

> Ficha Prática Nº 5

Objetivos:

- Introdução à segurança em aplicações web
- Introdução à segurança em aplicações web ASP.NET Core
- Introdução ao Microsoft Identity Core

> Parte I – Conceitos

>> Segurança em Sistemas de Informação

[HTTPS://PT.WIKIPEDIA.ORG/WIKI/SEGURAN%C3%A7A_DA_INFORMA%C3%A7%C3%A3o](https://pt.wikipedia.org/wiki/Seguran%C3%A7a_da_informa%C3%A7%C3%A3o)

A definição mais formal de segurança de sistemas de informação é o ato ou prática de proteger os sistemas de informação contra acesso, uso, modificação, destruição ou interrupção não autorizados.

Esta definição está, desde os anos 60, associada à confidencialidade, integridade e disponibilidade, também chamada como tríade da CIA (confidentiality, integrity, and availability - 3 pilares da segurança da informação). Mais recentemente e, devido à evolução e complexidade dos sistemas de informação/tecnologias/meios, associou-se um outro conceito, a autenticidade da informação.

Podemos então dizer que a segurança da informação tem como principais objetivos garantir os níveis adequados de integridade, autenticidade, disponibilidade e confidencialidade, por forma a mitigar o impacto de eventuais incidentes que possam comprometer o regular funcionamento dos sistemas de informação.

De uma forma resumida os 4 princípios da segurança em Sistemas de informação são:

- **Confidencialidade** - capacidade de proteger os dados daqueles que não estão autorizados a consultá-los.
- **Integridade** - capacidade de prevenir, recuperar e reverter alterações não autorizadas ou acidentais aos dados.
- **Disponibilidade** - Possibilidade de acesso aos dados de pessoas autorizadas e em tempo útil.
- **Autenticidade** - manutenção da fiabilidade da informação desde o momento da sua produção e ao longo de todo o seu ciclo de vida.

>> Segurança de aplicações web

[HTTPS://WWW.RFC-EDITOR.ORG/RF/RF9110.HTML#NAME-SECURITY-CONSIDERATIONS](https://www.rfc-editor.org/rfc/rfc9110.html#name-security-considerations)

[HTTPS://OWASP.ORG/](https://owasp.org/)

[HTTPS://DEVELOPER.MOZILLA.ORG/PT-BR/DOCS/LEARN/SERVER-SIDE/FIRST_STEPS/WEBSITE_SECURITY](https://developer.mozilla.org/pt-BR/docs/Learn/Server-side/First_steps/Website_security)

[HTTPS://WWW.CNCS.GOV.PT/PT/REGIME-JURIDICO/](https://www.cncs.gov.pt/pt/regime-juridico/)

Alguns dos tipos de ataque mais comuns:

- Cross site scripting (XSS) — vulnerabilidade que permite que um atacante injete scripts do lado do cliente numa página da internet para aceder a informações importantes diretamente, personificar o utilizador ou induzir o utilizador a revelar informações importantes.
- SQL injection (SQi) — injeção de SQL é um método pelo qual um atacante explora vulnerabilidades na maneira como a aplicação executa operações numa base de dados.
- Ataques de negação de serviço (DoS) e ataques de negação de serviço distribuída (DDoS) — por meio de uma variedade de vetores, os invasores conseguem sobrecarregar o servidor afetado ou sua infraestrutura circundante com diferentes tipos de tráfego de ataque. Quando um servidor perde a capacidade de processar com eficácia as solicitações recebidas, começa a comportar-se com lentidão e, eventualmente, passa a não conseguir responder / negar serviço às solicitações recebidas.
- Corrupção de memória — a corrupção da memória ocorre quando um local na memória é acidentalmente modificado, resultando em um possível comportamento inesperado do software. Os agentes mal-intencionados tentarão farejar e explorar a corrupção da memória por meio de ataques como injeções de código ou buffer overflow.
- Buffer overflow — o é uma anomalia que ocorre quando o software grava dados num espaço definido na memória conhecido como buffer. Se a capacidade do buffer for excedida, isso faz com que os dados em locais de memória adjacentes sejam substituídos. Esse comportamento pode ser explorado para injetar códigos maliciosos na memória, possivelmente criando uma vulnerabilidade na máquina afetada.
- Falsificação de solicitações entre sites (CSRF) — a falsificação de solicitações entre sites tem como objetivo induzir uma vítima a fazer uma solicitação que utiliza sua autenticação ou autorização. Ao tirar proveito dos privilégios de conta de um utilizador, um atacante é capaz de enviar uma solicitação como se fosse o utilizador. Após comprometer a conta de um utilizador, o invasor consegue extrair, destruir ou alterar informações importantes. As contas com muitos privilégios, como as de administradores ou executivos, são as mais frequentemente atacadas.
- Violação de dados — diferentemente dos vetores de ataque específicos, "violação de dados" é um termo generalizado que se refere à liberação de informações sensíveis ou confidenciais, que pode ocorrer por meio de ações mal-intencionadas ou por engano. O âmbito daquilo que

é considerado como uma violação de dados é bastante amplo e pode incluir desde alguns poucos registos, altamente valiosos, até milhões de contas de utilizadores expostas.

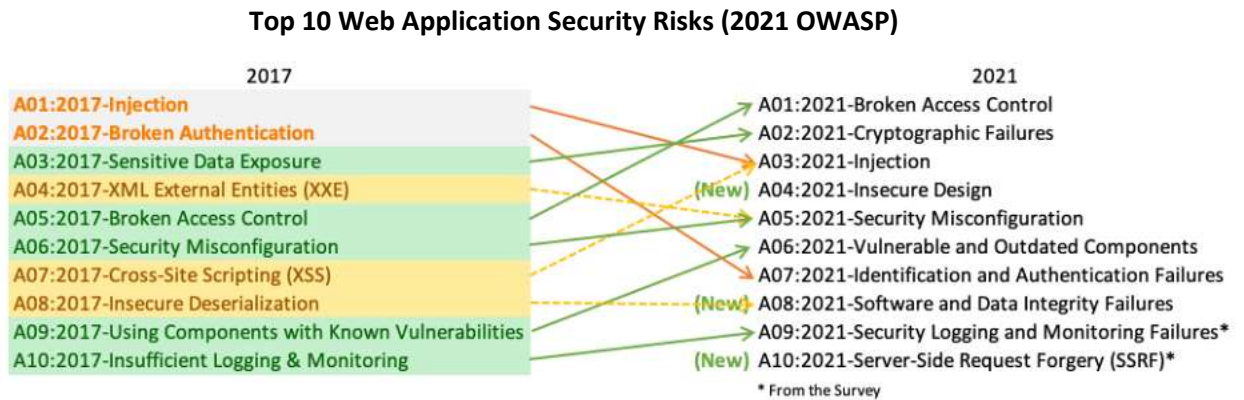


Figura 1 - Falhas de segurança mais comuns no ano 2021 – comparativo a 2017

Fonte: <https://owasp.org/www-project-top-ten/>

Como implementar os pilares da Segurança de SI em aplicações web?

- Confidencialidade
- Integridade
- Disponibilidade
- Autenticidade

>> Microsoft Identity Core

[HTTPS://LEARN.MICROSOFT.COM/EN-US/ASPNET/CORE/SECURITY/?VIEW=ASPNETCORE-6.0](https://learn.microsoft.com/en-us/aspnet/core/security/?view=aspnetcore-6.0)

[HTTPS://LEARN.MICROSOFT.COM/EN-US/ASPNET/CORE/SECURITY/AUTHENTICATION/IDENTITY?VIEW=ASPNETCORE-6.0&TABS=VISUAL-STUDIO](https://learn.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-6.0&tabs=visual-studio)

[HTTPS://LEARN.MICROSOFT.COM/EN-US/ASPNET/CORE/SECURITY/AUTHORIZATION/ROLES?VIEW=ASPNETCORE-6.0](https://learn.microsoft.com/en-us/aspnet/core/security/authorization/roles?view=aspnetcore-6.0)

- É uma API que dá suporte à funcionalidade de logon da interface do utilizador.
- Permite gerir utilizadores, senhas, dados de perfil, funções, declarações, tokens, confirmação de email e muito mais.
- Normalmente é configurado utilizando uma base de dados SQL Server para armazenar nomes de utilizador, senhas e dados de perfil. Como alternativa, podem ser utilizados outros repositórios persistentes, como por exemplo, o Azure, MariaDB, etc..

> Exercícios

- IMPLEMENTAR MICROSOFT IDENTITY CORE NO PROJECTO
- CRIAR / MODIFICAR FORMULÁRIOS DE LOGIN E REGISTO DOS UTILIZADORES
- PERSONALIZAR A CLASSE IDENTITYUSER
- MARCAR PROPRIEDADES COMO PERSONALDATA
- ASSOCIAR UM AGENDAMENTO A UM UTILIZADOR

>> Microsoft ASP.Net Identity Core – Configuração Inicial

1. Analise com o docente as configurações do projeto por forma a verificar a configuração do Microsoft Identity Core.
2. Crie um utilizador através do formulário de registo de utilizadores disponibilizado.
3. Verifique na base de dados, nas tabelas relativas ao *Microsoft Identity - “aspnet*”*, os registos adicionados.
4. Faça login do site com os dados do utilizador que criou no ponto 2.
5. Verifique as alterações no menu superior.
6. Analise o conteúdo do ficheiro “*_loginPartial.cshtml*”.
7. Navegue pelas opções disponibilizadas (quando clica no nome de utilizador no menu superior).
8. Pesquise no projeto pelas “páginas” responsáveis pela geração dos formulários de **Login**, **Registo** e **MyAccount**.

>> Microsoft ASP.Net Identity Core – Personalização do Modelo IdentityUser

8. Crie uma classe chamada **ApplicationUser** que herda a classe **IdentityUser** e adicione as seguintes propriedades.
 - PrimeiroNome - *string*
 - UltimoNome - *string*
 - DataNascimento - *DateTime*
 - NIF – *int*

9. Faça as alterações necessárias ao projeto para utilizar esta classe em vez da classe **IdentityUser**.

Nota:

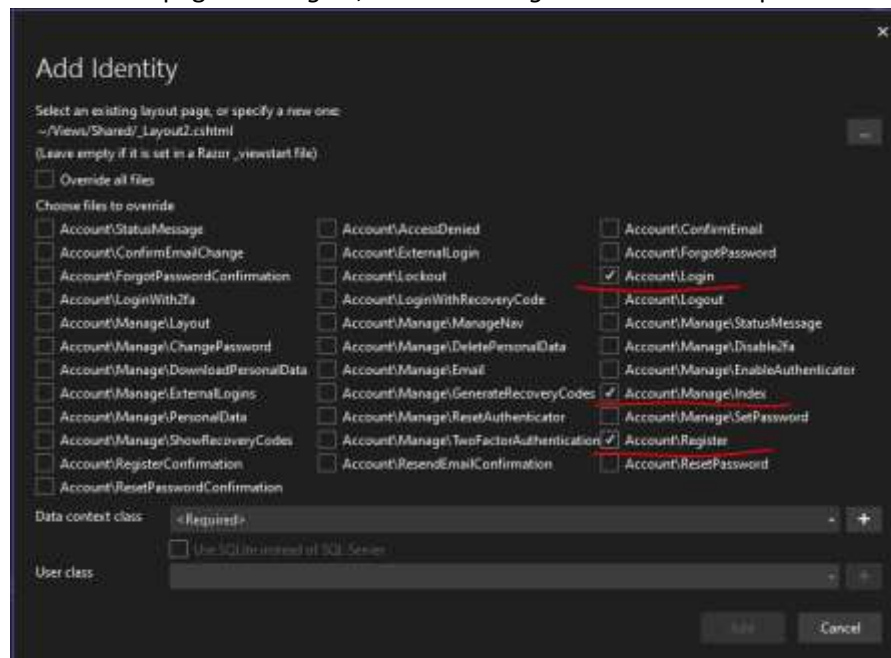
- Para isso necessita de modificar os ficheiros **Program.cs** e **ApplicationDbContext.cs**

10. Verifique se tem instalado o seguinte pacote NuGet:

Install-Package Microsoft.VisualStudio.Web.CodeGeneration.Design

11. Gere as seguintes páginas *Razor*, relativas ao *Identity*, através da opção **“Add » New Scaffolded Item » Identity”**.

Selecione as páginas: *“Login”*, *“Index”* e *“Register”* conforme a próxima imagem.



[HTTPS://LEARN.MICROSOFT.COM/EN-US/ASPNET/CORE/SECURITY/AUTHENTICATION/SCAFFOLD-IDENTITY?VIEW=ASPNETCORE-6.0&TABS=VISUAL-STUDIO](https://learn.microsoft.com/en-us/aspnet/core/security/authentication/scaffold-identity?view=aspnetcore-6.0&tabs=visual-studio)

12. Modifique a página de Registo de Utilizadores por forma a utilizar o modelo personalizado. Ou seja, no registo devem ser solicitados ao utilizador as propriedades que constam na classe **“ApplicationUser”**.

- Verifique se as propriedades *PrimeiroNome*, *UltimoNome*, *Data de Nascimento* e *NIF* são guardados na base de dados.

13. Modifique a página de perfil do utilizador por forma a utilizar o modelo personalizado.

- Verifique se as propriedades *PrimeiroNome*, *UltimoNome*, *Data de Nascimento* e *NIF* são guardados na base de dados, quando efectua alterações.

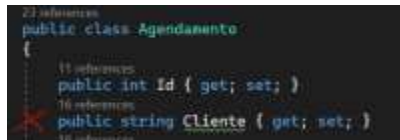
>> Microsoft ASP.Net Identity Core – Relacionamento com outras entidades

Nos próximos exercícios iremos fazer com os pedidos de agendamento apenas possam ser efetuados por utilizadores registados e autenticados.

Para isso iremos criar um relacionamento entre um agendamento e um utilizador. (*1 Utilizador – N Agendamentos*), bem como alterar os controllers e views que suportam esta funcionalidade.

14. Elimine todos os agendamentos existentes na base de dados.

15. Remova a propriedade **Cliente** da classe **Agendamento**, bem como todas as referências para esta propriedade (Views, ViewModels, Controller).



```
23 agendamentos
public class Agendamento
{
    15 agendamentos
    public int Id { get; set; }
    16 agendamentos
    public string Cliente { get; set; }
    17 agendamentos
}
```

16. Modifique as classes **Agendamento** e **ApplicationUser** por forma a definir o relacionamento anteriormente referido (*1 Utilizador – N Agendamentos*). Defina o relacionamento na sua totalidade:

Convenção 4 - Definir a relação na sua totalidade em ambas as entidades, indicando, além as propriedades de navegação, a chave estrangeira na entidade dependente.

17. Crie uma Migração e aplique-a.

18. Altere a View **“Pedido”**, do controller **Agendamento**, por forma a mostrar o formulário apenas aos utilizadores autenticados.

19. Mostre a seguinte mensagem no caso de os utilizadores não estarem autenticados.

Só utilizadores autenticados é que podem efetuar agendamentos. Por favor, autentique-se.

20. Faça as alterações necessárias no controller **Agendamento** para guardar o “utilizador” quando cria um agendamento.

21. Crie uma funcionalidade que permita mostrar os agendamentos do utilizador (***Os meus agendamentos***).

22. Crie uma opção no menu superior que “aponte” para esta funcionalidade.

- Esta opção só deve estar visível se o utilizador estiver autenticado.
23. Faça as alterações necessárias no ***controller Agendamento*** e respetivas vistas de forma a mostrar os seguintes dados do utilizador (PrimeiroNome, UltimoNome, Username) na Lista de Agendamentos e nos detalhes de um agendamento.