

LinksPlatform's Platform.Bot Class Library

1.1 ./csharp/Platform.Bot/Program.cs

```
1  using Interfaces;
2  using Octokit;
3  using Platform.Exceptions;
4  using Platform.IO;
5  using Storage.Local;
6  using Storage.Remote.GitHub;
7  using System;
8  using System.Collections.Generic;
9  using System.IO;
10 using System.Threading;
11 using Platform.Bot.Trackers;
12 using Platform.Bot.Triggers;
13
14 namespace Platform.Bot
15 {
16     /// <summary>
17     /// <para>
18     /// Represents the program.
19     /// </para>
20     /// <para></para>
21     /// </summary>
22     internal class Program
23     {
24         private static void Main(string[] args)
25         {
26             using var cancellation = new ConsoleCancellation();
27             var argumentIndex = 0;
28             var username = ConsoleHelpers.GetOrReadArgument(argumentIndex++, "Username", args);
29             var token = ConsoleHelpers.GetOrReadArgument(argumentIndex++, "Token", args);
30             var appName = ConsoleHelpers.GetOrReadArgument(argumentIndex++, "App Name", args);
31             var databaseFileName = ConsoleHelpers.GetOrReadArgument(argumentIndex++, "Database
32             ↪ file name", args);
33             var fileSetName = ConsoleHelpers.GetOrReadArgument(argumentIndex++, "File set name",
34             ↪ args);
35             var minimumInteractionIntervalStringInputInSeconds =
36             ↪ ConsoleHelpers.GetOrReadArgument(argumentIndex, "Minimum interaction interval in
37             ↪ seconds", args);
38             var minimumInteractionInterval = TimeSpan.FromSeconds(Int32.Parse(minimumInteraction
39             ↪ IntervalStringInputInSeconds));
40             var dbContext = new FileStorage(databaseFileName);
41             Console.WriteLine($"Bot has been started. {Environment.NewLine}Press CTRL+C to
42             ↪ close");
43             try
44             {
45                 while (true)
46                 {
47                     var api = new GitHubStorage(username, token, appName);
48                     var issueTracker = new IssueTracker(api,
49                     ↪ new HelloWorldTrigger(api, dbContext, fileSetName),
50                     ↪ new OrganizationLastMonthActivityTrigger(api),
51                     ↪ new LastCommitActivityTrigger(api),
52                     ↪ new ProtectMainBranchTrigger(api));
53                     var pullRequenstTracker = new PullRequestTracker(api, new
54                     ↪ MergeDependabotBumpsTrigger(api));
55                     var timestampTracker = new DateTimeTracker(api, new
56                     ↪ CreateAndSaveOrganizationRepositoriesMigrationTrigger(api, dbContext,
57                     ↪ Directory.GetCurrentDirectory()));
58                     issueTracker.Start(cancellation.Token);
59                     pullRequenstTracker.Start(cancellation.Token);
60                     timestampTracker.Start(cancellation.Token);
61                     Thread.Sleep(minimumInteractionInterval);
62                 }
63             }
64             catch (Exception ex)
65             {
66                 Console.WriteLine(ex.ToStringWithAllInnerExceptions());
67             }
68         }
69     }
70 }
```

1.2 ./csharp/Platform.Bot/Trackers/DateTimeTracker.cs

```
1  using System;
2  using System.Collections.Generic;
3  using System.Threading;
4  using Interfaces;
```

```

5 using Platform.Timestamps;
6 using Storage.Remote.GitHub;
7
8 namespace Platform.Bot.Trackers;
9
10 public class DateTimeTracker : ITracker<DateTime?>
11 {
12     private GitHubStorage _storage { get; }
13
14     private IList<ITrigger<DateTime?>> _triggers { get; }
15
16     public DateTimeTracker(GitHubStorage storage, params ITrigger<DateTime?>[] triggers)
17     {
18         _storage = storage;
19         _triggers = triggers;
20     }
21
22     public void Start(Cancellation_token cancellation_token)
23     {
24         foreach (var trigger in _triggers)
25         {
26             if (cancellation_token.IsCancellationRequested)
27             {
28                 return;
29             }
30             if (trigger.Condition(null))
31             {
32                 trigger.Action(null);
33             }
34         }
35     }
36 }

```

1.3 ./csharp/Platform.Bot/Trackers/IssueTracker.cs

```

1 using Interfaces;
2 using Octokit;
3 using Storage.Remote.GitHub;
4 using System.Collections.Generic;
5 using System.Threading;
6
7 namespace Platform.Bot.Trackers
8 {
9     /// <summary>
10     /// <para>
11     /// Represents the programmer role.
12     /// </para>
13     /// <para></para>
14     /// </summary>
15     public class IssueTracker : ITracker<Issue>
16     {
17         /// <summary>
18         /// <para>
19         /// The git hub api.
20         /// </para>
21         /// <para></para>
22         /// </summary>
23         private GitHubStorage _storage { get; }
24
25         /// <summary>
26         /// <para>
27         /// The triggers.
28         /// </para>
29         /// <para></para>
30         /// </summary>
31         private IList<ITrigger<Issue>> _triggers { get; }
32
33         /// <summary>
34         /// <para>
35         /// Initializes a new <see cref="IssueTracker"/> instance.
36         /// </para>
37         /// <para></para>
38         /// </summary>
39         /// <param name="triggers">
40         /// <para>A triggers.</para>
41         /// <para></para>
42         /// </param>
43         /// <param name="gitHubApi">
44         /// <para>A git hub api.</para>
45         /// <para></para>

```

```

46     /// </param>
47     public IssueTracker(GitHubStorage gitHubApi, params ITrigger<Issue>[] triggers)
48     {
49         _storage = gitHubApi;
50         _triggers = triggers;
51     }
52
53     /// <summary>
54     /// <para>
55     /// Starts the cancellation token.
56     /// </para>
57     /// <para></para>
58     /// </summary>
59     /// <param name="cancellationToken">
60     /// <para>The cancellation token.</para>
61     /// <para></para>
62     /// </param>
63     public void Start(CancellationToken cancellationToken)
64     {
65         foreach (var trigger in _triggers)
66         {
67             foreach (var issue in _storage.GetIssues())
68             {
69                 if (cancellationToken.IsCancellationRequested)
70                 {
71                     return;
72                 }
73                 if (trigger.Condition(issue))
74                 {
75                     trigger.Action(issue);
76                 }
77             }
78         }
79     }
80 }
81 }

```

1.4 ./csharp/Platform.Bot/Trackers/PullRequestTracker.cs

```

1  using Interfaces;
2  using Octokit;
3  using Storage.Remote.GitHub;
4  using System.Collections.Generic;
5  using System.Threading;
6  using Platform.Collections.Lists;
7  using Platform.Threading;
8
9  namespace Platform.Bot.Trackers
10 {
11     /// <summary>
12     /// <para>
13     /// Represents the programmer role.
14     /// </para>
15     /// <para></para>
16     /// </summary>
17     public class PullRequestTracker : ITracker<PullRequest>
18     {
19         /// <summary>
20         /// <para>
21         /// The git hub api.
22         /// </para>
23         /// <para></para>
24         /// </summary>
25         private GitHubStorage _storage;
26
27         /// <summary>
28         /// <para>
29         /// The triggers.
30         /// </para>
31         /// <para></para>
32         /// </summary>
33         private IList<ITrigger<PullRequest>> _triggers;
34
35         /// <summary>
36         /// <para>
37         /// Initializes a new <see cref="IssueTracker"/> instance.
38         /// </para>
39         /// <para></para>
40         /// </summary>
41         /// <param name="triggers">

```

```

42     /// <para>A triggers.</para>
43     /// <para></para>
44     /// </param>
45     /// <param name="storage">
46     /// <para>A git hub api.</para>
47     /// <para></para>
48     /// </param>
49     public PullRequestTracker(GitHubStorage storage, params ITrigger<PullRequest>[] triggers)
50     {
51         _storage = storage;
52         _triggers = triggers;
53     }
54
55     /// <summary>
56     /// <para>
57     /// Starts the cancellation token.
58     /// </para>
59     /// <para></para>
60     /// </summary>
61     /// <param name="cancellationToken">
62     /// <para>The cancellation token.</para>
63     /// <para></para>
64     /// </param>
65     public void Start(Cancellation_token cancellationToken)
66     {
67         foreach (var trigger in _triggers)
68         {
69             foreach (var repository in
70                 ↪ _storage.Client.Repository.GetAllForOrg("linksplatform").AwaitResult())
71             {
72                 foreach (var pullRequest in _storage.Client.PullRequest.GetAllForRepository(
73                     ↪ repository.Id).AwaitResult())
74                 {
75                     if (cancellationToken.IsCancellationRequested)
76                     {
77                         return;
78                     }
79                     var detailedPullRequest = _storage.Client.PullRequest.Get(repository.Id,
80                         ↪ pullRequest.Number).AwaitResult();
81                     if (trigger.Condition(detailedPullRequest))
82                     {
83                         trigger.Action(detailedPullRequest);
84                     }
85                 }
86             }
87         }
88     }

```

1.5 ./csharp/Platform.Bot/Triggers/Activity.cs

```

1 using System.Collections.Generic;
2
3 namespace Platform.Bot.Triggers
4 {
5     internal class Activity
6     {
7         public string Url { get; set; }
8
9         public List<string> Repositories { get; set; }
10    }
11 }

```

1.6 ./csharp/Platform.Bot/Triggers/CreateAndSaveOrganizationRepositoriesMigrationTrigger.cs

```

1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6 using Interfaces;
7 using Octokit;
8 using Platform.Data;
9 using Platform.Data.Doublets;
10 using Platform.Timestamps;
11 using Storage.Local;
12 using Storage.Remote.GitHub;
13
14 namespace Platform.Bot.Triggers;
15

```

```

16 public class CreateAndSaveOrganizationRepositoriesMigrationTrigger : ITrigger<DateTime?>
17 {
18     private readonly GitHubStorage _githubStorage;
19
20     private readonly FileStorage _linksStorage;
21
22     private string _filePath;
23
24     public CreateAndSaveOrganizationRepositoriesMigrationTrigger(GitHubStorage githubStorage,
25         ↪ FileStorage linksStorage, string filePath)
26     {
27         _githubStorage = githubStorage;
28         _linksStorage = linksStorage;
29         _filePath = filePath;
30     }
31     public bool Condition(DateTime? dateTime)
32     {
33         var allMigrations = _githubStorage.GetAllMigrations("linksplatform");
34         if (allMigrations.Count == 0)
35         {
36             return true;
37         }
38         var lastMigrationTimestamp = Convert.ToDateTime(allMigrations.Last().CreatedAt);
39         var timeAfterLastMigration = DateTime.Now - lastMigrationTimestamp;
40         return timeAfterLastMigration.Days > 1;
41     }
42
43     public async void Action(DateTime? dateTime)
44     {
45         var repositoryNames = _githubStorage.GetAllRepositoryNames("linksplatform");
46         var createMigrationResult = await _githubStorage.CreateMigration("linksplatform",
47             ↪ repositoryNames);
48         if (null == createMigrationResult || createMigrationResult.State.Value ==
49             ↪ Migration.MigrationState.Failed)
50         {
51             Console.WriteLine("Migration is failed.");
52             return;
53         }
54         Console.WriteLine($"Saving migration {createMigrationResult.Id}.");
55         await _githubStorage.SaveMigrationArchive("linksplatform", createMigrationResult.Id,
56             ↪ _filePath);
57         Console.WriteLine($"Migration {createMigrationResult.Id} is saved.");
58     }
59 }

```

1.7 ./csharp/Platform.Bot/Triggers/HelloWorldTrigger.cs

```

1 using Interfaces;
2 using Octokit;
3 using Storage.Local;
4 using Storage.Remote.GitHub;
5
6 namespace Platform.Bot.Triggers
7 {
8     using TContext = Issue;
9     /// <summary>
10    /// <para>
11    /// Represents the hello world trigger.
12    /// </para>
13    /// <para></para>
14    /// </summary>
15    /// <seealso cref="ITrigger{TContext}"/>
16    internal class HelloWorldTrigger : ITrigger<TContext>
17    {
18        private readonly GitHubStorage _storage;
19        private readonly FileStorage _fileStorage;
20        private readonly string _fileSetName;
21
22        /// <summary>
23        /// <para>
24        /// Initializes a new <see cref="HelloWorldTrigger"/> instance.
25        /// </para>
26        /// <para></para>
27        /// </summary>
28        /// <param name="storage">
29        /// <para>A git hub api.</para>
30        /// <para></para>
31        /// </param>
32        /// <param name="fileStorage">

```

```

33     /// <para>A file storage.</para>
34     /// <para></para>
35     /// </param>
36     /// <param name="fileSetName">
37     /// <para>A file set name.</para>
38     /// <para></para>
39     /// </param>
40     public HelloWorldTrigger(GitHubStorage storage, FileStorage fileStorage, string
        ↪ fileSetName)
41     {
42         this._storage = storage;
43         this._fileStorage = fileStorage;
44         this._fileSetName = fileSetName;
45     }
46
47
48     /// <summary>
49     /// <para>
50     /// Actions the context.
51     /// </para>
52     /// <para></para>
53     /// </summary>
54     /// <param name="context">
55     /// <para>The context.</para>
56     /// <para></para>
57     /// </param>
58
59     public void Action(TContext context)
60     {
61         foreach (var file in _fileStorage.GetFilesFromSet(_fileSetName))
62         {
63             _storage.CreateOrUpdateFile(context.Repository.Name,
        ↪ context.Repository.DefaultBranch, file);
64         }
65         _storage.CloseIssue(context);
66     }
67
68
69     /// <summary>
70     /// <para>
71     /// Determines whether this instance condition.
72     /// </para>
73     /// <para></para>
74     /// </summary>
75     /// <param name="context">
76     /// <para>The context.</para>
77     /// <para></para>
78     /// </param>
79     /// <returns>
80     /// <para>The bool</para>
81     /// <para></para>
82     /// </returns>
83     public bool Condition(TContext context) => context.Title.ToLower() == "hello world";
84 }
85 }

```

1.8 ./csharp/Platform.Bot/Triggers/LastCommitActivityTrigger.cs

```

1  using Interfaces;
2  using Octokit;
3  using Platform.Communication.Protocol.Lino;
4  using Storage.Remote.GitHub;
5  using System;
6  using System.Collections.Generic;
7  using System.Linq;
8  using System.Text;
9
10 namespace Platform.Bot.Triggers
11 {
12     using TContext = Issue;
13     internal class LastCommitActivityTrigger : ITrigger<TContext>
14     {
15         private readonly GitHubStorage _storage;
16
17         private readonly Parser _parser = new();
18
19         public LastCommitActivityTrigger(GitHubStorage storage) => _storage = storage;
20
21         public bool Condition(TContext context) => context.Title.ToLower() == "last 3 months
        ↪ commit activity";
22     }

```

```

23 public void Action(TContext context)
24 {
25     var issueService = _storage.Client.Issue;
26     var owner = context.Repository.Owner.Login;
27     var ignoredRepositories =
28         context.Body != null ? GetIgnoredRepositories(_parser.Parse(context.Body)) :
29         ↪ default;
30     var users = GetActivities(owner, ignoredRepositories);
31     StringBuilder sb = new();
32     foreach (var user in users)
33     {
34         sb.AppendLine($"- **{user.Url.Replace("api.", "").Replace("users/", "")}**");
35         // Break line
36         sb.AppendLine("-----");
37     }
38     var result = sb.ToString();
39     var comment = issueService.Comment.Create(owner, context.Repository.Name,
40     ↪ context.Number, result);
41     comment.Wait();
42     Console.WriteLine($"Last commit activity comment is added:
43     ↪ {comment.Result.HtmlUrl}");
44     _storage.CloseIssue(context);
45 }
46
47 public HashSet<string> GetIgnoredRepositories(IList<Link> links)
48 {
49     HashSet<string> ignoredRepos = new() { };
50     foreach (var link in links)
51     {
52         var values = link.Values;
53         if (values != null && values.Count == 3 && string.Equals(values.First().Id,
54         ↪ "ignore", StringComparison.OrdinalIgnoreCase) &&
55         ↪ string.Equals(values.Last().Id.Trim('.'), "repository",
56         ↪ StringComparison.OrdinalIgnoreCase))
57         {
58             ignoredRepos.Add(values[1].Id);
59         }
60     }
61     return ignoredRepos;
62 }
63
64 public HashSet<Activity> GetActivities(string owner, HashSet<string> ignoredRepositories
65 ↪ = default)
66 {
67     HashSet<Activity> activeUsers = new();
68     foreach (var repository in _storage.Client.Repository.GetAllForOrg(owner).Result)
69     {
70         if (ignoredRepositories?.Contains(repository.Name) ?? false)
71         {
72             continue;
73         }
74         foreach (var commit in _storage.GetCommits(repository.Owner.Login,
75         ↪ repository.Name, DateTime.Today.AddMonths(-3)))
76         {
77             if (!_storage.Client.Organization.Member.CheckMember(owner,
78             ↪ commit.Author.Login).Result)
79             {
80                 continue;
81             }
82             if (!activeUsers.Any(x => x.Url == commit.Author.Url))
83             {
84                 activeUsers.Add(new Activity() { Url = commit.Author.Url, Repositories =
85                 ↪ new List<string> { repository.Url } });
86             }
87             else
88             {
89                 if (!activeUsers.Any(x => x.Repositories.Any(y => y == repository.Url)
90                 ↪ == true))
91                 {
92                     activeUsers.FirstOrDefault(x => x.Url ==
93                     ↪ commit.Author.Url).Repositories.Add(repository.Url);
94                 }
95             }
96         }
97     }
98     return activeUsers;
99 }

```

```

88     }
89 }

```

1.9 ./csharp/Platform.Bot/Triggers/MergeDependabotBumpsTrigger.cs

```

1  using System;
2  using Interfaces;
3  using Octokit;
4  using Platform.Threading;
5  using Storage.Remote.GitHub;
6
7  namespace Platform.Bot.Triggers
8  {
9      public class MergeDependabotBumpsTrigger : ITrigger<PullRequest>
10     {
11         private const int DependabotId = 49699333;
12         private readonly GitHubStorage _githubStorage;
13         public MergeDependabotBumpsTrigger(GitHubStorage storage)
14         {
15             _githubStorage = storage;
16         }
17
18         public bool Condition(PullRequest pullRequest)
19         {
20             var isDependabotAuthor = DependabotId == pullRequest.User.Id;
21             if (!isDependabotAuthor)
22             {
23                 return false;
24             }
25             var isTestAndDeployCompleted = false;
26             var repositoryId = pullRequest.Base.Repository.Id;
27             var checks = _githubStorage.Client.Check.Run.GetAllForReference(repositoryId,
28                 ↪ pullRequest.Head.Sha).AwaitResult();
29             foreach (var checkRun in checks.CheckRuns)
30             {
31                 if (checkRun.Name is "testAndDeploy" or "deploy" && checkRun.Status.Value ==
32                     ↪ CheckStatus.Completed)
33                 {
34                     isTestAndDeployCompleted = true;
35                 }
36             }
37             var isMergable = pullRequest.Mergeable ?? false;
38             return isTestAndDeployCompleted && isMergable;
39         }
40
41         public void Action(PullRequest pullRequest)
42         {
43             var repositoryId = pullRequest.Base.Repository.Id;
44             var prMerge = _githubStorage.Client.PullRequest.Merge(repositoryId,
45                 ↪ pullRequest.Number, new MergePullRequest()).AwaitResult();
46             Console.WriteLine($"{{pullRequest.HtmlUrl}} is {{(prMerge.Merged ? "successfully":"not
47                 ↪ successfully"}} merged.");
48         }
49     }
50 }

```

1.10 ./csharp/Platform.Bot/Triggers/OrganizationLastMonthActivityTrigger.cs

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using Interfaces;
5  using Octokit;
6  using Platform.Communication.Protocol.Lino;
7  using Storage.Remote.GitHub;
8
9  namespace Platform.Bot.Triggers
10 {
11     using TContext = Issue;
12     /// <summary>
13     /// <para>
14     /// Represents the organization last month activity trigger.
15     /// </para>
16     /// <para></para>
17     /// </summary>
18     /// <seealso cref="ITrigger{Issue}"/>
19     internal class OrganizationLastMonthActivityTrigger : ITrigger<TContext>
20     {
21         private readonly GitHubStorage _storage;
22         private readonly Parser _parser = new();
23
24         /// <summary>

```



```

25     /// <para>
26     /// Initializes a new <see cref="OrganizationLastMonthActivityTrigger"/> instance.
27     /// </para>
28     /// <para></para>
29     /// </summary>
30     /// <param name="storage">
31     /// <para>A storage.</para>
32     /// <para></para>
33     /// </param>
34     public OrganizationLastMonthActivityTrigger(GitHubStorage storage) => _storage = storage;
35
36     /// <summary>
37     /// <para>
38     /// Determines whether this instance condition.
39     /// </para>
40     /// <para></para>
41     /// </summary>
42     /// <param name="context">
43     /// <para>The context.</para>
44     /// <para></para>
45     /// </param>
46     /// <returns>
47     /// <para>The bool</para>
48     /// <para></para>
49     /// </returns>
50     public bool Condition(TContext context) => context.Title.ToLower() == "organization last
    ↪ month activity";
51
52     /// <summary>
53     /// <para>
54     /// Actions the context.
55     /// </para>
56     /// <para></para>
57     /// </summary>
58     /// <param name="context">
59     /// <para>The context.</para>
60     /// <para></para>
61     /// </param>
62     public void Action(TContext context)
63     {
64         var issueService = _storage.Client.Issue;
65         var owner = context.Repository.Owner.Login;
66         var activeUsersString = string.Join("\n",
        ↪ GetActiveUsers(GetIgnoredRepositories(_parser.Parse(context.Body)), owner));
67         issueService.Comment.Create(owner, context.Repository.Name, context.Number,
        ↪ activeUsersString);
68         _storage.CloseIssue(context);
69     }
70
71     /// <summary>
72     /// <para>
73     /// Gets the ignored repositories using the specified links.
74     /// </para>
75     /// <para></para>
76     /// </summary>
77     /// <param name="links">
78     /// <para>The links.</para>
79     /// <para></para>
80     /// </param>
81     /// <returns>
82     /// <para>The ignored repos.</para>
83     /// <para></para>
84     /// </returns>
85     public HashSet<string> GetIgnoredRepositories(ICollection<Link> links)
86     {
87         HashSet<string> ignoredRepos = new() { };
88         foreach (var link in links)
89         {
90             var values = link.Values;
91             if (values != null && values.Count == 3 && string.Equals(values.First().Id,
        ↪ "ignore", StringComparison.OrdinalIgnoreCase) &&
        ↪ string.Equals(values.Last().Id.Trim('.'), "repository",
        ↪ StringComparison.OrdinalIgnoreCase))
        {
92                 ignoredRepos.Add(values[1].Id);
93             }
94         }
95     }
96     return ignoredRepos;

```

```

97     }
98
99     /// <summary>
100    /// <para>
101    /// Gets the active users using the specified ignored repositories.
102    /// </para>
103    /// <para></para>
104    /// </summary>
105    /// <param name="ignoredRepositories">
106    /// <para>The ignored repositories.</para>
107    /// <para></para>
108    /// </param>
109    /// <param name="owner">
110    /// <para>The owner.</para>
111    /// <para></para>
112    /// </param>
113    /// <returns>
114    /// <para>The active users.</para>
115    /// <para></para>
116    /// </returns>
117    public HashSet<string> GetActiveUsers(HashSet<string> ignoredRepositories, string owner)
118    {
119        HashSet<string> activeUsers = new();
120        var date = DateTime.Now.AddMonths(-1);
121        foreach (var repository in _storage.Client.Repository.GetAllForOrg(owner).Result)
122        {
123            if (ignoredRepositories.Contains(repository.Name))
124            {
125                continue;
126            }
127            foreach (var commit in _storage.GetCommits(repository.Owner.Login,
128                ↪ repository.Name))
129            {
130                activeUsers.Add(commit.Author.Login);
131            }
132            foreach (var pullRequest in _storage.GetPullRequests(repository.Owner.Login,
133                ↪ repository.Name))
134            {
135                foreach (var reviewer in pullRequest.RequestedReviewers)
136                {
137                    if (pullRequest.CreatedAt < date || pullRequest.UpdatedAt < date ||
138                        ↪ pullRequest.ClosedAt < date || pullRequest.MergedAt < date)
139                    {
140                        activeUsers.Add(reviewer.Login);
141                    }
142                }
143                foreach (var createdIssue in _storage.GetIssues(repository.Owner.Login,
144                    ↪ repository.Name))
145                {
146                    activeUsers.Add(createdIssue.User.Login);
147                }
148            }
149            return activeUsers;
150        }
151    }

```

1.11 ./csharp/Platform.Bot/Triggers/ProtectMainBranchTrigger.cs

```

1  using System.Collections.Generic;
2  using System.Linq;
3  using System.Text;
4  using Interfaces;
5  using Octokit;
6  using Storage.Remote.GitHub;
7
8  namespace Platform.Bot.Triggers
9  {
10     using TContext = Issue;
11     public class ProtectMainBranchTrigger : ITrigger<TContext>
12     {
13         private readonly GitHubStorage _storage;
14         public ProtectMainBranchTrigger(GitHubStorage storage) => _storage = storage;
15         public bool Condition(TContext context) => context.Title.ToLower() == "protect default
16             ↪ branch in all organization's repositories";
17
18         public void Action(TContext context)

```

```

18 {
19     var repositories =
20         ↪ _storage.Client.Repository.GetAllForOrg(context.Repository.Owner.Login).Result;
21     var results = UpdateRepositoriesDefaultBranchProtection(repositories);
22     StringBuilder failedRepositoriesComment = new(repositories.Count *
23         ↪ repositories[0].Name.Length);
24     foreach (var result in results.Where(result => !result.Value))
25     {
26         failedRepositoriesComment.AppendLine($"{result.Key}");
27     }
28     if (failedRepositoriesComment.Length != 0)
29     {
30         failedRepositoriesComment.AppendLine(
31             ↪ "TODO: Fix default branch protection of these repositories. Failed
32             ↪ repositories:");
33         _storage.Client.Issue.Comment.Create(context.Repository.Id, context.Number,
34             ↪ failedRepositoriesComment.ToString());
35     }
36     else
37     {
38         _storage.Client.Issue.Comment.Create(context.Repository.Id, context.Number,
39             ↪ "Success. All repositories default branch protection is updated.");
40         _storage.CloseIssue(context);
41     }
42 }
43
44 public Dictionary<string, bool>
45 ↪ UpdateRepositoriesDefaultBranchProtection(IReadOnlyList<Repository> repositories)
46 {
47     Dictionary<string, bool> result = new(repositories.Count);
48     foreach (var repository in repositories)
49     {
50         if (repository.Private)
51         {
52             continue;
53         }
54         var update = new BranchProtectionSettingsUpdate(new
55             ↪ BranchProtectionPushRestrictionsUpdate());
56         var request =
57             ↪ _storage.Client.Repository.Branch.UpdateBranchProtection(repository.Id,
58                 ↪ repository.DefaultBranch, update);
59         request.Wait();
60         result.Add(repository.Name, request.IsCompletedSuccessfully);
61     }
62     return result;
63 }
64
65 }
66
67 }
68
69 }
```

Index

[./csharp/Platform.Bot/Program.cs](#), 1
[./csharp/Platform.Bot/Trackers/DateTimeTracker.cs](#), 1
[./csharp/Platform.Bot/Trackers/IssueTracker.cs](#), 2
[./csharp/Platform.Bot/Trackers/PullRequestTracker.cs](#), 3
[./csharp/Platform.Bot/Triggers/Activity.cs](#), 4
[./csharp/Platform.Bot/Triggers/CreateAndSaveOrganizationRepositoriesMigrationTrigger.cs](#), 4
[./csharp/Platform.Bot/Triggers/HelloWorldTrigger.cs](#), 5
[./csharp/Platform.Bot/Triggers/LastCommitActivityTrigger.cs](#), 6
[./csharp/Platform.Bot/Triggers/MergeDependabotBumpsTrigger.cs](#), 8
[./csharp/Platform.Bot/Triggers/OrganizationLastMonthActivityTrigger.cs](#), 8
[./csharp/Platform.Bot/Triggers/ProtectMainBranchTrigger.cs](#), 10