

## 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                     ↪ CreateAndSaveOrganizationRepositoriesMigrationTrigger(api, dbContext,
55                     ↪ Path.Combine(Directory.GetCurrentDirectory(), "/github-migrations")));
56                     issueTracker.Start(cancellation.Token);
57                     pullRequenstTracker.Start(cancellation.Token);
58                     timestampTracker.Start(cancellation.Token);
59                     Thread.Sleep(minimumInteractionInterval);
60                 }
61             }
62             catch (Exception ex)
63             {
64                 Console.WriteLine(ex.ToStringWithAllInnerExceptions());
65             }
66         }
67     }
68 }
```

### 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         var allIssues = _storage.GetIssues();
66         foreach (var issue in allIssues)
67         {
68             foreach (var trigger in _triggers)
69             {
70                 if (cancellationToken.IsCancellationRequested)
71                 {
72                     return;
73                 }
74                 if (trigger.Condition(issue))
75                 {
76                     trigger.Action(issue);
77                 }
78             }
79         }
80     }
81 }
82 }

```

#### 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(CancellationToken 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.IO;
4 using System.Linq;
5 using System.Text;
6 using System.Threading.Tasks;
7 using Interfaces;
8 using Octokit;
9 using Platform.Data;
10 using Platform.Data.Doublets;
11 using Platform.Timestamps;
12 using Storage.Local;
13 using Storage.Remote.GitHub;
14

```

```

15 namespace Platform.Bot.Triggers;
16
17 public class CreateAndSaveOrganizationRepositoriesMigrationTrigger : ITrigger<DateTime?>
18 {
19     private readonly GitHubStorage _githubStorage;
20
21     private readonly FileStorage _linksStorage;
22
23     private string _directoryPath;
24
25     public CreateAndSaveOrganizationRepositoriesMigrationTrigger(GitHubStorage githubStorage,
26         ↪ FileStorage linksStorage, string directoryPath)
27     {
28         _githubStorage = githubStorage;
29         _linksStorage = linksStorage;
30         _directoryPath = directoryPath;
31     }
32
33     public bool Condition(DateTime? dateTime)
34     {
35         var allMigrations = _githubStorage.GetAllMigrations("linksplatform");
36         if (allMigrations.Count == 0)
37         {
38             return true;
39         }
40         var lastMigrationTimestamp = Convert.ToDateTime(allMigrations.Last().CreatedAt);
41         var timeAfterLastMigration = DateTime.Now - lastMigrationTimestamp;
42         return timeAfterLastMigration.Days > 1;
43     }
44
45     public async void Action(DateTime? dateTime)
46     {
47         var repositoryNames =
48             ↪ _githubStorage.GetAllRepositories("linksplatform").Result.Select(repository =>
49             ↪ repository.Name).ToList();
50         var createMigrationResult = await _githubStorage.CreateMigration("linksplatform",
51             ↪ repositoryNames);
52         if (null == createMigrationResult || createMigrationResult.State.Value ==
53             ↪ Migration.MigrationState.Failed)
54         {
55             Console.WriteLine("Migration is failed.");
56             return;
57         }
58         Console.WriteLine($"Saving migration {createMigrationResult.Id}.");
59         var fileName = Path.Combine(_directoryPath, $"migration_{createMigrationResult.Id}");
60         await _githubStorage.SaveMigrationArchive("linksplatform", createMigrationResult.Id,
61             ↪ fileName);
62         Console.WriteLine($"Migration {createMigrationResult.Id} is saved.");
63     }
64 }

```

## 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.Concurrent;
7  using System.Collections.Generic;
8  using System.Linq;
9  using System.Text;
10 using System.Threading.Tasks;
11
12 namespace Platform.Bot.Triggers
13 {
14     using TContext = Issue;
15     internal class LastCommitActivityTrigger : ITrigger<TContext>
16     {
17         private readonly GitHubStorage _githubStorage;
18
19         public LastCommitActivityTrigger(GitHubStorage storage) => _githubStorage = storage;

```

```

20
21 public bool Condition(TContext issue) => "last 3 months commit activity" ==
    ↳ issue.Title.ToLower();
22
23 public void Action(TContext issue)
24 {
25     var organizationName = issue.Repository.Owner.Login;
26     var allMembers = _githubStorage.GetOrganizationMembers(organizationName);
27     var usersAndRepositoriesTheyCommitted = new ConcurrentDictionary<User,
    ↳ HashSet<Repository>>();
28     allMembers.All(user =>
29     {
30         usersAndRepositoriesTheyCommitted.TryAdd(user, new HashSet<Repository>());
31         return true;
32     });
33     var allRepositories = _githubStorage.GetAllRepositories(organizationName).Result;
34     var allTasks = new Queue<Task>();
35     foreach (var repository in allRepositories)
36     {
37         var repositoryCommitsTask = _githubStorage.GetCommits(repository.Id, new
    ↳ CommitRequest{Since = DateTime.Now.AddMonths(-3)});
38         repositoryCommitsTask.ContinueWith(task =>
39         {
40             task.Result.All(commit =>
41             {
42                 allMembers.All(user =>
43                 {
44                     if (commit.Author?.Id == user.Id)
45                     {
46                         usersAndRepositoriesTheyCommitted[user].Add(repository);
47                     }
48                     return true;
49                 });
50                 return true;
51             });
52             });
53             allTasks.Enqueue(repositoryCommitsTask);
54         }
55         Task.WaitAll(allTasks.ToArray());
56         var activeUsersAndRepositoriesTheyCommitted =
    ↳ usersAndRepositoriesTheyCommitted.Where(userAndRepositoriesCommitted =>
    ↳ userAndRepositoriesCommitted.Value.Count > 0).ToDictionary(pair => pair.Key, pair
    ↳ => pair.Value);
57         StringBuilder messageSb = new();
58         AddTldrMessageToSb(activeUsersAndRepositoriesTheyCommitted, messageSb);
59         AddUsersAndRepositoriesTheyCommittedToSb(activeUsersAndRepositoriesTheyCommitted,
    ↳ messageSb);
60         var message = messageSb.ToString();
61         var comment = _githubStorage.CreateIssueComment(issue.Repository.Id, issue.Number,
    ↳ message);
62         comment.Wait();
63         Console.WriteLine($"Issue {issue.Title} is processed: {issue.Url}");
64         _githubStorage.CloseIssue(issue);
65     }
66
67 private void AddTldrMessageToSb(IDictionary<User, HashSet<Repository>>
    ↳ usersAndRepositoriesCommitted, StringBuilder sb)
68 {
69     sb.AppendLine("## TLDR:");
70     usersAndRepositoriesCommitted.Keys.All(user =>
71     {
72         sb.AppendLine($"[{user.Login}] ({user.Url})");
73         return true;
74     });
75     sb.AppendLine();
76     sb.AppendLine("---");
77     sb.AppendLine();
78 }
79
80 private void AddUsersAndRepositoriesTheyCommittedToSb(IDictionary<User,
    ↳ HashSet<Repository>> usersAndRepositoriesCommitted, StringBuilder sb)
81 {
82     foreach (var userAndCommittedRepositories in usersAndRepositoriesCommitted)
83     {
84         var user = userAndCommittedRepositories.Key;
85         var repositoriesUserCommitted = userAndCommittedRepositories.Value;
86         sb.AppendLine($"**[{user.Login}] ({user.Url})**");

```

```

87         repositoriesUserCommitted.All(repository => {
88             sb.AppendLine($"{repository.Name} ({repository.Url})");
89             return true;
90         });
91         sb.AppendLine("---");
92     }
93 }
94 }
95 }

```

## 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 readonly GitHubStorage _githubStorage;
12         public MergeDependabotBumpsTrigger(GitHubStorage storage)
13         {
14             _githubStorage = storage;
15         }
16
17         public bool Condition(PullRequest pullRequest)
18         {
19             var isDependabotAuthor = GitHubStorage.DependabotId == pullRequest.User.Id;
20             if (!isDependabotAuthor)
21             {
22                 return false;
23             }
24             var isDeployCheckCompleted = false;
25             var hasDeployCheck = 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")
32                 {
33                     hasDeployCheck = true;
34                     if (CheckStatus.Completed == checkRun.Status.Value)
35                     {
36                         isDeployCheckCompleted = true;
37                     }
38                 }
39             }
40             var isMergable = pullRequest.Mergeable ?? false;
41             return (isDeployCheckCompleted || !hasDeployCheck) && isMergable;
42         }
43
44         public void Action(PullRequest pullRequest)
45         {
46             var repositoryId = pullRequest.Base.Repository.Id;
47             var prMerge = _githubStorage.Client.PullRequest.Merge(repositoryId,
48                 ↪ pullRequest.Number, new MergePullRequest()).AwaitResult();
49             Console.WriteLine($"{pullRequest.HtmlUrl} is {(prMerge.Merged ? "successfully":"not
50                 ↪ successfully")} merged.");
51         }
52     }
53 }

```

## 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     {
93         ignoredRepos.Add(values[1].Id);
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         var commits = _storage.GetCommits(repository.Id, new CommitRequest { Since =
    ↪     date }).Result;
128         foreach (var commit in commits)
129         {
130
131             activeUsers.Add(commit.Author.Login);
132         }
133         foreach (var pullRequest in _storage.GetPullRequests(repository.Owner.Login,
    ↪     repository.Name))
134         {
135             foreach (var reviewer in pullRequest.RequestedReviewers)
136             {
137                 if (pullRequest.CreatedAt < date || pullRequest.UpdatedAt < date ||
    ↪     pullRequest.ClosedAt < date || pullRequest.MergedAt < date)
138                 {
139                     activeUsers.Add(reviewer.Login);
140                 }
141             }
142         }
143         foreach (var createdIssue in _storage.GetIssues(repository.Owner.Login,
    ↪     repository.Name))
144         {
145             activeUsers.Add(createdIssue.User.Login);
146         }
147     }
148     return activeUsers;
149 }
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
            ↳ branch in all organization's repositories";
16
17         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 }

```

## 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