

Spisak urađenih korektivnih akcija:

- Dodana dokumentacija za AdminPanelController

Nedostatak dokumentacije za AdminPanel Controller		😊 Open
DESCRIPTION		
☰ Nedostatak dokumentacije za klasu i metode		
LINKED ISSUES		
🐙 GitHub https://github.com/aljubuncic/vvs-project/issues/3		
📄 Burgija/Controllers/AdminPanelController.cs		⚙️
15. public class AdminPanelController : Controller		
📄 Burgija/Controllers/AdminPanelController.cs		⚙️
26. private async Task InitializeLists()		
📄 Burgija/Controllers/AdminPanelController.cs		⚙️
37. // GET: AdminPanel 38. public async Task<IActionResult> Index()		
📄 Burgija/Controllers/AdminPanelController.cs		⚙️
50. // GET: AdminPanel/Details/5 51. public async Task<IActionResult> Details(i		
📄 Burgija/Controllers/AdminPanelController.cs		⚙️
72. // GET: AdminPanel/AddTool		

51	<code>public async Task<ActionResult> Details(int? id)</code>	62	<code>public async Task<ActionResult> Details(int? id)</code>
52	<code>{</code>	63	<code>{</code>
53	<code> if (id == null)</code>	64	<code> if (id == null)</code>
54	<code> {</code>	65	<code> {</code>
55	<code> return NotFound();</code>	66	<code> return NotFound();</code>
56	<code> }</code>	67	<code> }</code>
57	-		
58	<code>var tool = await _context.Tool</code>	68	<code>var tool = await _context.Tool</code>
59	<code>.Include(t => t.Store)</code>	69	<code>.Include(t => t.Store)</code>
60	<code>.Include(t => t.ToolType)</code>	70	<code>.Include(t => t.ToolType)</code>
+	<code>@@ -69,7 +79,9 @@ public async Task<ActionResult> Details(int? id)</code>		
69	<code> return View(tool);</code>	79	<code> return View(tool);</code>
70	<code>}</code>	80	<code>}</code>
71		81	
72	- <code>// GET: AdminPanel/AddTool</code>	82	+ <code>/// <summary></code>
		83	+ <code>/// Displays a form for adding a new tool in the admin panel.</code>
		84	+ <code>/// </summary></code>
73	<code>public async Task<ActionResult> AddTool()</code>	85	<code>public async Task<ActionResult> AddTool()</code>
74	<code>{</code>	86	<code>{</code>
75	<code> await InitializeLists();</code>	87	<code> await InitializeLists();</code>
+	<code>@@ -78,11 +90,9 @@ public async Task<ActionResult> AddTool()</code>		
78	<code> return View();</code>	90	<code> return View();</code>
79	<code>}</code>	91	<code>}</code>
80		92	
81	-	93	+ <code>/// <summary></code>
82	-	94	+ <code>/// Handles the submission of the form for adding a new tool.</code>
83	- <code>// POST: AdminPanel/AddTool</code>	95	+ <code>/// </summary></code>

- Dodana dokumentacija za RentController

Nedostatak dokumentacije u Rent controlleru

Open

DESCRIPTION

Nedostatak adekvatne dokumentacije za klasu i metode

LINKED ISSUES

GitHub <https://github.com/aljubuncic/vvs-project/issues/2>

Burgija/Controllers/RentController.cs

19. public class RentController : Controller

Burgija/Controllers/RentController.cs

30. [Authorize]
31. public async Task<IActionResult> RentHistory

Burgija/Controllers/RentController.cs

42. [HttpPost]
43. public IActionResult GetToolType(int? toolTy

Burgija/Controllers/RentController.cs

55. // GET: Rent/Create
56. public async Task<IActionResult> Create(int?

16 namespace Burgija.Controllers 17 { 18 19 public class RentController : Controller 20 { 21 private readonly ApplicationDbContext _context; 22 private readonly UserManager<IdentityUser<int>> _userManager; 23 24 public RentController(ApplicationDbContext context, UserManager<IdentityUser<int>> userManager) 25 { 26 _context = context; 27 _userManager = userManager; 28 } 29 30 [Authorize] 31 public async Task<IActionResult> RentHistory() 32 { 33 var userId = Int32.Parse(User.FindFirst(ClaimTypes.NameIdentifier).Value);	16 namespace Burgija.Controllers 17 { 18 19 + /// <summary> 20 + /// Controller responsible for handling rental-related actions. 21 + /// </summary> 22 public class RentController : Controller 23 { 24 private readonly ApplicationDbContext _context; 25 private readonly UserManager<IdentityUser<int>> _userManager; 26 27 + /// <summary> 28 + /// Initializes a new instance of the <see cref="RentController"/> class. 29 + /// </summary> 30 + /// <param name="context">The application database context.</param> 31 + /// <param name="userManager">The user manager for identity management.</param> 32 public RentController(ApplicationDbContext context, UserManager<IdentityUser<int>> userManager) 33 { 34 _context = context; 35 _userManager = userManager; 36 } 37 38 + /// <summary> 39 + /// Displays the rental history for the authenticated user. 40 + /// </summary> 41 + /// <returns>The rental history view.</returns> 42 [Authorize] 43 public async Task<IActionResult> RentHistory() 44 { 45 + 46 var userId = Int32.Parse(User.FindFirst(ClaimTypes.NameIdentifier).Value);
---	--

- Dodana dokumentacija za HomeController

Nedostatak dokumentacije

Open

DESCRIPTION

Nedostatak dokumentacije za kontroler HomeController.

LINKED ISSUES

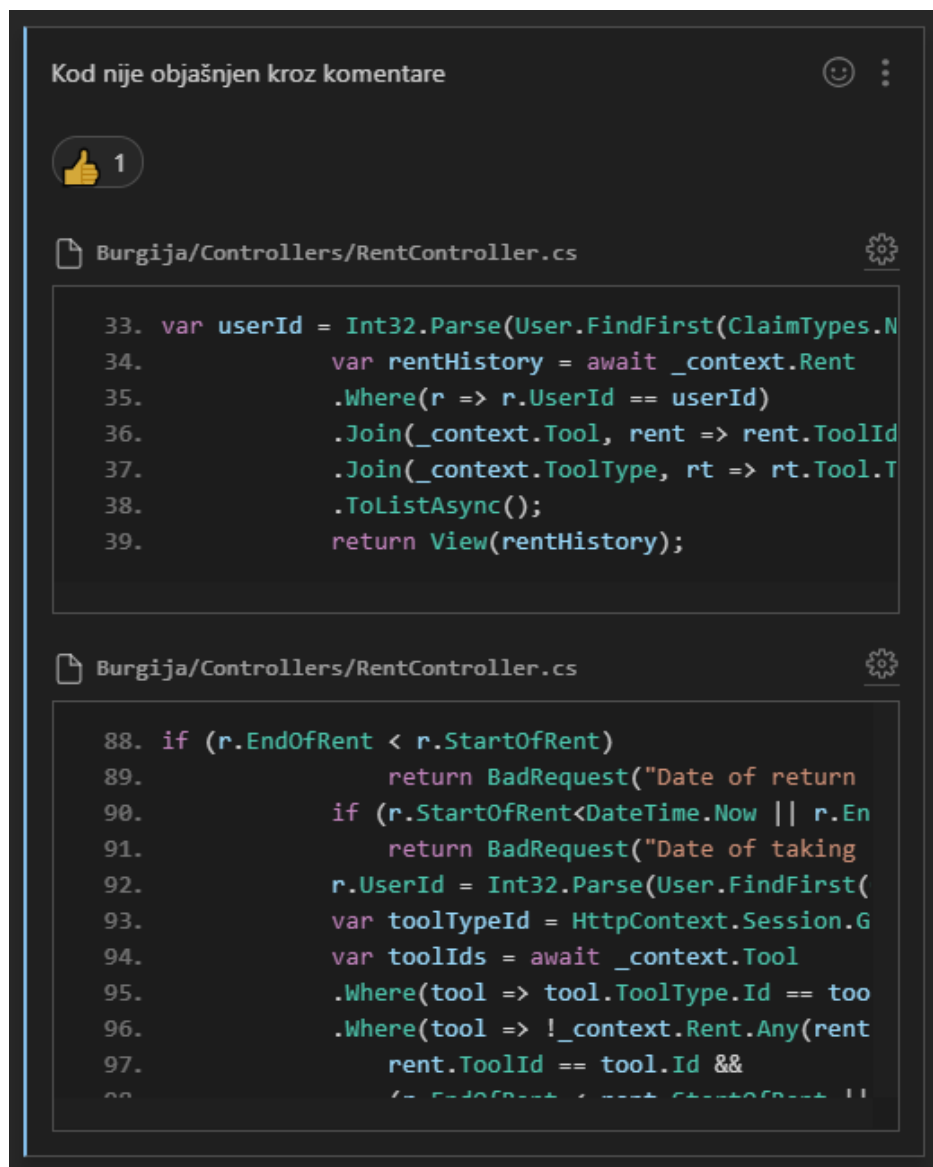
GitHub <https://github.com/aljubuncic/vvs-project/issues/8>

Burgija/Controllers/HomeController.cs

22. public class HomeController : Controller

<pre> 94 - 95 public Task<ActionResult> FilterTools(double? priceFrom, double? priceTo, string sortOptions) 96 { 97 priceFrom ??= 0; 98 priceTo ??= 10000; 99 var queryParameters = new Dictionary<string, string> 100 { 101 { "pricefrom", priceFrom.ToString() }, 102 { "priceto", priceTo.ToString() }, 103 { "sortOptions", sortOptions }; 104 }; 105 return Task.FromResult<ActionResult>(RedirectToAction("Index", queryParameters)); 106 } 107 108 private static List<ToolType> LinearSearch(List<ToolType> toolTypes, string search) 109 { 110 if (string.IsNullOrWhiteSpace(search)) 111 return toolTypes; 112 } </pre>	<pre> 87 + /// <summary> 88 + /// Handles filtering of tools based on price range and sorting options. 89 + /// </summary> 90 public Task<ActionResult> FilterTools(double? priceFrom, double? priceTo, string sortOptions) 91 { 92 priceFrom ??= 0; 93 priceTo ??= 10000; 94 var queryParameters = new Dictionary<string, string> 95 { 96 { "pricefrom", priceFrom.ToString() }, 97 { "priceto", priceTo.ToString() }, 98 { "sortOptions", sortOptions }; 99 }; 100 return Task.FromResult<ActionResult>(RedirectToAction("Index", queryParameters)); 101 } 102 103 + /// <summary> 104 + /// Performs a case-insensitive linear search for tool types containing the specified search term. 105 + /// </summary> 106 + /// <param name="toolTypes">The list of tool types to search.</param> 107 + /// <param name="search">The search term.</param> 108 + /// <returns>A list of tool types matching the search term.</returns> 109 private static List<ToolType> LinearSearch(List<ToolType> toolTypes, string search) 110 { 111 if (string.IsNullOrWhiteSpace(search)) </pre>
---	--

- Adekvatno komentarisano kod u metodama prethodno spomenutih klasa



63	<code>public async Task<IActionResult> Create(int? typeId)</code>	90	<code>/// </returns></code>
64	<code>{</code>	91	<code>public async Task<IActionResult> Create(int? typeId)</code>
65	<code>if (typeId == null)</code>	92	<code>{</code>
66	<code>{</code>	93	<code>// Check if the tool type ID is null</code>
67	<code>return NotFound();</code>	94	<code>if (typeId == null)</code>
68	<code>}</code>	95	<code>{</code>
69	<code></code>	96	<code>// If tool type ID is null, return a "NotFound" result</code>
70	<code>// Retrieve the tool type using the provided ID</code>	97	<code>return NotFound();</code>
71	<code>var toolType = await _context.ToolType.FirstOrDefault(m</code>	98	<code>}</code>
72	<code>=> m.Id == typeId);</code>	99	<code></code>
73	<code>if (toolType == null)</code>	100	<code>// Retrieve the tool type with the specified ID from the</code>
74	<code>{</code>	101	<code>database</code>
75	<code>return NotFound();</code>	102	<code>var toolType = await _context.ToolType.FirstOrDefault(m</code>
76	<code>}</code>	103	<code>=> m.Id == typeId);</code>
77	<code></code>	104	<code>// Check if the tool type with the specified ID is not found</code>
78	<code>ViewBag.ToolTypeImage = toolType.Image;</code>	105	<code>if (toolType == null)</code>
79	<code>ViewBag.ToolType = toolType;</code>	106	<code>{</code>
80	<code></code>	107	<code>// If tool type is not found, return a "NotFound" result</code>
81	<code></code>	108	<code>return NotFound();</code>
82	<code>return View();</code>	109	<code>}</code>
		110	<code>// Set ViewBag properties for the view</code>
		111	<code>ViewBag.ToolTypeImage = toolType.Image;</code>
		112	<code>ViewBag.ToolType = toolType;</code>
		113	<code></code>
		114	<code>// Return the view for creating a new rental with details</code>
		115	<code>related to the specified tool type</code>
			<code>return View();</code>

- Dodana provjera parametara konstruktora u modelima

Defensive Programming Issue

DESCRIPTION

Nedostaje validacija parametara u konstruktorima svih modela.

Burgija/Models/ToolType.cs

07. public class ToolType : IComparable<ToolType>

Burgija/Models/Tool.cs

09. public class Tool

Burgija/Models/Review.cs

08. public class Review

Burgija/Models/Rent.cs

08. public class Rent : IComparable<Rent>

Burgija/Models/RegisteredUser.cs

06. public class RegisteredUser : User

Burgija/Models/Location.cs

07. public class Location


<pre> 7 #region Constructors 8 9 public Administrator(int id, string username, string email, string password) 10 { 11 Id=id; 12 13 Username = username; 14 Email=email; 15 Password=password; 16 } 17 18 public Administrator() { } 19 #endregion </pre>	<pre> 7 #region Constructors 8 9 public Administrator(int id, string username, string email, string password) 10 { 11 if (id <= 0) 12 { 13 throw new ArgumentOutOfRangeException(nameof(id), "Id mora biti veći od 0."); 14 } 15 16 if (string.IsNullOrEmpty(username)) 17 { 18 throw new ArgumentException("Username ne može biti null ili prazno.", nameof(username)); 19 } 20 21 if (string.IsNullOrEmpty(email)) 22 { 23 throw new ArgumentException("Email ne može biti null ili prazno.", nameof(email)); 24 } 25 26 if (string.IsNullOrEmpty(password)) 27 { 28 throw new ArgumentException("Password ne može biti null ili prazno.", nameof(password)); 29 } 30 31 Id = id; 32 Username = username; 33 Email = email; 34 Password = password; 35 } 36 37 public Administrator() { } 38 #endregion </pre>
--	---

- Reorganizacija koda u metodi Index klase HomeController

<pre> 58 return View(await _context.ToolType.ToListAsync()); 59 } 60 61 if (search != null) 62 { 63 var toolTypes = await _context.ToolType.ToListAsync(); 64 var searchResults = LinearSearch(toolTypes, search); 65 return View(searchResults); 66 } 67 68 List<ToolType> filterResults = await _context.ToolType.ToListAsync(); 69 70 if (priceFrom != null && priceTo != null) 71 { 72 filterResults = LinearSearchByPrice(filterResults, (double)priceFrom, 73 (double)priceTo); 74 } 75 76 if (!string.IsNullOrEmpty(sortOptions)) 77 { 78 switch (sortOptions) </pre>	<pre> 58 return View(await _context.ToolType.ToListAsync()); 59 } 60 61 List<ToolType> filterResults = await _context.ToolType.ToListAsync(); 62 63 if (priceFrom != null && priceTo != null) 64 { 65 filterResults = LinearSearchByPrice(filterResults, (double)priceFrom, 66 (double)priceTo); 67 } 68 69 if (search != null) 70 { 71 filterResults = LinearSearch(filterResults, search); 72 } 73 74 if (!string.IsNullOrEmpty(sortOptions)) 75 { 76 switch (sortOptions) </pre>
--	---

- Uklonjen redundantni atribut _userManager u klasi RentController

@aljubuncic: _userManager se nigdje ne koristi u klasi, redundantan atribut

 1

Burgija/Controllers/RentController.cs

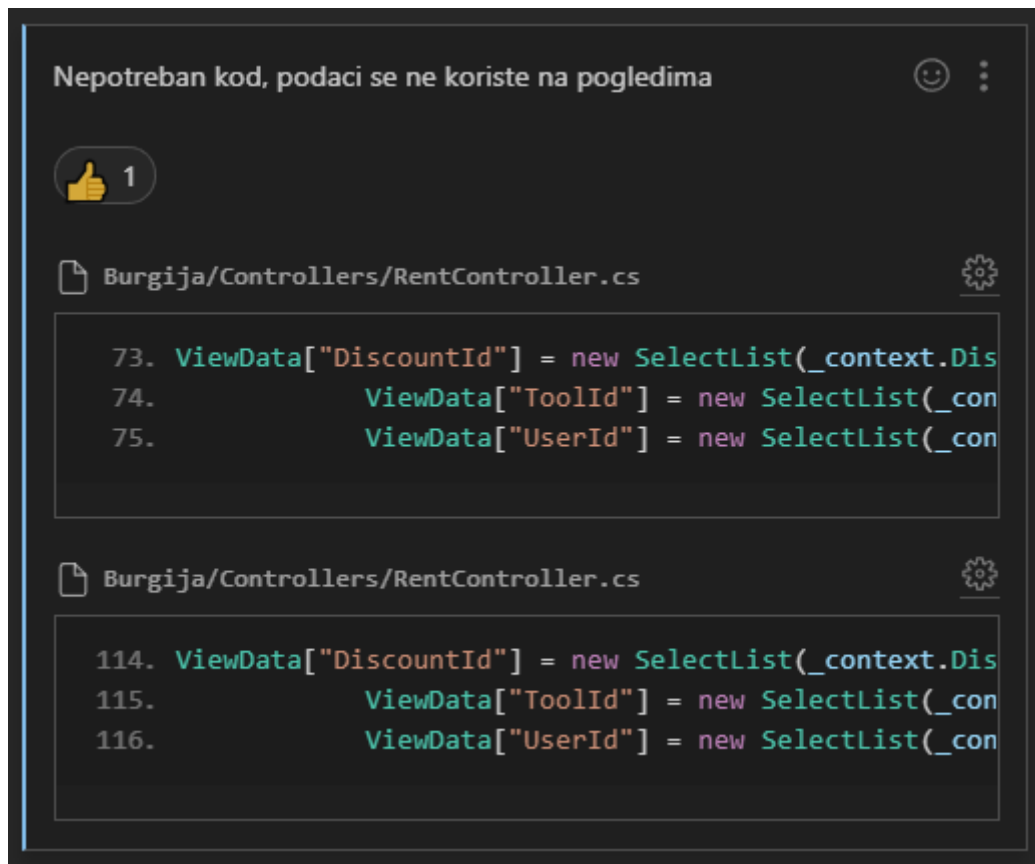
```

22. private readonly UserManager<IdentityUser<int>> _u
23.
24.     public RentController(ApplicationDbContext
25.     {
26.         _context = context;
27.         _userManager = userManager;
28.     }

```

19	public class RentController : Controller	19	public class RentController : Controller
20	{	20	{
21	private readonly ApplicationDbContext _context;	21	private readonly ApplicationDbContext _context;
22	private readonly UserManager<IdentityUser<int>> _userManager;	22	
23		23	public RentController(ApplicationDbContext context)
24	public RentController(ApplicationDbContext context, UserManager<IdentityUser<int>> userManager)	24	{
25	{	25	_context = context;
26	_context = context;	26	}
27	_userManager = userManager;		
28	}		

- Uklonjene redundantne linije koda u klasi RentController



103	else	125	else
104	return BadRequest("There are no tools available in this period");	126	{
		127	// Return a BadRequest response if no tools are available in the specified period
		128	return BadRequest("There are no tools available in this period");
		129	}
105	r.DiscountId = null;	130	
		131	// To be implemented: Set the DiscountId (currently set to null as a placeholder)
		132	r.DiscountId = null;
106	var toolType = await _context.ToolType.FindAsync(toolTypeId);	133	
		134	// Retrieve the tool type from the database
		135	var toolType = await _context.ToolType.FindAsync(toolTypeId);
107	r.RentPrice = toolType.Price * r.EndOfRent.Subtract(r.StartOfRent).TotalDays;	136	
		137	// Calculate the rent price based on the tool type's price and the rental duration
		138	r.RentPrice = toolType.Price * r.EndOfRent.Subtract(r.StartOfRent).TotalDays;
108	if (ModelState.IsValid)	139	
109	{	140	// Check if the model state is valid
		141	if (ModelState.IsValid)
		142	{
110	_context.Add(r);	143	// Add the rent to the context and save changes
111	await _context.SaveChangesAsync();	144	_context.Add(r);
		145	await _context.SaveChangesAsync();
112	return RedirectToAction("RentHistory");	146	
113	}	147	// Redirect to the RentHistory action if successful
		148	return RedirectToAction("RentHistory");
		149	}
114	- ViewData["DiscountId"] = new SelectList(_context.Discounts, "Id", "Id", r.DiscountId);		
115	- ViewData["ToolId"] = new SelectList(_context.Tools, "Id", "Id", r.ToolId);		
116	- ViewData["UserId"] = new SelectList(_context.Users, "Id", "Id", r.UserId);		
117	- return RedirectToAction("Create", new { toolTypeId });		
118	- }		
119	}		
120	}	150	
121	}	151	// Redirect to the Create action with the tool type ID if the model state is not valid
		152	return RedirectToAction("Create", new { toolTypeId });
		153	}
		154	}
		155	}

- Postavljena autorizacija na registrovanog korisnika za historiju iznajmljivanja

Historiju iznajmljivanja bi trebao moći vidjeti samo RegisteredUser, 😊
a ne korisnici sa ostalim rolama (Admin i Courier)



Burgija/Controllers/RentController.cs



30. [Authorize]

```
1  @@ -19,23 +19,30 @@ namespace Burgija.Controllers
19  public class RentController : Controller
20  {
21      private readonly ApplicationDbContext _context;
22      - private readonly UserManager<IdentityUser<int>> _userManager;
23
24      - public RentController(ApplicationDbContext context, UserManager<IdentityUser<int>> userManager)
25      {
26          _context = context;
27      -     _userManager = userManager;
28      }
29
30      - [Authorize]
31      public async Task<ActionResult> RentHistory()
32      {
33          var userId = Int32.Parse(User.FindFirst(ClaimTypes.NameIdentifier)?.Value);
34
35          var rentHistory = await _context.Rent
36      -     .Where(r => r.UserId == userId)
37      -     .Join(_context.Tool, rent => rent.ToolId, tool => tool.Id, (rent, tool) => new { Rent = rent, Tool = tool })
38      -     .Join(_context.ToolType, rt => rt.Tool.ToolTypeId, toolType => toolType.Id, (rt, toolType) => new
39      -         RentAndToolType(rt.Rent, toolType))
40      -     .ToListAsync();
41
42      -     return View(rentHistory);
43      }
44
45      + public RentController(ApplicationDbContext context)
46      + {
47      +     _context = context;
48      + }
49
50      + [Authorize(Roles = "RegisteredUser")]
51      + public async Task<ActionResult> RentHistory()
52      + {
53      +     // Extract the user ID from the claims associated with the current user
54      +     var userId = Int32.Parse(User.FindFirst(ClaimTypes.NameIdentifier)?.Value);
55      +
56      +     // Query the database to retrieve rent history information
57      +     var rentHistory = await _context.Rent
58      +     // Filter by the current user's ID
59      +     .Where(r => r.UserId == userId)
60      +     // Join the Rent table with the Tool table using the ToolId
61      +     .Join(_context.Tool, rent => rent.ToolId, tool => tool.Id, (rent, tool) => new { Rent = rent, Tool = tool })
62      +     // Join the result with the ToolType table using the ToolTypeId from the Tool table
63      +     .Join(_context.ToolType, rt => rt.Tool.ToolTypeId, toolType => toolType.Id, (rt, toolType) => new
64      +         RentAndToolType(rt.Rent, toolType))
65      +     // Convert the result to a list asynchronously
66      +     .ToListAsync();
67      +
68      +     // Return the rent history to the corresponding view
69      +     return View(rentHistory);
70      }
71
72      + }
```