBirth": "2024-02-25 01:10:14.207696"}, "Email": "akineendmail@gmail.com", "Subject": "Öğrenci kayıt bilgi formu", "Body": "Test", "Created": "2025-02-25 00:49:38.686784", "Sent": "2025-02-25 00:49:38.686784", "Error": null}
119	147	2 akineendmail@gmail.co	Öğrenci kayıt bilgi formu	Test	2025-02-25 00:51:41.879824	2025-02-25 00:51:41.879824	[null]	{"PersonInfo": {"Name": "aaaa", "Level": 0, "Gender": "Not Given", "Surname": "bbbb", "IdNumber": "111111111111", "DateOfBirth": "2024-02-25 01:10:14.207696"}, "Email": "akineendmail@gmail.com", "Subject": "Öğrenci kayıt bilgi formu", "Body": "Test", "Created": "2025-02-25 00:51:41.879824", "Sent": "2025-02-25 00:51:41.879824", "Error": null}
120	148	2 akineendmail@gmail.co	Öğrenci kayıt bilgi formu	Test	2025-02-25 01:00:09.708401	2025-02-25 01:00:09.708401	[null]	{"PersonInfo": {"Name": "aaaa", "Level": 0, "Gender": "Not Given", "Surname": "bbbb", "IdNumber": "111111111111", "DateOfBirth": "2024-02-25 01:10:14.207696"}, "Email": "akineendmail@gmail.com", "Subject": "Öğrenci kayıt bilgi formu", "Body": "Test", "Created": "2025-02-25 01:00:09.708401", "Sent": "2025-02-25 01:00:09.708401", "Error": null}
121	149	2 akineendmail@gmail.co	Öğrenci kayıt bilgi formu	Test	2025-02-25 01:01:51.8521	2025-02-25 01:01:51.8521	[null]	{"PersonInfo": {"Name": "aaaa", "Level": 0, "Gender": "Not Given", "Surname": "bbbb", "IdNumber": "111111111111", "DateOfBirth": "2024-02-25 01:10:14.207696"}, "Email": "akineendmail@gmail.com", "Subject": "Öğrenci kayıt bilgi formu", "Body": "Test", "Created": "2025-02-25 01:01:51.8521", "Sent": "2025-02-25 01:01:51.8521", "Error": null}
122	150	2 akineendmail@gmail.co	Öğrenci kayıt bilgi formu	Test	2025-02-25 01:10:14.207696	2025-02-25 01:10:14.207696	[null]	{"PersonInfo": {"Name": "aaaa", "Level": 0, "Gender": "Not Given", "Surname": "bbbb", "IdNumber": "111111111111", "DateOfBirth": "2024-02-25 01:10:14.207696"}, "Email": "akineendmail@gmail.com", "Subject": "Öğrenci kayıt bilgi formu", "Body": "Test", "Created": "2025-02-25 01:10:14.207696", "Sent": "2025-02-25 01:10:14.207696", "Error": null}
123	151	2 akineendmail@gmail.co	Öğrenci kayıt bilgi formu	Test	2025-02-25 01:13:39.529296	2025-02-25 01:13:39.529296	[null]	{"PersonInfo": {"Name": "aaaa", "Level": 0, "Gender": "Not Given", "Surname": "bbbb", "IdNumber": "111111111111", "DateOfBirth": "2024-02-25 01:10:14.207696"}, "Email": "akineendmail@gmail.com", "Subject": "Öğrenci kayıt bilgi formu", "Body": "Test", "Created": "2025-02-25 01:13:39.529296", "Sent": "2025-02-25 01:13:39.529296", "Error": null}
124	152	2 akineendmail@gmail.co	Öğrenci kayıt bilgi formu	Test	2025-02-25 01:19:29.236983	2025-02-25 01:19:29.236983	[null]	{"PersonInfo": {"Name": "aaaa", "Level": 0, "Gender": "Not Given", "Surname": "bbbb", "IdNumber": "111111111111", "DateOfBirth": "2024-02-25 01:10:14.207696"}, "Email": "akineendmail@gmail.com", "Subject": "Öğrenci kayıt bilgi formu", "Body": "Test", "Created": "2025-02-25 01:19:29.236983", "Sent": "2025-02-25 01:19:29.236983", "Error": null}
125	153	2 akineendmail@gmail.co	Öğrenci kayıt bilgi formu	Test	2025-02-25 01:19:32.709725	2025-02-25 01:19:32.709725	[null]	{"PersonInfo": {"Name": "aaaa", "Level": 0, "Gender": "Not Given", "Surname": "bbbb", "IdNumber": "111111111111", "DateOfBirth": "2024-02-25 01:10:14.207696"}, "Email": "akineendmail@gmail.com", "Subject": "Öğrenci kayıt bilgi formu", "Body": "Test", "Created": "2025-02-25 01:19:32.709725", "Sent": "2025-02-25 01:19:32.709725", "Error": null}
126	154	2 akineendmail@gmail.co	Öğrenci kayıt bilgi formu	Test	2025-02-25 01:22:07.666996	2025-02-25 01:22:07.666996	[null]	{"PersonInfo": {"Name": "aaaa", "Level": 0, "Gender": "Not Given", "Surname": "bbbb", "IdNumber": "111111111111", "DateOfBirth": "2024-02-25 01:10:14.207696"}, "Email": "akineendmail@gmail.com", "Subject": "Öğrenci kayıt bilgi formu", "Body": "Test", "Created": "2025-02-25 01:22:07.666996", "Sent": "2025-02-25 01:22:07.666996", "Error": null}
127	155	0 akineendmail@gmail.co	Öğrenci kayıt bilgi formu	Test	2025-02-25 01:22:26.675795	2025-02-25 01:22:26.675795	[null]	{"PersonInfo": {"Name": "aaaa", "Level": 0, "Gender": "Not Given", "Surname": "bbbb", "IdNumber": "111111111111", "DateOfBirth": "2024-02-25 01:10:14.207696"}, "Email": "akineendmail@gmail.com", "Subject": "Öğrenci kayıt bilgi formu", "Body": "Test", "Created": "2025-02-25 01:22:26.675795", "Sent": "2025-02-25 01:22:26.675795", "Error": null}
128	156	0 akineendmail@gmail.co	Öğrenci kayıt bilgi formu	Test	2025-02-25 01:23:36.370751	2025-02-25 01:23:36.370751	[null]	{"PersonInfo": {"Name": "aaaa", "Level": 0, "Gender": "Not Given", "Surname": "bbbb", "IdNumber": "111111111111", "DateOfBirth": "2024-02-25 01:10:14.207696"}, "Email": "akineendmail@gmail.com", "Subject": "Öğrenci kayıt bilgi formu", "Body": "Test", "Created": "2025-02-25 01:23:36.370751", "Sent": "2025-02-25 01:23:36.370751", "Error": null}
129	157	0 akineendmail@gmail.co	Öğrenci kayıt bilgi formu	Test	2025-02-25 01:28:53.674462	2025-02-25 01:28:53.674462	[null]	{"PersonInfo": {"Name": "aaaa", "Level": 0, "Gender": "Not Given", "Surname": "bbbb", "IdNumber": "111111111111", "DateOfBirth": "2024-02-25 01:10:14.207696"}, "Email": "akineendmail@gmail.com", "Subject": "Öğrenci kayıt bilgi formu", "Body": "Test", "Created": "2025-02-25 01:28:53.674462", "Sent": "2025-02-25 01:28:53.674462", "Error": null}
130	158	0 akineendmail@gmail.co	Öğrenci kayıt bilgi formu	Test	2025-02-25 01:29:46.688724	2025-02-25 01:29:46.688724	[null]	{"PersonInfo": {"Name": "aaaa", "Level": 0, "Gender": "Not Given", "Surname": "bbbb", "IdNumber": "111111111111", "DateOfBirth": "2024-02-25 01:10:14.207696"}, "Email": "akineendmail@gmail.com", "Subject": "Öğrenci kayıt bilgi formu", "Body": "Test", "Created": "2025-02-25 01:29:46.688724", "Sent": "2025-02-25 01:29:46.688724", "Error": null}

SQLQuery1.sql - AKI...t_Records (sa (63))

```
SELECT TOP (1000) [Id]
    ,[FullName]
    ,[FullSurname]
    ,[DateOfBirth]
    ,[MainLanguage]
    ,[ForeignLanguage]
    ,[ForeignLanguageLevel]
    ,[Description]
    ,[PhoneNumber]
    ,[Email]
    ,[Adress]
    ,[PicturePath]
FROM [Student_Records].[dbo].[StudentInformations]
```

Results

Id	FullName	FullSurname	DateOfBirth	MainLanguage	ForeignLanguage	ForeignLanguageLevel	Description	PhoneNumber	Email	Adress	PicturePath	
724	3...	aaaa	bbbb	2024-02-29 00:00:00.0000	English	Spanish	0	cccccccc	05344508315	akinincecik@gmail.com	eeee	0xFFD8FFE000104A4649
725	3...	aaaa	bbbb	2024-02-29 00:00:00.0000	English	Spanish	0	cccccccc	05344508315	akinincecik@gmail.com	eeee	0xFFD8FFE000104A4649
726	3...	aaaa	bbbb	2024-02-29 00:00:00.0000	English	Spanish	0	cccccccc	05344508315	akinincecik@gmail.com	eeee	0xFFD8FFE000104A4649
727	3...	aaaa	bbbb	2024-02-29 00:00:00.0000	English	Spanish	0	cccccccc	05344508315	akinincecik@gmail.com	eeee	0xFFD8FFE000104A4649
728	3...	aaaa	bbbb	2024-02-29 00:00:00.0000	English	Spanish	0	cccccccc	05344508315	akinincecik@gmail.com	eeee	0xFFD8FFE000104A4649
729	3...	aaaa	bbbb	2024-02-29 00:00:00.0000	English	Spanish	0	cccccccc	05344508315	akinincecik@gmail.com	eeee	0xFFD8FFE000104A4649
730	3...	aaaa	bbbb	2024-02-29 00:00:00.0000	English	Spanish	0	cccccccc	05344508315	akinincecik@gmail.com	eeee	0xFFD8FFE000104A4649
731	3...	aaaa	bbbb	2024-02-29 00:00:00.0000	English	Spanish	0	cccccccc	05344508315	akinincecik@gmail.com	eeee	0xFFD8FFE000104A4649
732	3...	aaaa	bbbb	2024-02-29 00:00:00.0000	English	Spanish	0	cccccccc	05344508315	akinincecik@gmail.com	eeee	0xFFD8FFE000104A4649
733	3...	aaaa	bbbb	2024-02-29 00:00:00.0000	English	Spanish	0	cccccccc	05344508315	akinincecik@gmail.com	eeee	0xFFD8FFE000104A4649
734	3...	aaaa	bbbb	2024-02-29 00:00:00.0000	English	Spanish	0	cccccccc	05344508315	akinincecik@gmail.com	eeee	0xFFD8FFE000104A4649
735	3...	aaaa	bbbb	2024-02-29 00:00:00.0000	English	Spanish	0	cccccccc	05344508315	akinincecik@gmail.com	eeee	0xFFD8FFE000104A4649
736	3...	aaaa	bbbb	2024-02-29 00:00:00.0000	English	Spanish	0	cccccccc	05344508315	akinincecik@gmail.com	eeee	0xFFD8FFE000104A4649
737	3...	aaaa	bbbb	2024-02-29 00:00:00.0000	English	Spanish	0	cccccccc	05344508315	akinincecik@gmail.com	eeee	0xFFD8FFE000104A4649
738	3...	aaaa	bbbb	2024-02-29 00:00:00.0000	English	Spanish	0	cccccccc	05344508315	akinincecik@gmail.com	eeee	0xFFD8FFE000104A4649

The saved data is stored with a "Queued" status, ready for email delivery. The background service picks up this data, formats it into an email, and automatically initiates the sending process.

Database Listener Service

This service analyzes the statuses of the data saved by the CRM demo application. It identifies the records that are ready for email delivery and forwards them to the RabbitMQ queue. This way, the necessary data for the sending process is automatically transferred to the next stage.

```

1 reference
public static void ProcessEmailQueue()
{
    Console.WriteLine($"⌚ Yeni kayıtlar kontrol ediliyor...");

    // StatusId = 0 olan e-mailleri al
    DataTable newEmails = SQLUtils.GetSqlQueuedEmails();

    foreach (DataRow row in newEmails.Rows)
    {
        int emailId = Convert.ToInt32(row["Id"]);

        // Her bir emailId için detaylı veriyi al
        DataTable emailDetails = SQLUtils.GetEmailFromId(emailId);

        if (emailDetails.Rows.Count > 0)
        {
            // Veritabanından gelen veriyi uygun formata getir
            var emailData = emailDetails.Rows[0];

            var emailParams = new Dictionary<string, object>
            {
                { "@Recipient", emailData["Recipient"] },
                { "@Subject", emailData["Subject"] },
                { "@Body", emailData["Body"] },
                { "@JsonData", emailData["JsonData"] }, // JSON formatındaki veri
                { "@HtmlData", emailData["HtmlData"] }, // HTML formatındaki veri
                { "@EmailId", emailData["Id"] } // EmailId parametresi eklendi
            };

            // JSON'a çevir
            string jsonData = JsonConvert.SerializeObject(emailParams);

            try
            {
                // RabbitMQ'ya gönder
                RabbitMQHelper.SendMessage(jsonData);

                // Başarıyla RabbitMQ'ya gönderildiye durumu 6 (Sent) yap
                SQLUtils.UpdateSqlEmailStatus(emailId, 6);
                Console.WriteLine($"✉️ Email ID {emailId} RabbitMQ'ya başarıyla gönderildi");
            }
            catch (Exception ex)
            {
                // RabbitMQ'ya gönderilemediye durumu 4 (Failed) yap
                SQLUtils.UpdateSqlEmailStatus(emailId, 4);
                Console.WriteLine($"✖️ Email ID {emailId}: RabbitMQ gönderimi başarısız oldu");
            }
        }
    }
}

```

Overview		Messages		Message rates						
Virtual host	Name	Type	Features	State	Ready	Unacked	Total	incoming	deliver / get	ack
/	emailQueue	classic	Args	idle	0	0	0		0.00/s	0.00/s
/	email_queue	classic	D Args	idle	1	0	1	0.00/s	0.00/s	0.00/s

RabbitMQ Listener Service

This service continuously listens to the RabbitMQ queue to check for new data. When data is found in the queue, it retrieves the appropriate email template stored in the database, combines it with the relevant data, and executes the email delivery.

```

public static class EmailSender
{
    public static void ProcessAndSendEmails(Dictionary<string, object> emailData, string dbType)
    {
        try
        {
            if (!IsValidEmailData(emailData, out int emailId, out string recipient, out string subject, out string body, out string jsonData))
            {
                UpdateEmailStatus(dbType, emailId, 5); // JSON Hatalı (Failed)
                return;
            }

            string finalBody = PrepareEmailBody(dbType, emailId, body, jsonData);

            if (string.IsNullOrWhiteSpace(finalBody))
            {
                Console.WriteLine($"⚠️ Email ID {emailId}: HTML tablo oluşturulamadı.");
                UpdateEmailStatus(dbType, emailId, 5);
                return;
            }

            if (SendEmailWithRetry(recipient, subject, finalBody))
            {
                Console.WriteLine($"✉️ Email ID {emailId} başarıyla gönderildi.");
                UpdateEmailStatus(dbType, emailId, 2); // Sent
            }
            else
            {
                Console.WriteLine($"✖️ Email ID {emailId} gönderilmedi. Retrying...");
                UpdateEmailStatus(dbType, emailId, 4); // Retrying
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine($"✖️ Email işlenirken hata oluştu: {ex.Message}");
        }
    }

    private static bool IsValidEmailData(Dictionary<string, object> emailData, out int emailId, out string recipient, out string subject, out string body, out string jsonData)
    {
        emailId = emailData.ContainsKey("@EmailId") ? Convert.ToInt32(emailData["@EmailId"]) : 0;

        recipient = emailData.TryGetValue("@Recipient", out var recipientObj) ? recipientObj?.ToString() ?? "" : "";
        subject = emailData.TryGetValue("@Subject", out var subjectObj) ? subjectObj?.ToString() ?? "" : "";
        body = emailData.TryGetValue("@Body", out var bodyObj) ? bodyObj?.ToString() ?? "" : "";
        jsonData = emailData.TryGetValue("@jsonData", out var jsonObj) ? jsonObj?.ToString() ?? "" : "";

        if (string.IsNullOrWhiteSpace(recipient) || string.IsNullOrWhiteSpace(subject) || string.IsNullOrWhiteSpace(body) || string.IsNullOrWhiteSpace(jsonData))
        {
            Console.WriteLine($"⚠️ Email ID {emailId}: Eksik email alanları tespit edildi.");
            return false;
        }

        if (!Utils.IsValidJson(jsonData))
        {
            Console.WriteLine($"⚠️ Email ID {emailId}: JSON verisi hatalı.");
            return false;
        }

        return true;
    }

    private static string PrepareEmailBody(string dbType, int emailId, string body, string jsonData)
    {
        int studentId = GetStudentIdFromJson(jsonData);
        string picturePath = SQLUtils.GetStudentPicturePath(dbType, studentId);
        string htmlData = Utils.GenerateHTMLTable(jsonData);

        if (!string.IsNullOrWhiteSpace(picturePath))
        {
            body += $"{<br><br><img src='{picturePath}' width='200' height='200' />}";
        }

        if (string.IsNullOrWhiteSpace(htmlData))
        {
            Console.WriteLine($"⚠️ Email ID {emailId}: HTML tablo oluşturulamadı.");
            return null;
        }

        return body + "<br><br>" + htmlData;
    }

    private static bool SendEmailWithRetry(string recipient, string subject, string finalBody, int retryCount = 3)
    {
        for (int i = 0; i < retryCount; i++)
        {
            string result = Utils.SendEmail(recipient, subject, finalBody);

            if (result == "OK")
                return true;

            Console.WriteLine($"⚠️ Email gönderme denemesi başarısız (Deneme {i + 1}/{retryCount})");
            Thread.Sleep(2000); // 2 saniye bekleme
        }

        return false;
    }
}

```

TC Kimlik No	1111111111
Ad	aaaa
Soyad	bbbb
Dogum Tarihi	2024-02-29T00:00:00
Ana Dil	Spanish
Yabancı Dil	Seçiniz
Yabancı Dil Seviyesi	
Telefon Numarası	
E-Mail	Adres

```

public class RabbitMQListener
{
    private static IConnection _connection;
    private static IModel _channel;

    private readonly string _queueName = "email_queue";
    private readonly string _hostName = "localhost"; // RabbitMQ sunucusu adresi
    private readonly string _username = "guest";
    private readonly string _password = "guest";

    public void StartListening(string dbType)
    {
        try
        {
            var factory = new ConnectionFactory()
            {
                HostName = _hostName,
                UserName = _username,
                Password = _password
            };

            _connection = factory.CreateConnection();
            _channel = _connection.CreateModel();

            _channel.QueueDeclare(queue: _queueName, durable: true, exclusive: false, autoDelete: false, arguments: null);
            var consumer = new EventingBasicConsumer(_channel);

            consumer.Received += (model, ea) =>
            {
                try
                {
                    var body = ea.Body.ToArray();
                    var message = Encoding.UTF8.GetString(body);
                    Console.WriteLine($"✉️ [+] Received message: {message}");

                    var emailData = JsonConvert.DeserializeObject<Dictionary<string, object>>(message);

                    if (emailData != null && emailData.ContainsKey("@EmailId"))
                    {
                        int emailId = Convert.ToInt32(emailData["@EmailId"]);
                        Console.WriteLine($"✉️ [+] İşlem başlatılıyor, Email ID: {emailId}");

                        EmailSender.ProcessAndSendEmails(emailData, dbType);
                    }
                    else
                    {
                        Console.WriteLine("⚠️ Geçersiz mesaj formatı.");
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"✖️ Hata: {ex.Message}");
                }
            };
        }
        catch (Exception ex)
        {
            Console.WriteLine($"✖️ Hata: {ex.Message}");
        }
    }

    // Mesajı onayla
    _channel.BasicAck(ea.DeliveryTag, false);

    _channel.BasicConsume(queue: _queueName, autoAck: false, consumer: consumer);
    Console.WriteLine("✉️ Kuruk dinleniyor...");

    // Thread'i kilitleden açık tutuyoruz
    Task.Delay(-1).Wait(); // ✅ Sonsuz döngü yerine Task.Delay kullanıyoruz
}

class Worker
{
    private static Object _lock = new Object();
    private static DatabaseParameters _dbParams = new DatabaseParameters(); // ✅ Bağlantı bilgilerini merkezi olarak çekiyoruz.

    public static void testMethod(object arg)
    {
        ThreadParameters threadParameters = (ThreadParameters)arg;
        List<string> databases = new List<string> { "SQL/*", "PostgreSQL/*" }; // ✅ Veritabanı türlerini listeye alıyoruz.

        while (true)
        {
            try
            {
                lock (_lock)
                {
                    foreach (var dbType in databases)
                    {
                        RabbitMQListener subscriber = new RabbitMQListener();
                        subscriber.StartListening(dbType);
                    }
                }
            }
            catch (Exception ex)
            {
                LogError(ex);
            }
            Thread.Sleep(20);
        }
    }

    private static void LogError(Exception ex)
    {
        using (EventLog eventLog = new EventLog("Application"))
        {
            eventLog.Source = "Application";
            eventLog.WriteEntry($"WindowsServiceTest Error: {ex.StackTrace}, EventLogEntryType.Error, 101, 1");
        }
    }
}

```

MVC With IIS

This project is an MVC-based admin panel application built using the AdminLTE template. It provides efficient management with a user-friendly interface and can operate in a real environment on IIS. With its modern and responsive design, it offers core functions like data management and user interaction. Additionally, thanks to AdminLTE's flexibility and customizability, it can be easily extended.

```
namespace AdminLTE.Controllers
{
    [Authorize(Roles = "admin", AuthenticationSchemes = CookieAuthenticationDefaults.AuthenticationScheme)]
    public class SupplierController : Controller
    {
        private readonly DataBaseContext _dataBaseContext;
        private readonly IMapper _mapper;
        private readonly IHasher _hasher;

        public SupplierController(DataBaseContext dataBaseContext, IMapper mapper, IHasher hasher)
        {
            _dataBaseContext = dataBaseContext;
            _mapper = mapper;
            _hasher = hasher;
        }

        public IActionResult Index()
        {
            ViewBag.CurrentPage = 1;
            ViewBag.ActiveMenu = "supplier-management"; // Açık kalmasını istediğiniz menü

            return View();
        }

        public IActionResult SupplierListPartial()
        {
            List<SupplierViewModel> suppliers =
                _dataBaseContext.Suppliers.ToList()
                .Select(x => _mapper.Map<SupplierViewModel>(x)).ToList();

            return PartialView("_SupplierListPartial", suppliers);
        }

        public IActionResult SupplierListPartial(int page = 1)
        {
            Console.WriteLine($"Gelen Sayfa: {page}");

            ViewBag.CurrentPage = page;

            const int pageSize = 25;
            int skipCount = (page - 1) * pageSize;
            Console.WriteLine($"Şu anki sayfa: {page}, Atlanacak veri: {skipCount}");

            var suppliers = _dataBaseContext.Suppliers
                .OrderByDescending(s => s.Id)
                .Skip(skipCount)
                .Take(pageSize)
                .Select(s => _mapper.Map<SupplierViewModel>(s))
                .ToList();

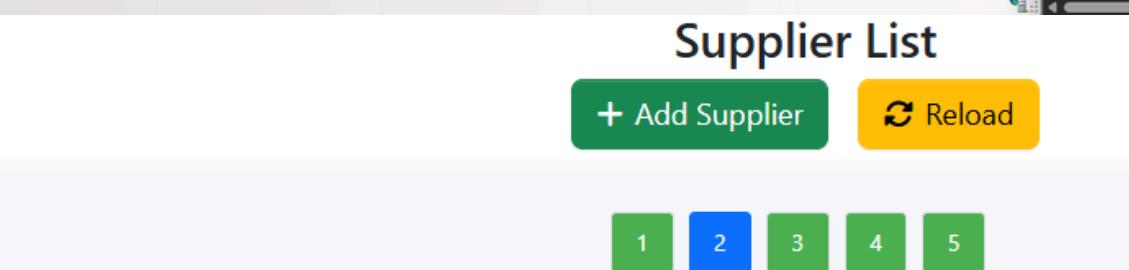
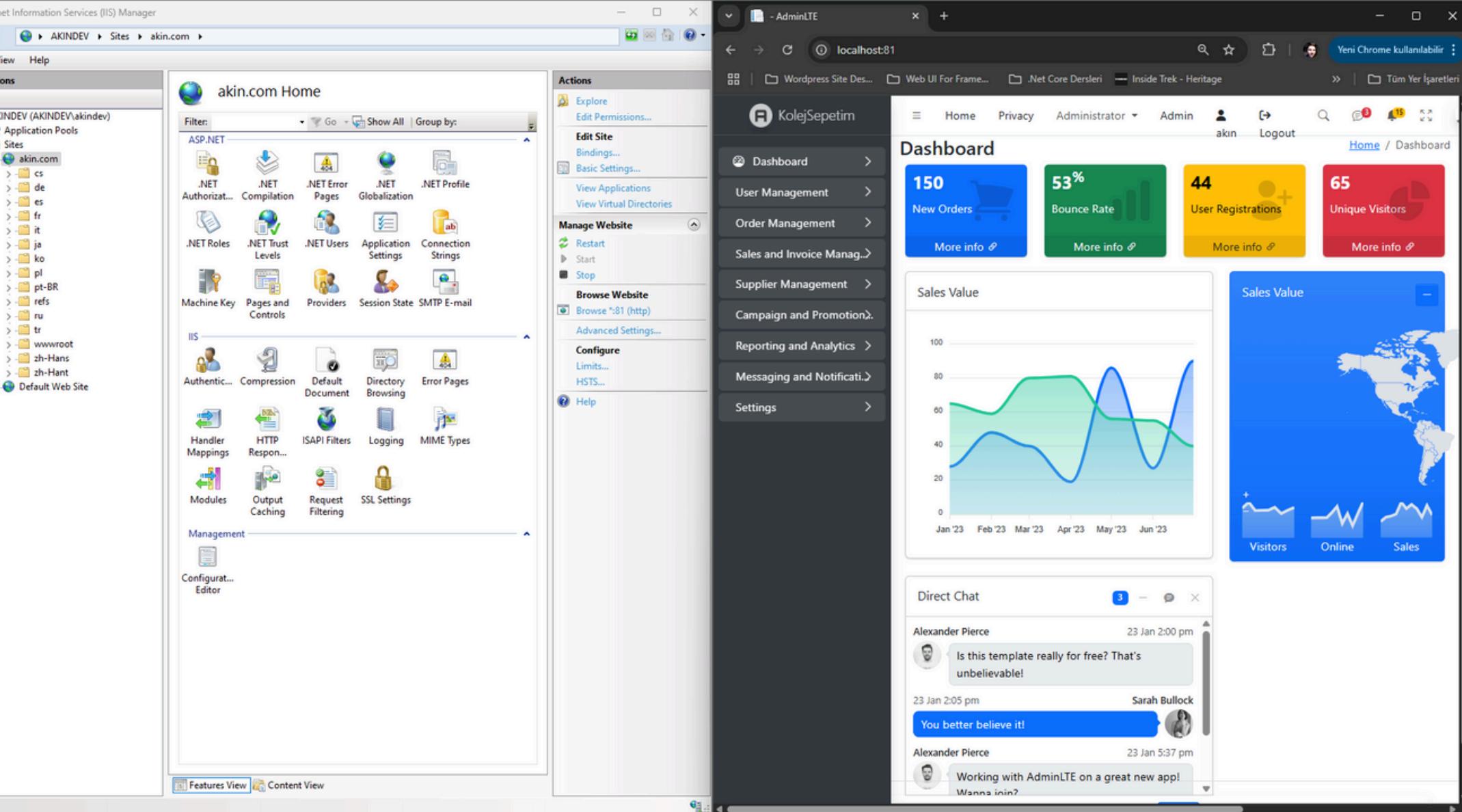
            Console.WriteLine($"Sayfa: {page}, Skip: {skipCount}, Toplam Çekilen: {suppliers.Count}");

            var totalSuppliers = _dataBaseContext.Suppliers.Count();
            var totalPages = (int)Math.Ceiling(totalSuppliers / (double)pageSize);

            Console.WriteLine($"Toplam: {totalSuppliers}, Toplam Sayfa: {totalPages}");

            var model = new SupplierListViewModel
            {
                Suppliers = suppliers,
                TotalPages = totalPages
            };
            return PartialView("_SupplierListPartial", model);
        }
    }
}
```

With its modern and responsive design, it offers core functions like data management and user interaction. Additionally, AdminLTE's flexibility and customizability allow for easy expansion.



Supplier ID	Supplier Name	Address	Phone Number	Email	Created At
79	Supplier 79	Street 79, City, USA	+1-555-1079	supplier79@mail.com	2025-01-07 23:51
78	Supplier 78	Street 78, City, USA	+1-555-1078	supplier78@mail.com	2025-01-08 23:51
77	Supplier 77	Street 77, City, USA	+1-555-1077	supplier77@mail.com	2025-01-09 23:51
76	Supplier 76	Street 76, City, USA	+1-555-1076	supplier76@mail.com	2025-01-10 23:51
75	Supplier 75	Street 75, City, USA	+1-555-1075	supplier75@mail.com	2025-01-11 23:51
74	Supplier 74	Street 74, City, USA	+1-555-1074	supplier74@mail.com	2025-01-12 23:51
73	Supplier 73	Street 73, City, USA	+1-555-1073	supplier73@mail.com	2025-01-13 23:51
72	Supplier 72	Street 72, City, USA	+1-555-1072	supplier72@mail.com	2025-01-14 23:51
71	Supplier 71	Street 71, City, USA	+1-555-1071	supplier71@mail.com	2025-01-15 23:51

Angular Web Project

The Angular-First-App project is a simple blog page developed using the Angular framework. This project was created to learn and apply the fundamental features of Angular. A modular structure was built using Angular's strengths such as component-based architecture and data binding. The project is developed with TypeScript, HTML, and CSS and includes important Angular concepts like routing, user interaction, and service integration between key components.

The screenshot displays a development setup with two main windows. On the left, the Visual Studio Code interface shows the project structure of 'FIRSTAPP' with files like 'home.component.html', 'home.component.spec.ts', and 'home.component.css'. The 'home.component.spec.ts' file is open, showing a unit test for the HomeComponent using TestBed. On the right, a web browser window titled 'Applications1' shows the 'BlogApp' application running at 'localhost:4200/home'. The app has a sidebar with 'POPÜLER KATEGORİLER' (Technology, Health, Travel, Food) and a main content area with 'Blog Yazılıları' (Blog Posts). Below the posts are sections for 'SON YORUMLAR' (Reviews) and 'Angular'a Giriş' (Introduction to Angular). The bottom of the browser window shows the terminal output of the 'ng serve' command, indicating the application is running and listening on port 4200.

React Web Project

The React-first-app project is a simple web application developed using the React framework. This project was created to learn and apply the core concepts of React. The app offers a modular design using component-based architecture and efficiently manages state and user interactions with React's powerful features. The project serves as an example to understand key functions of React like useState, useEffect, and event handling. Additionally, a dynamic and interactive user interface is created using modern JavaScript and JSX.

The screenshot shows a development environment with a code editor and a browser window.

Code Editor (VS Code):

- EXPLORER:** Shows the project structure of "FIRST-APP".
- EDITOR:** Displays the `Home.js` file content:

```
src > pages > Home.js
1 import React, { useState } from "react";
2 import { Link } from "react-router-dom";
3
4 const Home = () => {
5   // Blog yazıları için sahte veri
6   const [blogs] = useState([
7     { id: 1, title: "React ile Blog Yapımı", content: "React ile nasıl blog yapılır?" },
8     { id: 2, title: "Tailwind CSS Nedir?", content: "Tailwind CSS nedir ve nasıl kullanılır?" }
9   ]);
10
11   return (
12     <div>
13       <h1>Blog Sayfası</h1>
14       {blogs.map((blog) => (
15         <div key={blog.id}>
16           <Link to={`/blog/${blog.id}`}>
17             <h2>{blog.title}</h2>
18             <p>{blog.content.substring(0, 50)}...</p>
19           </Link>
20         </div>
21       ))
22     </div>
23   );
24 }
25
26 export default Home;
```

Terminal:

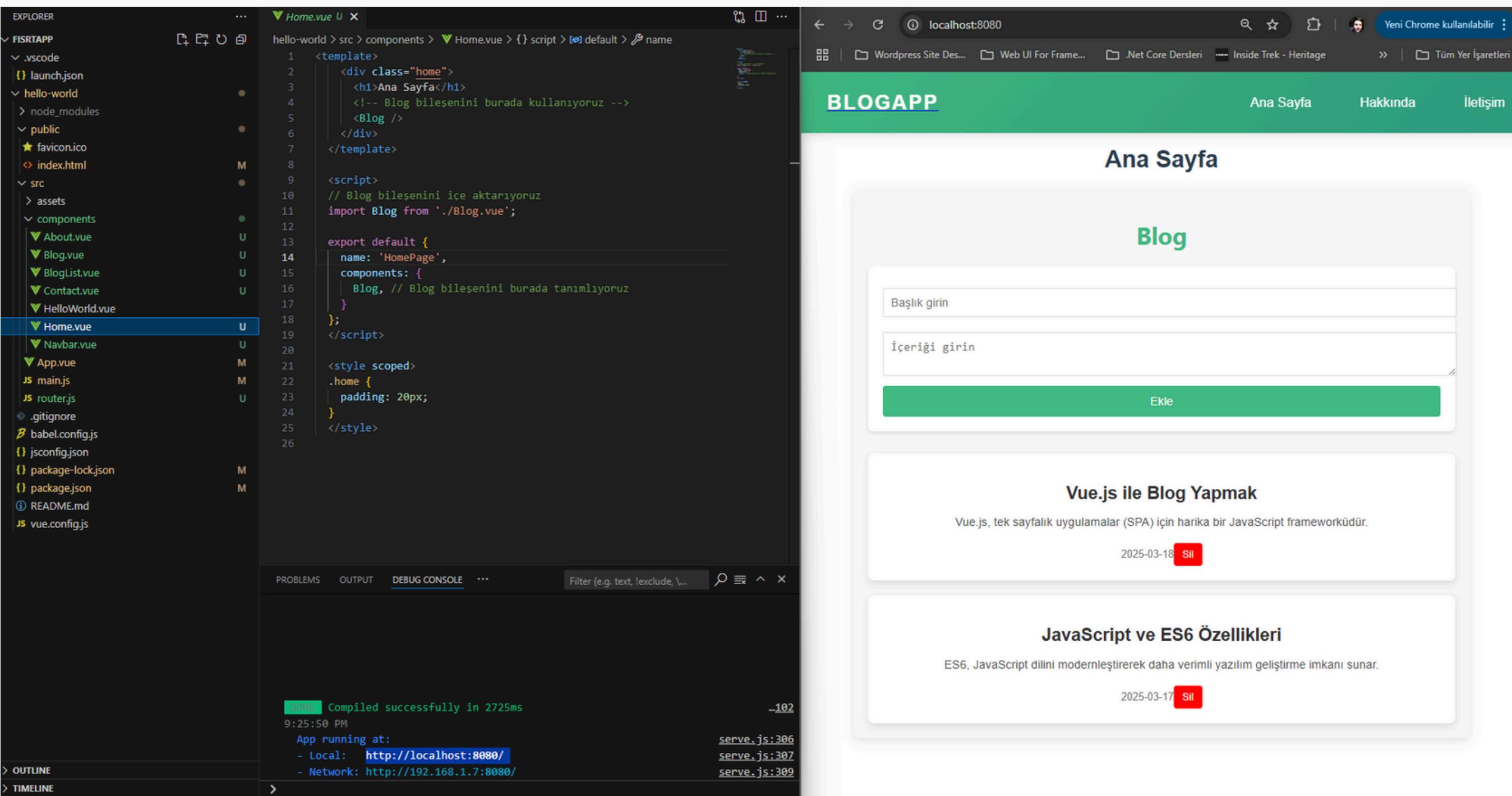
- Shows the build status: "Compiled successfully!"
- Provides local and network URLs: "Local: http://localhost:3001" and "On Your Network: http://172.19.128.1:3001"
- Notes: "Note that the development build is not optimized. To create a production build, use npm run build."
- Final message: "webpack compiled successfully"

Browser:

- Title:** React App
- Address:** localhost:3000
- Content:** My Blog
- Components:**
 - Günlük Haberler:** A list of three news items.
 - Yeni Blog Ekle:** A form with fields for "Başlık" and "İçerik", and a "Ekle" button.
 - İlk Blog Yazım:** A box containing the text "Bu benim ilk blog yazım." with a "Sil" button.
 - React Öğreniyorum:** A box containing the text "React gerçekten çok güclü bir kütüphane!" with a "Sil" button.
 - Hakkında:** A box containing the text "Merhaba! Ben bir blog yazarıyım."

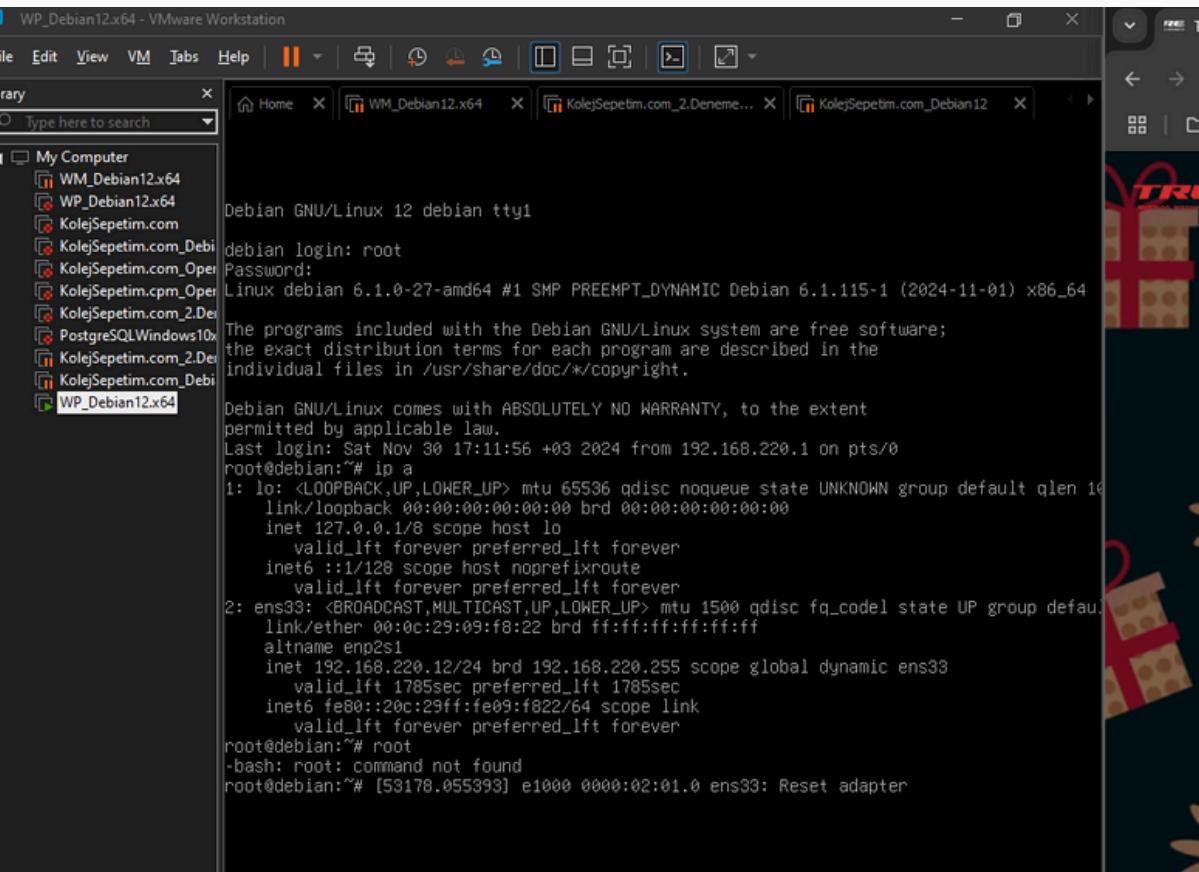
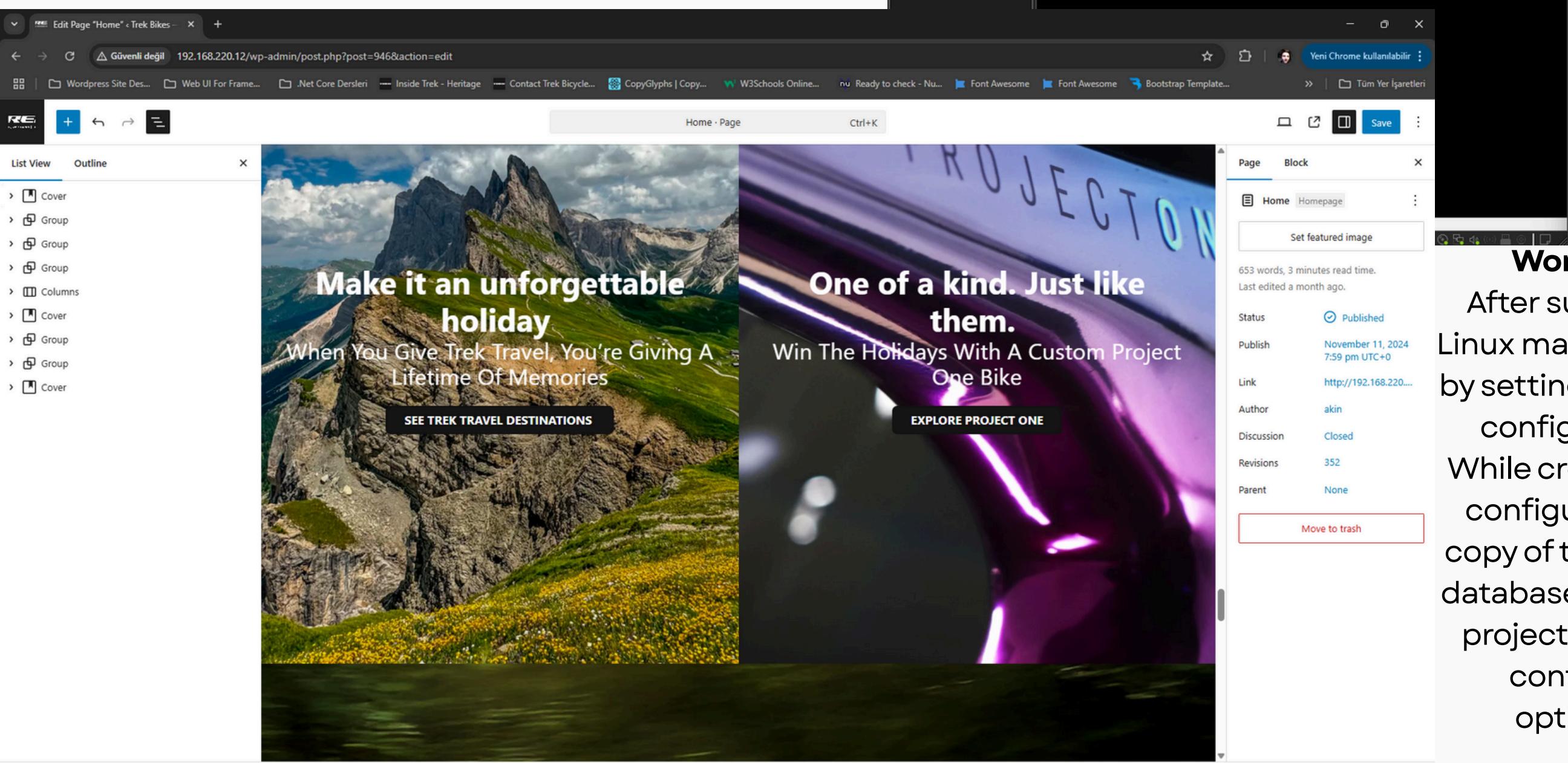
Vue Web Project

The Vue-first-app project is a simple web application developed using the Vue.js framework. This project was created to learn and apply the fundamental concepts of Vue.js. The app uses component-based architecture to offer a modular design, efficiently managing data binding, state management, and user interactions. The project demonstrates Vue.js directives like v-bind, v-for, v-if, and event handling. Additionally, the project, initiated with Vue CLI, creates a dynamic and interactive user interface using modern JavaScript and Vue features.



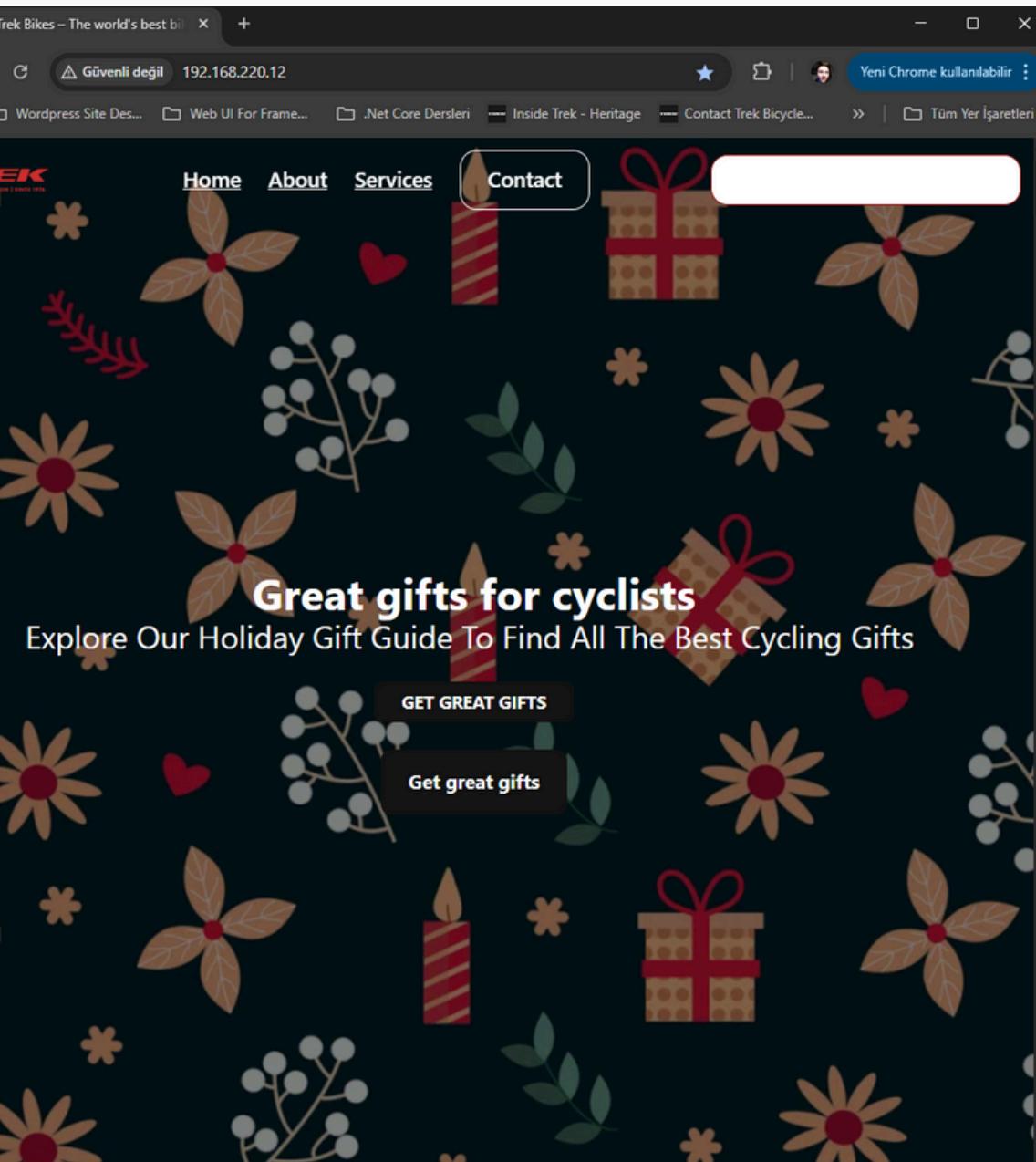
Wordpress Project

During the project, I set up a virtual Linux machine and configured a web server and database management system. I installed the Nginx web server on a Debian-based OS in VMware and routed web traffic. Additionally, I installed MongoDB to provide flexible database management. By working with package managers and system files in the Linux command line, I created a compatible environment.



```
Debian GNU/Linux 12 debian tty1
debian login: root
Password:
Linux debian 6.1.0-27-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.115-1 (2024-11-01) x86_64
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Nov 30 17:11:56 +03 2024 from 192.168.220.1 on pts/0
root@debian:~# ip a
1: lo <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 10
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
        inet6 ::1/128 scope host noprefixroute
            valid_lft forever preferred_lft forever
2: ens33 <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:09:f8:22 brd ff:ff:ff:ff:ff:ff
    altname enp2s1
    inet 192.168.220.12/24 brd 192.168.220.255 scope global dynamic ens33
        valid_lft 1785sec preferred_lft 1785sec
        inet6 fe80::20c:29ff:fe09:f822/64 scope link
            valid_lft forever preferred_lft forever
root@debian:~# root
-bash: root: command not found
root@debian:~# [53178.055393] e1000 0000:02:01.0 ens33: Reset adapter
```

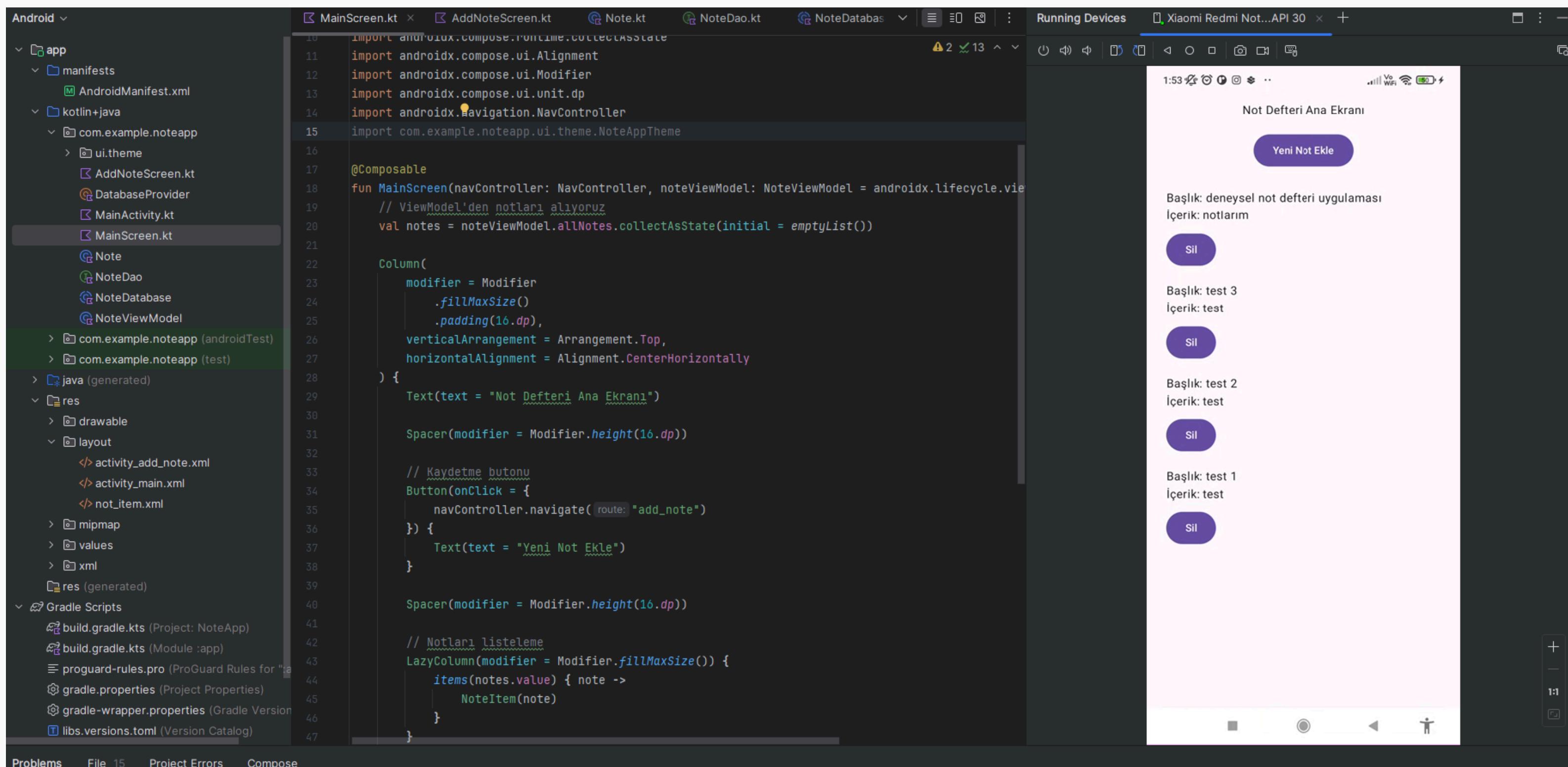


WordPress Installation and Replication Process:

After successfully installing Nginx and MongoDB on the Linux machine, I proceeded with the WordPress installation by setting up the required PHP version and dependencies. I configured Nginx to work seamlessly with WordPress. While creating the replica of the WordPress site, I properly configured the theme and plugins, resulting in an exact copy of the original site. This process included file transfer, database backup, and configuration settings. Through this project, I gained valuable experience in both web server configuration and WordPress management while optimizing system performance for a successful replication.

Android Notepad App

AndroidNoteApp is a user-friendly application designed to simplify note-taking and management. Users can quickly create, edit, and delete notes. The app stores data locally using Activity, RecyclerView, and SQLite technologies, offering an interface in line with Material Design principles. This project provides an opportunity for beginners in Android development to learn fundamental topics such as UI design, database operations, and RecyclerView, effectively managing basic CRUD operations.



Python Web Project

PythonWebProject is a project focused on the process of developing web applications using Python. Built with the Flask web framework, this project includes common features used in web applications, such as basic database operations, routing, HTML templates, and static file management. Data storage is performed with SQLite, and dynamic web pages are presented to the user. This project serves as a foundational example for those wanting to develop Python-based web applications.

The screenshot displays a development environment with two main windows. On the left is a code editor showing a JavaScript file named `index.js` with code related to a blog application. The code handles post retrieval, rendering, and deletion. On the right is a web browser window showing the running application at `localhost:63343/PythonProject/index/index.html`. The browser title is "Blog Sayfası". The page contains a form for "Yeni Blog Yazısı Ekle" (New Blog Post Add) with fields for "Başlık" (Title) and "İçerik" (Content), and a "Yazısı Kaydet" (Save Article) button. Below this is a "Yazılar" (Articles) section listing "Deneysel Python Projesi" and "Test 1", each with a "Sil" (Delete) button. The browser's address bar shows the full URL, and the top status bar indicates the project is shared on GitHub.

```

const postForm = document.getElementById("postForm");
const postTitleInput = document.getElementById("postTitle");
const postContentInput = document.getElementById("postContent");
const postsContainer = document.getElementById("postsContainer");

// Yazzıları yükleme ve gösterme
function loadPosts() {
    const posts = JSON.parse(localStorage.getItem("posts")) || [];
    postsContainer.innerHTML = "";

    posts.forEach((post, index) => {
        const postElement = document.createElement("div");
        postElement.classList.add("post");

        postElement.innerHTML =
            `<h3>${post.title}</h3>
            <p>${post.content}</p>
            <button class="delete-btn" data-index="${index}">Sil</button>`;

        const deleteBtn = postElement.querySelector(".delete-btn");
        deleteBtn.addEventListener("click", () => {
            deletePost(index);
        });

        postsContainer.appendChild(postElement);
    });
}

const deleteBtn = postElement.querySelector(".delete-btn");
deleteBtn.addEventListener("click", () => {
    deletePost(index);
});

postsContainer.appendChild(postElement);

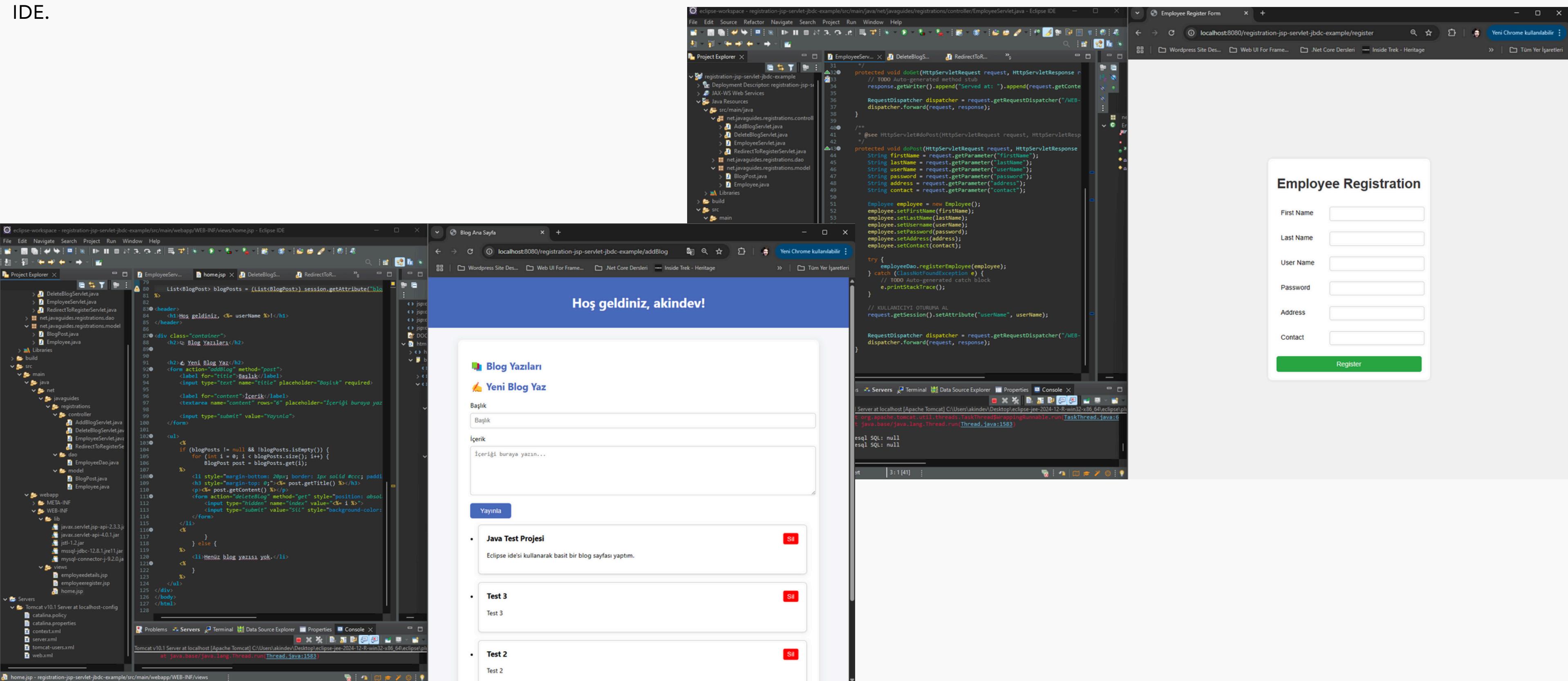
}

// Yazi silme
function deletePost(index) {
    const posts = JSON.parse(localStorage.getItem("posts")) || [];
    posts.splice(index, 1); // Belirtilen index'teki yazıyı sil
    localStorage.setItem("posts", JSON.stringify(posts));
    loadPosts(); // Yeniden yükle
}

```

Java Web Project

IN THIS PROJECT, I DEVELOPED A USER REGISTRATION AND BLOG SYSTEM USING JAVA, JSP, AND SERVLET TECHNOLOGIES. REGISTERED USERS CAN LOG IN TO THE APPLICATION RUNNING ON THE APACHE TOMCAT SERVER. THE HOMEPAGE FEATURES A SIMPLE BLOG INTERFACE WHERE USERS CAN CREATE AND DELETE BLOG POSTS. DATABASE OPERATIONS ARE HANDLED USING JDBC, WITH A CONNECTION TO MS SQL SERVER. THE PROJECT WAS DEVELOPED USING THE ECLIPSE IDE.



SUNAN: AKIN İNCECİK

Thanks!