

Final Module

Summary and Cloud Deployment

דגשים לפרוייקט מסכם

חזרה על נושא API

חזרה על צד לקוח - תקשורת עם API

העלאה לענן

בדיקה שהזריעה של הדטה-בייס עובדת

נפעיל את הפרוייקט - נשתמש בswagger ונבדוק שאנחנו מצליחים לבצע לוגין.

```
modelBuilder.Entity<AppUser>()
```

במידה ויש תקלות - נבדוק את הזריעה של הדטה-בייס

```
.HasData( [
```

```
    new AppUser(){
```

```
        Id = 1,
```

```
        Email = "tomербу@gmail.com",
```

```
        NormalizedEmail = "TOMERBU@GMAIL.COM",
```

```
        UserName = "TomerBu",
```

```
        NormalizedUserName = "TOMERBU",
```

```
        SecurityStamp = Guid.NewGuid().ToString(),
```

```
        PasswordHash = hasher.HashPassword(null, "123456")
```

```
    }
```

```
]);
```

בדיקה שהזריעה של הדטה-בייס עובדת

במידה ויש תקלות - נבדוק את הזריעה של הדטה-בייס

אם זיהינו שזרענו בצורה שגויה/חסרה/בלתי-מספקת:

**drop-database
remove-migration
add-migration SeedAuth
update-database**

הוספת משתמש לבדיקות

נוסיף משתמש בעזרת Register רגיל עם Swagger

המשתמש הרגיל אינו admin ולכן נוכל להשתמש בו לבדיקות שונות

**כעת יש לנו משתמש מסוג Admin
ומשתמש רגיל ללא הרשאות Admin**

נוסיף הגדרות לעבודה עם JWT בספריה Identity בקובץ Program.cs

```
//system configuration for jwt validation:
var jwtSettings = builder.Configuration.GetSection("JwtSettings");
builder.Services.AddAuthentication(options =>
{
    options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;
    options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;
})
.AddJwtBearer(options =>
{
    options.TokenValidationParameters = new TokenValidationParameters
    {
        ValidateLifetime = true,
        ValidateIssuerSigningKey = true,
        ValidIssuer = jwtSettings["Issuer"],
        ValidAudience = jwtSettings["Audience"],
        IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(jwtSettings["SecretKey"]))
    };
});

//end of jwt
```

החלק המרכזי כאן -

בדיקת תקינות של JWT
מתבצעת ע"י יצירת חתימה לחלק הלא מוצפן של הJWT
בעזרת אותו מפתח סתרים שיצר אותו

מקבלים מהלקוח JWT עם 2 חלקים: payload והחתימה.

יוצרים שוב חתימה עם מפתח הסתרים
החתימה שיצרנו כעת חייבת להיות זהה לזו שקיבלנו מהלקוח.

FinalApi/Dtos

```
public class CreateProductDto
{
    [Required]
    [MinLength(1), MaxLength(30)]
    public required string Name { get; set; }

    [Required]
    [MinLength(1), MaxLength(30)]
    public required string Description { get; set; }
    public decimal Price { get; set; }

    [Required]
    [MinLength(1), MaxLength(30)]
    public required string ImageUrl { get; set; }

    [Required]
    [Range(1, 100)]
    public required int CategoryId { get; set; }
}

public static class CreateProductDtoExtensions
{
    public static Product ToProduct(this CreateProductDto dto)
    {
        return new Product
        {
            CategoryId = dto.CategoryId,
            ImageUrl = dto.ImageUrl,
            Name = dto.Name,
            Description = dto.Description,
            Price = dto.Price,
        };
    }
}
```

נרצה לאפשר רק למשתמש רשום להוסיף
מוצרים לדטה-בייס שלנו:

לשם כך ניצור DTO ונראה בעמוד הבא איך
ליישם זאת בController

מתודה שמאפשרת רק למשתמש מחובר להכנס עם JWT

```
public class ProductsController(
    ProductsRepository repository,
    CategoryRepository categoryRepository) : ControllerBase{
    //...
    //CAN ADD PRODUCT ONLY WITH VALID JWT
    [HttpPost]
    [Authorize]
    public ActionResult AddProduct(CreateProductDto dto)
    {
        if (ModelState.IsValid)
        {
            var category = categoryRepository.GetById(dto.CategoryId);
            if (category is null)
            {
                return BadRequest(new { message = "Invalid Category" });
            }
            var product = dto.ToProduct();

            repository.Add(product);
            product.Category = category;
            return CreatedAtAction(nameof(GetProductById), new { id = product.Id }, product.ToDto());
        }
        return BadRequest(ModelState);
    }
}
```

נזריק את הrepository של קטגוריות כדי שנוכל לוודא תקינות על קטגוריה ביצירת מוצר

נבדוק באמצעות Postman ונגלה שרק לקוח רשום יכול להוסיף מוצר
בין אם הוא **admin** ובין אם הוא רק **משתמש רשום**

מתודה שמאפשרת גישה רק לAdmin (אותה מתודה - פשוט שינינו לAdmin)

```
public class ProductsController(
    ProductsRepository repository,
    CategoryRepository categoryRepository) : ControllerBase{
    //...
    //CAN ADD PRODUCT ONLY WITH VALID JWT
    [HttpPost]
    [Authorize(Roles = "admin")]
    public ActionResult AddProduct(CreateProductDto dto)
    {
        if (ModelState.IsValid)
        {
            var category = categoryRepository.GetById(dto.CategoryId);
            if (category is null)
            {
                return BadRequest(new { message = "Invalid Category" });
            }
            var product = dto.ToProduct();

            repository.Add(product);
            product.Category = category;
            return CreatedAtAction(nameof(GetProductById), new { id = product.Id }, product.ToDto());
        }
        return BadRequest(ModelState);
    }
}
```


בדיקה של Claims למשתמש מסויים:
(אותה מתודה - פשוט הוספנו את בדיקת הClaims)

```
public class ProductsController(
    ProductsRepository repository,
    CategoryRepository categoryRepository) : ControllerBase{
    //...
    //CAN ADD PRODUCT ONLY WITH VALID JWT
    [HttpPost]
    [Authorize(Roles = "admin")]
    public ActionResult AddProduct(CreateProductDto dto)
    {
        if (User.Claims.Any(c => c.Type == "isHappy" && c.Value == "true")) {
            Console.WriteLine("Welcome Happy Person");
        }

        if (ModelState.IsValid)
        {
            var category = categoryRepository.GetById(dto.CategoryId);
            if (category is null)
            {
                return BadRequest(new { message = "Invalid Category" });
            }
            var product = dto.ToProduct();

            repository.Add(product);
            product.Category = category;
            return CreatedAtAction(nameof(GetProductById), new { id = product.Id }, product.ToDto());
        }
        return BadRequest(ModelState);
    }
}
```

הגדרות CORS בפרויקט FinalAPI

Program.cs

```
var corsPolicy = "CorsPolicy";

//var host = builder.Configuration.GetValue<string>("AllowedHosts");
builder.Services.AddCors(options =>
{
    options.AddPolicy(name: corsPolicy, policy =>
    {
        policy.WithOrigins([
            "http://localhost:3000",
            "http://localhost:5173",
            "http://localhost:5174",
            //host
        ])
        .AllowAnyHeader()
        .AllowAnyMethod()
        .AllowCredentials();
    });
});

var app = builder.Build();

app.UseCors(corsPolicy);
```

נגדיר כאן את הכתובות של צד הלקוח
את מתודות http שאנו מרשים לצד הלקוח לבצע
וכן אילו headers אנו מרשים ללקוח לשלוח

קוראים לזה מדיניות CORS
וכמובן שדיברנו על זה במודול API

יצירת פרוייקט צד לקוח עם Vite

נפתח את התיקיה של הפרוייקטים עם VSCode

ונכתוב את הפקודות הבאות בטרמינל של VSCode

npm create vite

שם הפרוייקט frontend

ספריה react

שפה typescript

cd frontend

npm install

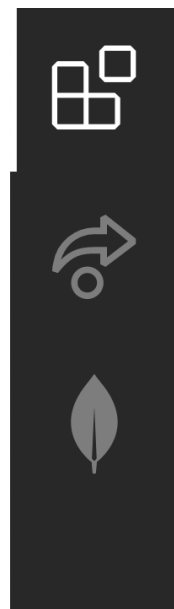
npm run dev



סרגל ניווט בצד לקוח

[Home](#)[About](#)[Products](#)[Login](#)[Register](#)

התקנת הספרייה:
npm i react-router-dom

תוסף מומלץ לVSCode

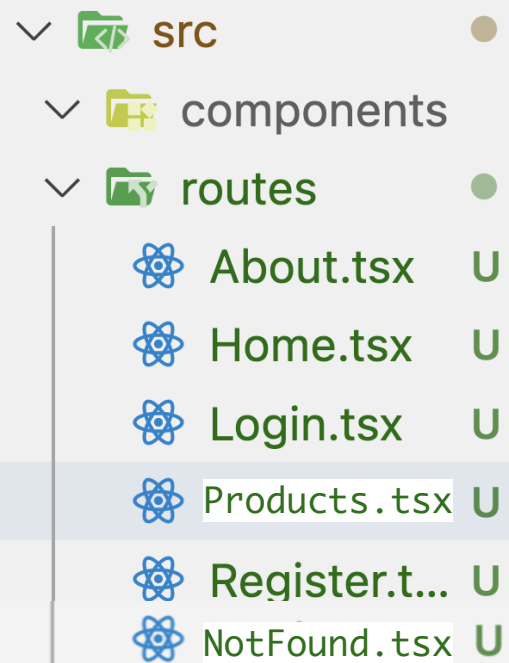


	GoBystrok ReactJS	Install
	React snippets A set of snippets for React runningcoder	🔄 55K ★ 5 Install
	ES7+ React/Redux/React-Native snippets Extensions for React, React-Native and Redux in JS/TS w... dsznajder	🕒 378ms ⚙️

1

קיצור - rafce ליצירת קומפוננטה ועוד קיצורים נחמדים

ניצור כמה קומפוננטות עבור העמודים השונים באפליקציה:



```
const Home = () => {
  return (
    <div>Home</div>
  )
}

export default Home
```

```
const About = () => {
  return (
    <div>About</div>
  )
}

export default About
```

```
const Login = () => {
  return (
    <div>Login</div>
  )
}

export default Login
```

```
const Register = () => {
  return (
    <div>Register</div>
  )
}

export default Register
```

```
const Products = () => {
  return (
    <div>Products </div>
  )
}

export default Products
```

```
import React from 'react'

const NotFound = () => {
  return (
    <div>NotFound</div>
  )
}

export default NotFound
```

עבודה עם הספרייה React-Router-Dom

שלב הSetup

main.tsx

```
import ReactDOM from "react-dom/client";
import App from "./App.tsx";
import "./index.css";
import { BrowserRouter } from "react-router-dom";

const root = document.getElementById("root")!;

//alt shift f (format the code)
ReactDOM.createRoot(root).render(
  <BrowserRouter>
    <App />
  </BrowserRouter>
);
```

הספרייה "תופסת" קישורים ל/about
מעדכנת את רכיבי הUI בהתאם לקישור

כדי שהאפליקציה תוכל להגיב לכל הנתונים
כל האפליקציה עטופה בתגית העוטפת של Router

הגדרת Routes

```
import { Route, Routes } from "react-router-dom";
import Home from "../routes/Home";
import About from "../routes/About";
import Register from "../routes/Register";
import Login from "../routes/Login";
import Posts from "../routes/Products";
import NotFound from "../routes/NotFound";

const App = () => {
  return (
    <>
      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/home" element={<Home />} />
        <Route path="/about" element={<About />} />
        <Route path="/register" element={<Register />} />
        <Route path="/login" element={<Login />} />
        <Route path="/products" element={<Products />} />
        <Route path="*" element={<NotFound />} />
      </Routes>
    </>
  );
};

export default App;
```


ניצור קומפוננטה עבור סרגל ניווט:

src
components
Navbar.tsx

```
import { NavLink } from "react-router-dom";

const Navbar = () => {
  return (
    <nav>
      <NavLink to="/home">Home</NavLink>
      <NavLink to="/about">About</NavLink>
      <NavLink to="/posts">Posts</NavLink>
      <NavLink to="/login">Login</NavLink>
      <NavLink to="/register">Register</NavLink>
    </nav>
  );
};

export default Navbar;
```

קיצורי מקשים:

alt shift f

פרמוט המסמך

alt ↓

הורדת שורה למטה

alt shift ↓

שכפול שורה

ctrl .

quick fix

(מקביל לalt enter)

**המשך בשיעור הבא -
נתראה!**