```
 1  using DistLab2.Areas.Identity.Data;
 2  using DistLab2.Core;
 3  using DistLab2.Core.Interfaces;
 4  using DistLab2.Persistence;
 5  using DistLab2.ViewModels;
 6  using Microsoft.AspNetCore.Authorization;
 7  using Microsoft.AspNetCore.Identity;
 8  using Microsoft.AspNetCore.Mvc;
 9  using System.Diagnostics;
10
11  namespace DistLab2.Controllers
12  {
13      [Authorize]
14      //[Authorize(Roles = "Admin")] //endast admin har access till dessa
           resurser
15      public class AuctionController : Controller
16      {
17          private readonly IAuctionService AuctionService;
18          private readonly UserManager<DistUser> _userManager;
19
20          public AuctionController(IAuctionService
              service,UserManager<DistUser> userManager)
21          {
22              AuctionService = service;
23              _userManager = userManager;
24
25          }
26
27
28          //visar alla auctions
29          // GET: AuctionController
30          public ActionResult Index()
31          {
32              if (ModelState.IsValid)
33              {
34                  List<Auction> auctions = AuctionService.GetOngoing();
35                  List<AllAuctionsViewModel> auctionVm = new();
36                  foreach (Auction a in auctions)
37                  {
38                      // Check if the user related to the auction exists
39                      var user = _userManager.FindByNameAsync
                        (a.Username).Result;
40
41                      if (user != null)
42                      {
43                          double highestBid =
                        AuctionService.GetHighestBid(a.Id);
44                          auctionVm.Add(AllAuctionsViewModel.FromAuction
                        (a, highestBid));
45                      }
46                  }
47                  return View(auctionVm);
48              }
```

```
49              else return View();
50          }
51
52          //visar alla auctions som usern har själv lagt upp, samt vilka
              auktioner användaren har vunnit.
53          // GET: AuctionController
54          public ActionResult UserAuctions()
55          {
56              string currentUser = GetCurrentUser(); //name måste vara
                  unikt. Dubbelkolla att det är så.
57              if (currentUser != null)
58              {
59                  List<Auction> auctions =
                      AuctionService.GetAllByUsername(currentUser);
60                  List<AllAuctionsViewModel> auctionVm = new();
61
62                  List<Auction> wonAuctions =
                      AuctionService.GetWonAuctions(currentUser);
63
64                  List<AllAuctionsViewModel> wonAuctionsVm = new();
65
66                  foreach (Auction a in auctions)
67                  {
68                      double highestBid = AuctionService.GetHighestBid
                    (a.Id);
69                      auctionVm.Add(AllAuctionsViewModel.FromAuction(a,
                    highestBid));
70                  }
71
72                  foreach (Auction b in wonAuctions)
73                  {
74                      double highestBid = AuctionService.GetHighestBid
                    (b.Id);
75                      wonAuctionsVm.Add(AllAuctionsViewModel.FromAuction
                    (b, highestBid));
76                  }
77
78
79                  MyAuctionsViewModel viewModel = new MyAuctionsViewModel
                      ();
80                  viewModel.UserWinnedAuctions.AddRange(wonAuctionsVm);
81                  viewModel.UserOwnedAuctions.AddRange(auctionVm);
82
83
84                  return View(viewModel);
85              }
86              return View();
87          }
88
89          //Hämtar Auctions som en användare lagt bud på
90          public ActionResult UserBids()
91          {
92
```

```
93              string currentUser = GetCurrentUser();
94              if (currentUser == null) return RedirectToAction("Index");
95
96              List<Auction> userBiddedAuctions =
                  AuctionService.GetAuctionsWithUserBids(currentUser);
97              Debug.WriteLine(userBiddedAuctions.Count);
98
99
100             List<AllAuctionsViewModel> auctionViews = new();
101
102             foreach (Auction auction in userBiddedAuctions)
103             {
104                 double highestBid = AuctionService.GetHighestBid
                      (auction.Id);
105                 AllAuctionsViewModel auctionViewModel =
                      AllAuctionsViewModel.FromAuction(auction,
                      highestBid);
106                 auctionViews.Add(auctionViewModel);
107             }
108             return View(auctionViews);
109         }
110
111         // GET: AuctionController/Details/5
112         public ActionResult Details(int id, string name)
113         {
114             List<Bid> bids = AuctionService.GetBids(id);
115             Debug.WriteLine(bids.Count);
116             if (bids.Count > 0)
117             {
118                 BidDetailViewModel vm = BidDetailViewModel.FromBid
                      (bids, name);
119                 return View(vm);
120
121             }
122             TempData["NoBids"] = "No bids to display!";
123             return Redirect(Request.Headers["Referer"].ToString()); //
                  Returns to previous page if there are no bids
124         }
125
126         // GET: AuctionController/Create. Detta visas första gången
              användare går in på sidan. Då är formuläret tomt.
127         public ActionResult Create()
128         {
129             return View();
130         }
131
132         // POST: AuctionController/Create. Det som händer när
              användaren har fyllt i formuläret och tryckt på submit
133         [HttpPost]
134         [ValidateAntiForgeryToken]
135         public ActionResult Create(AuctionCreateViewModel vm)
136         {
137             if (ModelState.IsValid)
```

```
138             {
139                 Auction auction = new();
140                 auction.Username = GetCurrentUser();
141                 auction.Name = vm.Name;
142                 auction.Description = vm.Description;
143                 auction.StartingPrice = vm.StartingPrice;
144                 AuctionService.Add(auction);
145                 return RedirectToAction("Index");
146             } else return View();
147         }
148 
149         // GET: AuctionController/Edit/5
150         public ActionResult Edit(int id)
151         {
152             if(AuctionService.UserIsOwner(GetCurrentUser(), id))
153             {
154                 return View();
155             }
156             return RedirectToAction("Index");
157         }
158 
159         // POST: AuctionController/Edit/5
160         [HttpPost]
161         [ValidateAntiForgeryToken]
162         public ActionResult Edit(int id, IFormCollection collection,
              AuctionViewModel auction)
163         {
164             //if (!AuctionService.userIsOwnerOfAuction
                  (User.Identity.Name, id)) return RedirectToAction
                  ("Index");
165            //TODO: man borde validera här också. Den ovan funkar inte
                   av någon anledning
166             //måste man ha en IFormCollection
167 
168             try
169             {
170                 //TODO: kollar inte om ModelState är valid. Det funkade
                       inte för mig när jag testade med det.
171                 AuctionService.EditDescription(auction.Description,
                      id);
172                 return RedirectToAction("Index");
173                 //return RedirectToAction(nameof(Index));
174             }
175             catch
176             {
177                 return View();
178             }
179         }
180 
181         // GET: AuctionController/Delete/5
182         public ActionResult Delete(int id)
183         {
184             return View();
```

```
185         }
186
187         // POST: AuctionController/Delete/5
188         [HttpPost]
189         [ValidateAntiForgeryToken]
190         public ActionResult Delete(int id, IFormCollection collection)
191         {
192             try
193             {
194                 return RedirectToAction(nameof(Index));
195             }
196             catch
197             {
198                 return View();
199             }
200         }
201
202
203         [HttpPost]
204         [ValidateAntiForgeryToken]
205         public ActionResult AddBid(int id, string username, string
              auctionName, DateTime endDate)
206         {
207
208             string? currentUser = GetCurrentUser();
209
210             if (currentUser == null || currentUser == username)
211             {
212                 var referer = Request.Headers["Referer"].ToString(); //
                        If no user or current user auction
213                 TempData["OwnAuction"] = "You can not bid on your own
                      auction!";
214                 return Redirect(referer);
215             }
216
217             CreateBidViewModel vm = new CreateBidViewModel();
218             vm.CurrentHighestBid = AuctionService.GetHighestBid(id);
219             vm.AuctionId = id;
220             vm.AuctionName = auctionName;
221             vm.EndDate = endDate;
222
223
224             return View(vm);
225         }
226
227         [HttpPost]
228         [ValidateAntiForgeryToken]
229         public ActionResult CreateBid(CreateBidViewModel vm)
230         {
231             if (GetCurrentUser() != null)
232             {
233                 if (ModelState.IsValid && AuctionService.CheckIfOngoing
                      (vm.EndDate))
```

```
234                     {
235                         Bid bid = new();
236                         bid.BidAmount = vm.BidAmount;
237                         bid.Username = GetCurrentUser();
238                         bid.AuctionId = vm.AuctionId;
239                         AuctionService.AddBid(bid);
240                         return RedirectToAction("Index");
241                     }
242                     else
243                     {
244                         return View("AddBid", vm);
245                     }
246                 }
247 
248                 return RedirectToAction("Index");
249             }
250 
251 
252 
253             private string GetCurrentUser()
254             {
255                 return User.Identity.Name;
256             }
257 
258 
259 
260         }
261 
262 
263 }
264 
```