Documentation v0.1

City Crowd

Back-end

Farhad Eftekhari

2015

Contents

[Tables 10](#_Toc424277919)

[Admins 11](#_Toc424277920)

[Blog 13](#_Toc424277921)

[Bookmarks 15](#_Toc424277922)

[Events 16](#_Toc424277923)

[EventsBoardsMessages 20](#_Toc424277924)

[EventsInvites 21](#_Toc424277925)

[EventsRecommendations 22](#_Toc424277926)

[EventsRequests 23](#_Toc424277927)

[EventsRequestsAccepted 25](#_Toc424277928)

[EventsSeen 26](#_Toc424277929)

[Feeds 27](#_Toc424277930)

[Feedbacks 28](#_Toc424277931)

[Followers 29](#_Toc424277932)

[ForgotPassword 31](#_Toc424277933)

[Hashtags 33](#_Toc424277934)

[LocationRequests 34](#_Toc424277935)

[Locations 35](#_Toc424277936)

[LoginSessions 37](#_Toc424277937)

[MessagesLists 38](#_Toc424277938)

[Messages 40](#_Toc424277939)

[Notifications 42](#_Toc424277940)

[Reports 44](#_Toc424277941)

[ReviewRequests 46](#_Toc424277942)

[Reviews 47](#_Toc424277943)

[Settings 49](#_Toc424277944)

[Statistics 50](#_Toc424277945)

[Users 52](#_Toc424277946)

[VerificationEmail 60](#_Toc424277947)

[Indexes 61](#_Toc424277948)

[ix\_Admins\_UserId 62](#_Toc424277949)

[ix\_Bookmarks\_UserId 63](#_Toc424277950)

[ix\_Events\_OwnerId 64](#_Toc424277951)

[ix\_EventsBoardsMessages\_EventId 65](#_Toc424277952)

[ix\_EventsInvites\_ReceiverId 66](#_Toc424277953)

[ix\_EventsRecommendations\_UserId 67](#_Toc424277954)

[ix\_EventsRequests\_OwnerId 68](#_Toc424277955)

[ix\_EventsRequestsAccepted\_EventId 69](#_Toc424277956)

[ix\_Feed\_OwnerId 70](#_Toc424277957)

[ix\_Followers\_FollowerId 71](#_Toc424277958)

[ix\_Followers\_UserId 72](#_Toc424277959)

[ix\_Messages\_OwnerId 73](#_Toc424277960)

[ix\_MessageLists\_OwnerId 74](#_Toc424277961)

[ix\_Notifications\_UserId 75](#_Toc424277962)

[ix\_ReviewRequests\_OwnerId 76](#_Toc424277963)

[ix\_Reviews\_UserId 77](#_Toc424277964)

[ix\_Users\_Username 78](#_Toc424277965)

[Triggers 79](#_Toc424277966)

[tr\_Events\_ForDelete 80](#_Toc424277967)

[tr\_Events\_ForInsert 82](#_Toc424277968)

[tr\_Events\_ForInsert\_Hashtag 83](#_Toc424277969)

[tr\_EventsRequests\_ForDelete 85](#_Toc424277970)

[tr\_EventsRequests\_ForInsert 86](#_Toc424277971)

[tr\_Followers\_ForDelete 87](#_Toc424277972)

[tr\_Followers\_ForDelete 88](#_Toc424277973)

[tr\_Followers\_ForInsert 89](#_Toc424277974)

[tr\_LoginSessions\_ForInsert 90](#_Toc424277975)

[tr\_MessageLists\_ForDelete 91](#_Toc424277976)

[tr\_Notifications\_ForInsert 92](#_Toc424277977)

[tr\_Users\_ForInsert 93](#_Toc424277978)

[Store Procedures 94](#_Toc424277979)

[sp\_adminAdd 95](#_Toc424277980)

[sp\_adminBlogEntryAdd 97](#_Toc424277981)

[sp\_adminBlogEntryDelete 98](#_Toc424277982)

[sp\_adminBlogEntryEdit 99](#_Toc424277983)

[sp\_adminBlogEntryInfo 100](#_Toc424277984)

[sp\_adminCheckExists 101](#_Toc424277985)

[sp\_adminEdit 102](#_Toc424277986)

[sp\_adminEventDelete 103](#_Toc424277987)

[sp\_adminEventInfo 104](#_Toc424277988)

[sp\_adminEventStatusEdit 105](#_Toc424277989)

[sp\_adminInfo 106](#_Toc424277990)

[sp\_adminModify 107](#_Toc424277991)

[sp\_adminPermissions 108](#_Toc424277992)

[sp\_admins 109](#_Toc424277993)

[sp\_adminSettings 110](#_Toc424277994)

[sp\_adminSettingsEdit 111](#_Toc424277995)

[sp\_adminStats 112](#_Toc424277996)

[sp\_adminStatsAdd 113](#_Toc424277997)

[sp\_adminStatsDelete 119](#_Toc424277998)

[sp\_adminStatsInfo 120](#_Toc424277999)

[sp\_adminUsersInfoByUserId 121](#_Toc424278000)

[sp\_adminUsersInfoByUsername 122](#_Toc424278001)

[sp\_adminUsersStatusEdit 123](#_Toc424278002)

[sp\_blog 124](#_Toc424278003)

[sp\_blogAdd 125](#_Toc424278004)

[sp\_blogEdit 126](#_Toc424278005)

[sp\_blogInfo 127](#_Toc424278006)

[sp\_bookmarkEventCheckExists 128](#_Toc424278007)

[sp\_bookmarkEventAdd 129](#_Toc424278008)

[sp\_bookmarkEventDelete 130](#_Toc424278009)

[sp\_calendar 131](#_Toc424278010)

[sp\_completion 132](#_Toc424278011)

[sp\_countries 133](#_Toc424278012)

[sp\_eventAdd 134](#_Toc424278013)

[sp\_eventAddHashtagsAdd 136](#_Toc424278014)

[sp\_eventBoardMessages 137](#_Toc424278015)

[sp\_eventBoardMessagesAdd 138](#_Toc424278016)

[sp\_eventCancelParticipation 139](#_Toc424278017)

[sp\_eventDelete 140](#_Toc424278018)

[sp\_eventInviteAdd 141](#_Toc424278019)

[sp\_eventInfoGet 142](#_Toc424278020)

[sp\_eventModifyGet 143](#_Toc424278021)

[sp\_eventModifySet 144](#_Toc424278022)

[sp\_eventNameByEventId 145](#_Toc424278023)

[sp\_eventOwnerIdByEventId 146](#_Toc424278024)

[Goal 146](#_Toc424278025)

[sp\_eventParticipants 147](#_Toc424278026)

[sp\_eventRecommend 149](#_Toc424278027)

[sp\_eventRemoveParticipant 151](#_Toc424278028)

[sp\_eventRequestAdd 152](#_Toc424278029)

[sp\_eventsAccepted 153](#_Toc424278030)

[sp\_eventsBookmarked 154](#_Toc424278031)

[sp\_eventsCreated 155](#_Toc424278032)

[sp\_eventsRequested 156](#_Toc424278033)

[sp\_exploreStatusSet 157](#_Toc424278034)

[sp\_feedbackAdd 158](#_Toc424278035)

[sp\_feedCityAvailableEvents 159](#_Toc424278036)

[sp\_feedFollowingEvents 160](#_Toc424278037)

[sp\_followers 161](#_Toc424278038)

[sp\_following 162](#_Toc424278039)

[sp\_followUser 163](#_Toc424278040)

[sp\_forgotPasswordEmailByVCode 164](#_Toc424278041)

[sp\_forgotPasswordExists 165](#_Toc424278042)

[sp\_forgotPasswordRecover 166](#_Toc424278043)

[sp\_forgotPasswordRequest 167](#_Toc424278044)

[sp\_garbageCollector 168](#_Toc424278045)

[sp\_getNotificationsNumberByUserId 169](#_Toc424278046)

[sp\_getProfileInfoByUserId 170](#_Toc424278047)

[sp\_getProfileInfoByUsername 171](#_Toc424278048)

[sp\_getUserIdByEmail 172](#_Toc424278049)

[sp\_getUserFullNameByUserId 173](#_Toc424278050)

[sp\_getUserLastEvent 174](#_Toc424278051)

[sp\_getUserLocationByUserId 175](#_Toc424278052)

[sp\_getUserLocationInfoByUserId 176](#_Toc424278053)

[sp\_getUsernameByUserId 177](#_Toc424278054)

[sp\_getUserStatusById 178](#_Toc424278055)

[sp\_isUserFollower 179](#_Toc424278056)

[sp\_isUserOwner 180](#_Toc424278057)

[sp\_isUserParticipant 181](#_Toc424278058)

[sp\_lastRecommendationSet 182](#_Toc424278059)

[sp\_locationCities 183](#_Toc424278060)

[sp\_locationInfoByCityId 184](#_Toc424278061)

[sp\_locationInfoOnlyId 185](#_Toc424278062)

[sp\_locationRequestsAdd 186](#_Toc424278063)

[sp\_locationUsersCount 187](#_Toc424278064)

[sp\_loginEmail 188](#_Toc424278065)

[sp\_loginMobile 189](#_Toc424278066)

[sp\_loginSessionRead 190](#_Toc424278067)

[sp\_loginSessionWrite 191](#_Toc424278068)

[sp\_messageDelete 192](#_Toc424278069)

[sp\_messageListDelete 193](#_Toc424278070)

[sp\_messageListDelete 194](#_Toc424278071)

[sp\_messageLists 195](#_Toc424278072)

[sp\_messages 196](#_Toc424278073)

[sp\_messagesAllRead 197](#_Toc424278074)

[sp\_messagesRead 198](#_Toc424278075)

[sp\_messagesSendAdd 199](#_Toc424278076)

[sp\_messagesSenderReceiverInfo 202](#_Toc424278077)

[sp\_notificationAdd 203](#_Toc424278078)

[sp\_notificationDelete 204](#_Toc424278079)

[sp\_notifications 205](#_Toc424278080)

[sp\_notificationsAllRead 206](#_Toc424278081)

[sp\_register 207](#_Toc424278082)

[sp\_reportError 208](#_Toc424278083)

[sp\_reportEvent 209](#_Toc424278084)

[sp\_reportUser 210](#_Toc424278085)

[sp\_requestAccept 211](#_Toc424278086)

[sp\_requestCheck 212](#_Toc424278087)

[sp\_requestsInfo 213](#_Toc424278088)

[sp\_requestReject 214](#_Toc424278089)

[sp\_requestsAllRead 215](#_Toc424278090)

[sp\_requestsInfo 216](#_Toc424278091)

[sp\_requestsUserEvents 217](#_Toc424278092)

[sp\_reviewAdd 218](#_Toc424278093)

[sp\_reviewCheck 219](#_Toc424278094)

[sp\_reviewEdit 220](#_Toc424278095)

[sp\_reviewRead 221](#_Toc424278096)

[sp\_reviews 222](#_Toc424278097)

[sp\_searchByType 223](#_Toc424278098)

[sp\_searchByHashtag 224](#_Toc424278099)

[sp\_searchByUsername 225](#_Toc424278100)

[sp\_settings 226](#_Toc424278101)

[sp\_settingsAccountEmailEdit 227](#_Toc424278102)

[sp\_settingsAccountGet 228](#_Toc424278103)

[sp\_settingsAccountMobileEdit 229](#_Toc424278104)

[sp\_settingsAccountPassword 230](#_Toc424278105)

[sp\_settingsAccountPasswordEdit 231](#_Toc424278106)

[sp\_getSettingsDeactivate 232](#_Toc424278107)

[sp\_settingsInformationGet 233](#_Toc424278108)

[sp\_settingsInformationEdit 234](#_Toc424278109)

[sp\_settingsAboutSet 235](#_Toc424278110)

[sp\_settingsBirthdateSet 236](#_Toc424278111)

[sp\_settingsCoverSet 237](#_Toc424278112)

[sp\_settingsFirstnameSet 238](#_Toc424278113)

[sp\_settingsGenderSet 239](#_Toc424278114)

[sp\_settingsLastnameSet 240](#_Toc424278115)

[sp\_settingsPhotoSet 241](#_Toc424278116)

[sp\_settingsUsernameSet 242](#_Toc424278117)

[sp\_settingsLocationGet 243](#_Toc424278118)

[sp\_settingsLocationEdit 244](#_Toc424278119)

[sp\_getSettingsPhotoGet 245](#_Toc424278120)

[sp\_settingsPhotoEdit 246](#_Toc424278121)

[sp\_settingsPreferences 247](#_Toc424278122)

[sp\_settingsPereferencesEdit 248](#_Toc424278123)

[sp\_settingsPrivacyGet 249](#_Toc424278124)

[sp\_settingsPrivacyEdit 250](#_Toc424278125)

[sp\_unfollowUser 251](#_Toc424278126)

[sp\_userInfoMaster 252](#_Toc424278127)

[sp\_usernameCheckExists 253](#_Toc424278128)

[sp\_verifyEmail 254](#_Toc424278129)

[Classes 255](#_Toc424278130)

[Errors 257](#_Toc424278131)

[Notification Codes 259](#_Toc424278132)

[Automatic Tasks 262](#_Toc424278133)

[Folders 264](#_Toc424278134)

[AdminPages 265](#_Toc424278135)

[Classes 266](#_Toc424278136)

[EmailTemplates 267](#_Toc424278137)

[Files 268](#_Toc424278138)

[Images 269](#_Toc424278139)

[JS 270](#_Toc424278140)

[Styles 271](#_Toc424278141)

[Pages 272](#_Toc424278142)

[Routing 274](#_Toc424278143)

# Tables

## Admins

### Goal

Storing admins information and permissions.

### Columns

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Null** | **PK** | **FK** | **IX** | **Descriptions** |
| AdminId | smallint |  | X |  |  |  |
| UserId | int |  |  | X |  |  |
| AdminDate | Smalldatetime |  |  |  |  | Default: GETDATE() |
| Status | tinyint |  |  |  |  |  |
| PermAdmins | bit |  |  |  |  |  |
| PermBlog | bit |  |  |  |  |  |
| PermContent | bit |  |  |  |  |  |
| PermComments | bit |  |  |  |  |  |
| PermEvents | bit |  |  |  |  |  |
| PermLocations | bit |  |  |  |  |  |
| PermNewsletter | bit |  |  |  |  |  |
| PermSettings | bit |  |  |  |  |  |
| PermStats | bit |  |  |  |  |  |
| PermSupport | bit |  |  |  |  |  |
| PermUsers | bit |  |  |  |  |  |

### Create Command

CREATE TABLE [dbo].[Admins](

[AdminId] [smallint] IDENTITY(1,1) NOT NULL,

[UserId] [int] NOT NULL,

[AdminDate] [smalldatetime] NOT NULL CONSTRAINT [DF\_Admins\_AdminDate] DEFAULT (getdate()),

[Status] [tinyint] NOT NULL,

[PermAdmins] [bit] NOT NULL,

[PermBlog] [bit] NOT NULL,

[PermContent] [bit] NOT NULL,

[PermComments] [bit] NOT NULL,

[PermEvents] [bit] NOT NULL,

[PermNewsletter] [bit] NOT NULL,

[PermSettings] [bit] NOT NULL,

[PermStats] [bit] NOT NULL,

[PermSupport] [bit] NOT NULL,

[PermUsers] [bit] NOT NULL,

[PermLocations] [bit] NOT NULL,

[PermStatistics] [bit] NOT NULL

) ON [PRIMARY]

### Indexes

### Store Procedures

* sp\_adminAdd
* sp\_adminCheckExists
* sp\_adminEdit
* sp\_adminInfo
* sp\_adminModify
* sp\_adminPermissions
* sp\_admins

## Blog

### Goal

Storing users’ login information

### Columns

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Null** | **PK** | **FK** | **IX** | **Descriptions** |
| EntryId | int |  | X |  | X |  |
| Title | varchar(200) |  |  |  |  |  |
| BrowserTitle | varchar(200) |  |  |  |  |  |
| Body | nvarchar(MAX) |  |  |  |  |  |
| SubmitDate | smalldatetime |  |  |  |  | Default: GETDATE() |
| ViewCount | int |  |  |  |  | Default: 0 |

### Create Command

CREATE TABLE [dbo].[Blog](

[EntryId] [int] IDENTITY(1,1) NOT NULL,

[Title] [varchar](200) NOT NULL,

[BrowserTitle] [varchar](200) NOT NULL,

[Body] [nvarchar](max) NOT NULL,

[SubmitDate] [smalldatetime] NOT NULL CONSTRAINT [DF\_Blog\_SubmitDate] DEFAULT (getdate()),

[ViewCount] [int] NOT NULL CONSTRAINT [DF\_Blog\_ViewCount] DEFAULT ((0)),

CONSTRAINT [PK\_Blog] PRIMARY KEY CLUSTERED

(

[EntryId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

### Indexes

* PK\_Blog

### Store Procedures

* sp\_adminBlogEntryAdd
* sp\_adminBlogEntryDelete
* sp\_adminBlogEntryEdit
* sp\_adminBlogEntryInfo
* sp\_blog
* sp\_blogAdd
* sp\_blogEdit
* sp\_blogInfo

## Bookmarks

### Goal

Storing users’ login information

### Columns

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Null** | **PK** | **FK** | **IX** | **Descriptions** |
| BookmarkId | bigint |  | X |  | X |  |
| UserId | int |  |  | X | X |  |
| EventId | bigint |  |  | X |  |  |
| SubmitDate | smalldatetime |  |  |  |  | Default: GETDATE() |

### Create Command

CREATE TABLE [dbo].[Bookmarks](

[BookmarkId] [bigint] IDENTITY(1,1) NOT NULL,

[UserId] [int] NOT NULL,

[EventId] [bigint] NOT NULL,

[SubmitDate] [smalldatetime] NOT NULL CONSTRAINT [DF\_Bookmarks\_SubmitDate] DEFAULT (getdate()),

CONSTRAINT [PK\_Bookmarks] PRIMARY KEY CLUSTERED

(

[BookmarkId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

### Indexes

* PK\_Bookmarks
* ix\_Bookmarks\_UserId

### Store Procedures

* sp\_bookmarkEventCheckExists
* sp\_bookmarkEventAdd
* sp\_bookmarkEventDelete

## Events

### Goal

Storing events information

### Columns

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Null** | **PK** | **FK** | **IX** | **Descriptions** |
| EventId | bigint |  | X |  | X |  |
| OwnerId | int |  |  | X | X |  |
| Name | nvarchar(50) |  |  |  |  |  |
| Date | Smalldatetime |  |  |  | X |  |
| Participants | smallint |  |  |  |  |  |
| ParticipantsAccepted | smallint |  |  |  |  | Default: 0 |
| ParticipantsRemained | smallint |  |  |  |  |  |
| Status | tinyint |  |  |  |  | 1 available 2 full 3 passed 4 deleted Default: 1 |
| LocationId | int |  |  |  | X |  |
| Address | Nvarchar(255) | X |  |  |  |  |
| TypeId | tinyint |  |  | X |  |  |
| CoverId | Smallint |  |  |  |  |  |
| Languages | varchar(50) |  |  |  |  |  |
| Descriptions | nvarchar(500) | X |  |  |  |  |
| SubmitDate | smalldatetime |  |  |  |  | Default: GETDATE() |
| ModifyDate | smalldatetime | X |  |  |  |  |
| ViewCount | int |  |  |  |  | Default: 0 |
| RequestsCount | int |  |  |  |  | Default: 0 |
| ReportCount |  |  |  |  |  | Default: 0 |

### Create Command

CREATE TABLE [dbo].[Events](

[EventId] [bigint] IDENTITY(1,1) NOT NULL,

[OwnerId] [int] NOT NULL,

[Name] [nvarchar](50) NOT NULL,

[Date] [smalldatetime] NOT NULL,

[Duration] [smallint] NOT NULL,

[Participants] [smallint] NOT NULL,

[ParticipantsAccepted] [smallint] NOT NULL,

[ParticipantsRemained] [smallint] NULL,

[LocationId] [int] NOT NULL,

[Status] [tinyint] NOT NULL,

[Privacy] [tinyint] NOT NULL,

[TypeId] [tinyint] NOT NULL,

[CoverId] [smallint] NOT NULL,

[Languages] [varchar](50) NOT NULL,

[Descriptions] [varchar](500) NULL,

[SubmitDate] [smalldatetime] NOT NULL,

[ModifyDate] [smalldatetime] NULL,

[ViewCount] [int] NOT NULL,

[RequestsCount] [int] NOT NULL,

[ReportCount] [int] NOT NULL,

[HasPhoto] [bit] NOT NULL,

[OpenEvent] [bit] NOT NULL,

CONSTRAINT [PK\_Events] PRIMARY KEY CLUSTERED

(

[EventId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

SET ANSI\_PADDING OFF

GO

ALTER TABLE [dbo].[Events] ADD CONSTRAINT [DF\_Table\_1\_Duration] DEFAULT ((0)) FOR [Duration]

GO

ALTER TABLE [dbo].[Events] ADD CONSTRAINT [DF\_Table\_1\_ParticipantsAccepted] DEFAULT ((0)) FOR [ParticipantsAccepted]

GO

ALTER TABLE [dbo].[Events] ADD CONSTRAINT [DF\_Events\_Status] DEFAULT ((1)) FOR [Status]

GO

ALTER TABLE [dbo].[Events] ADD CONSTRAINT [DF\_Events\_SubmitDate] DEFAULT (getdate()) FOR [SubmitDate]

GO

ALTER TABLE [dbo].[Events] ADD CONSTRAINT [DF\_Events\_ViewCount] DEFAULT ((0)) FOR [ViewCount]

GO

ALTER TABLE [dbo].[Events] ADD CONSTRAINT [DF\_Events\_RequestCount] DEFAULT ((0)) FOR [RequestsCount]

GO

ALTER TABLE [dbo].[Events] ADD CONSTRAINT [DF\_Events\_ReportCount] DEFAULT ((0)) FOR [ReportCount]

GO

### Indexes

* PK\_Events
* ix\_Events\_OwnerId

### Triggers

* tr\_Events\_ForDelete
* tr\_Events\_ForInsert
* tr\_Events\_ForInsert\_aHashtags

### Store Procedures

* sp\_adminEventDelete
* sp\_adminEventInfo
* sp\_adminEventStatusEdit
* sp\_calendar
* sp\_eventAdd
* sp\_eventDelete
* sp\_eventInfoGet
* sp\_eventModifyGet
* sp\_eventModifySet
* sp\_eventNameByEventId
* sp\_eventRecommend
* sp\_eventsAccepted
* sp\_eventsCreated
* sp\_eventsRequested
* sp\_feedCityAvailableEvents
* sp\_feedMapEvents
* sp\_feedFollowingEvents
* sp\_garbageCollector
* sp\_reportEvent
* sp\_requestAccept
* sp\_reviewRead
* sp\_searchByHashtag
* sp\_searchByType

## EventsBoardsMessages

### Goal

Storing users’ login information

### Columns

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Null** | **PK** | **FK** | **IX** | **Descriptions** |
| MessageId | bigint |  | **X** |  |  |  |
| EventId | bigint |  |  | X | X |  |
| UserId | int |  |  | X |  |  |
| SubmitDate | smalldatetime |  |  |  |  | Default: GETDATE() |
| Message | nvarchar(200) |  |  |  |  |  |

### Create Command

CREATE TABLE [dbo].[EventsBoardsMessages](

[MessageId] [bigint] IDENTITY(1,1) NOT NULL,

[EventId] [bigint] NOT NULL,

[UserId] [int] NOT NULL,

[SubmitDate] [smalldatetime] NOT NULL,

[Message] [nvarchar](200) NOT NULL

) ON [PRIMARY]

### Indexes

* ix\_EventsBoardsMessages\_EventId

### Store Procedures

* sp\_eventBoardMessages
* sp\_eventBoardMessagesAdd

## EventsInvites

### Goal

Storing users’ login information

### Columns

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Null** | **PK** | **FK** | **IX** | **Descriptions** |
| EventInviteId | bigint |  | X |  |  |  |
| SenderId | int |  |  | X |  |  |
| ReceiverId | int |  |  | X | X |  |
| EventId | bigint |  |  | X |  |  |
| ExpireDate | smalldatetime |  |  |  |  |  |
| SubmitDate | smalldatetime |  |  |  |  | Default: GETDATE() |

### Create Command

CREATE TABLE [dbo].[EventsInvites](

[EventInviteId] [bigint] IDENTITY(1,1) NOT NULL,

[SenderId] [int] NOT NULL,

[ReceiverId] [int] NOT NULL,

[EventId] [bigint] NOT NULL,

[ExpireDate] [smalldatetime] NOT NULL,

[SubmitDate] [smalldatetime] NOT NULL

) ON [PRIMARY]

### Indexes

* PK\_EventsInvites
* ix\_EventsInvites\_ReceiverId

### Store Procedures

* sp\_eventInviteAdd
* sp\_garbageCollector

## EventsRecommendations

### Goal

Storing users’ login information

### Columns

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Null** | **PK** | **FK** | **IX** | **Descriptions** |
| RecordId | bigint |  | X |  |  |  |
| UserId | int |  |  | X | X |  |
| EventId | bigint |  |  | X |  |  |
| ExpireDate | smalldatetime |  |  |  |  |  |

### Create Command

CREATE TABLE [dbo].[EventsRecommendations](

[RecordId] [bigint] IDENTITY(1,1) NOT NULL,

[UserId] [int] NOT NULL,

[EventId] [bigint] NOT NULL,

[ExpireDate] [smalldatetime] NOT NULL

) ON [PRIMARY]

### Indexes

* PK\_EventsRecommendations
* ix\_EventRecommendations\_UserId

### Store Procedures

## EventsRequests

### Goal

Storing users’ login information

### Columns

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Null** | **PK** | **FK** | **IX** | **Descriptions** |
| RequestId | bigint |  | X |  |  |  |
| UserId | int |  |  | X |  |  |
| EventId | bigint |  |  | X |  |  |
| OwnerId | int |  |  | X | X |  |
| SubmitDate | smalldatetime |  |  |  |  | Default: GETDATE() |
| Message | nvarchar(200) | X |  |  |  |  |

### Create Command

CREATE TABLE [dbo].[EventsRecommendations](

[RecordId] [bigint] IDENTITY(1,1) NOT NULL,

[UserId] [int] NOT NULL,

[EventId] [bigint] NOT NULL,

[ExpireDate] [smalldatetime] NOT NULL

) ON [PRIMARY]

### Indexes

* PK\_EventsRequests
* ix\_EventsRequests\_OwnerId

### Store Procedures

* sp\_eventDelete
* sp\_eventRequestAdd
* sp\_eventRequests
* sp\_eventsRequested
* sp\_garbageCollector
* sp\_requestAccept
* sp\_requestsInfo
* sp\_requestInfo
* sp\_requestReject

## EventsRequestsAccepted

### Goal

Storing users’ login information

### Columns

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Null** | **PK** | **FK** | **IX** | **Descriptions** |
| RecordId | Bigint |  |  |  |  |  |
| UserId | Int |  |  | X |  |  |
| EventId | bigint |  |  | X | X |  |
| AcceptationDate | smalldatetime |  |  |  |  | Default: GETDATE() |

### Create Command

CREATE TABLE [dbo].[EventsRequestsAccepted](

[RecordId] [bigint] IDENTITY(1,1) NOT NULL,

[UserId] [int] NOT NULL,

[EventId] [bigint] NOT NULL,

[AcceptationDate] [smalldatetime] NOT NULL CONSTRAINT [DF\_EventsRequestsAccepted\_AcceptationDate] DEFAULT (getdate())

) ON [PRIMARY]

### Indexes

* ix\_EventsRequestsAccepted\_EventId

### Store Procedures

* sp\_calendar
* sp\_eventDelete
* sp\_eventParticipants
* sp\_eventsAccepted
* sp\_requestAccept

## EventsSeen

### Goal

Storing users’ login information

### Columns

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Null** | **PK** | **FK** | **IX** | **Descriptions** |
| RecordId | bigint |  |  |  |  |  |
| UserId | int |  |  | X |  |  |
| EventId | bigint |  |  | X |  |  |
| ExpireDate | smalldatetime |  |  |  |  | Event’s date |

### Create Command

CREATE TABLE [dbo].[EventsSeen](

[RecordId] [bigint] IDENTITY(1,1) NOT NULL,

[UserId] [int] NOT NULL,

[EventId] [bigint] NOT NULL,

[ExpireDate] [smalldatetime] NOT NULL

) ON [PRIMARY]

### Indexes

### Store Procedures

* sp\_garbageCollector

## Feeds

### Goal

Storing users’ login information

### Columns

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Null** | **PK** | **FK** | **IX** | **Descriptions** |
| FeedId | bigint |  | X |  | X |  |
| OwnerId | int |  |  | X | X |  |
| EventId | bigint |  |  | X |  |  |
| UserId | int |  |  |  |  |  |
| ExpireDate | smalldatetime |  |  |  |  |  |
| CreateDate | smalldatetime |  |  |  |  |  |
| SubmitDate | smalldatetime |  |  |  |  | Default: GETDATE() |

### Create Command

CREATE TABLE [dbo].[EventsSeen](

[RecordId] [bigint] IDENTITY(1,1) NOT NULL,

[UserId] [int] NOT NULL,

[EventId] [bigint] NOT NULL,

[ExpireDate] [smalldatetime] NOT NULL

) ON [PRIMARY]

### Indexes

* PK\_Feed
* ix\_Feed\_OwnerId

### Store Procedures

* sp\_feedFollowingEvents
* sp\_garbageCollector

## Feedbacks

### Goal

Storing users’ login information

### Columns

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Null** | **PK** | **FK** | **IX** | **Descriptions** |
| RecordId | int |  |  |  |  |  |
| UserId | int |  |  | X |  |  |
| Rate | tinyint |  |  |  |  |  |
| Message | Nvarchar(2000) | X |  |  |  |  |
| SubmitDate | smalldatetime |  |  |  |  | Default: GETDATE() |

### Create Command

CREATE TABLE [dbo].[Feedbacks](

[RecordId] [int] IDENTITY(1,1) NOT NULL,

[UserId] [int] NOT NULL,

[Rate] [tinyint] NOT NULL,

[Message] [nvarchar](2000) NULL,

[SubmitDate] [smalldatetime] NOT NULL

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Feedbacks] ADD CONSTRAINT [DF\_Feedbacks\_SubmitDate] DEFAULT (getdate()) FOR [SubmitDate]

GO

### Indexes

### Store Procedures

* sp\_feedbackAdd

## Followers

### Goal

Storing users’ login information

### Columns

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Null** | **PK** | **FK** | **IX** | **Descriptions** |
| FollowId | bigint |  | X |  |  |  |
| FollowerId | int |  |  | X | X |  |
| UserId | int |  |  | X | X |  |
| SubmitDate | smalldatetime |  |  |  |  | Default: GETDATE() |

### Create Command

CREATE TABLE [dbo].[Feeds](

[FeedId] [bigint] IDENTITY(1,1) NOT NULL,

[OwnerId] [int] NOT NULL,

[EventId] [bigint] NOT NULL,

[UserId] [int] NOT NULL,

[ExpireDate] [smalldatetime] NOT NULL,

[CreateDate] [smalldatetime] NOT NULL,

[SubmitDate] [smalldatetime] NOT NULL CONSTRAINT [DF\_Feeds\_SubmitDate] DEFAULT (getdate()),

CONSTRAINT [PK\_Feeds] PRIMARY KEY CLUSTERED

(

[FeedId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

### Indexes

* PK\_Followers
* ix\_Followers\_FollowerId
* ix\_Followers\_UserId

### Triggers

* tr\_Followers\_ForDelete
* tr\_Followers\_ForInsert

### Store Procedures

* sp\_followers
* sp\_following
* sp\_followUser
* sp\_isUserFollower
* sp\_unfollowUser

## ForgotPassword

### Goal

Storing users’ login information

### Columns

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Null** | **PK** | **FK** | **IX** | **Descriptions** |
| RecordId | int |  |  |  |  |  |
| UserId | int |  |  | X |  |  |
| VerificationCode | nvarchar(MAX) |  |  |  |  |  |
| Email | varchar(50) |  |  |  |  |  |
| SubmitDate | smalldatetime |  |  |  |  | Default: GETDATE() |

### Create Command

CREATE TABLE [dbo].[ForgotPassword](

[RecordId] [int] IDENTITY(1,1) NOT NULL,

[UserId] [int] NOT NULL,

[VerificationCode] [nvarchar](max) NOT NULL,

[Email] [varchar](50) NOT NULL,

[SubmitDate] [smalldatetime] NOT NULL CONSTRAINT [DF\_ForgotPassword\_SubmitDate] DEFAULT (getdate())

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

### Indexes

### Store Procedures

* sp\_forgotPasswordEmailByVCode
* sp\_forgotPasswordExists
* sp\_forgotPasswordRecover
* sp\_forgotPasswordRequest
* sp\_garbageCollector

## Hashtags

### Goal

Storing users’ login information

### Columns

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Null** | **PK** | **FK** | **IX** | **Descriptions** |
| RecordId | bigint |  | X |  |  |  |
| EventId | bigint |  |  | X |  |  |
| LocationId | int |  |  | X |  |  |
| HashTag | nvarchar(50) |  |  |  |  |  |
| Lifetime | smalldatetime |  |  |  |  |  |

### Create Command

CREATE TABLE [dbo].[Hashtags](

[RecordId] [bigint] IDENTITY(1,1) NOT NULL,

[EventId] [bigint] NOT NULL,

[LocationId] [int] NOT NULL,

[Hashtag] [nvarchar](50) NOT NULL,

[ExpireDate] [smalldatetime] NOT NULL

) ON [PRIMARY]

### Indexes

### Store Procedures

* sp\_searchByHashtag

## LocationRequests

### Goal

Storing location info

### Columns

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Null** | **PK** | **FK** | **IX** | **Descriptions** |
| RequestId | int |  | X |  |  |  |
| Country | varchar(2) |  |  |  |  |  |
| City | nvarchar(255) |  |  |  |  |  |
| UserId | varchar(50) |  |  | X |  |  |

### Create Command

CREATE TABLE [dbo].[LocationRequests](

[RequestId] [int] IDENTITY(1,1) NOT NULL,

[Country] [nvarchar](100) NOT NULL,

[City] [nvarchar](100) NOT NULL,

[UserId] [int] NOT NULL

) ON [PRIMARY]

### Indexes

* PK\_Locations

### Store Procedures

* sp\_countries
* sp\_getUserLocationInfoByUserId
* sp\_locationCities
* sp\_locationInfoByCityId
* sp\_locationInfoOnlyId

## Locations

### Goal

Storing location info

### Columns

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Null** | **PK** | **FK** | **IX** | **Descriptions** |
| LocationId | int |  | X |  |  |  |
| CountryCode | varchar(2) |  |  |  |  |  |
| CountryName | nvarchar(255) |  |  |  |  |  |
| CityName | varchar(50) | X |  |  |  |  |
| TimeZone | nvarchar(255) | X |  |  |  |  |

### Create Command

CREATE TABLE [dbo].[Invites](

[InvitationId] [bigint] IDENTITY(1,1) NOT NULL,

[Email] [varchar](320) NOT NULL,

[UserId] [int] NOT NULL,

[SubmitDate] [smalldatetime] NOT NULL CONSTRAINT [DF\_Invites\_SubmitDate] DEFAULT (getdate()),

[Type] [smallint] NOT NULL,

[Name] [nvarchar](50) NOT NULL,

[Message] [nvarchar](255) NOT NULL,

CONSTRAINT [PK\_Invites] PRIMARY KEY CLUSTERED

(

[InvitationId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

### Indexes

* PK\_Locations

### Store Procedures

* sp\_countries
* sp\_getUserLocationInfoByUserId
* sp\_locationCities
* sp\_locationInfoByCityId
* sp\_locationInfoOnlyId

## LoginSessions

### Goal

Storing users’ login information

### Columns

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Null** | **PK** | **FK** | **IX** | **Descriptions** |
| RecordId | bigint |  |  |  |  |  |
| UserId | int |  |  | X |  |  |
| VerificationCode | nvarchar(MAX) |  |  |  |  |  |
| ExpireDate | smalldatetime |  |  |  |  |  |

### Create Command

CREATE TABLE [dbo].[LoginSessions](

[RecordId] [bigint] IDENTITY(1,1) NOT NULL,

[UserId] [int] NOT NULL,

[VerificationCode] [nvarchar](max) NOT NULL,

[ExpireDate] [smalldatetime] NOT NULL

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

### Indexes

### Triggers

* tr\_LoginSessions\_ForInsert

### Store Procedures

* sp\_loginSessionRead
* sp\_loginSessionWrite
* sp\_garbageCollector
* sp\_loginSessionRead
* sp\_loginSessionWrite

## MessagesLists

### Goal

Storing users’ login information

### Columns

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Null** | **PK** | **FK** | **IX** | **Descriptions** |
| MessageListId | bigint |  | X |  | X |  |
| OwnerId | int |  |  | X | X |  |
| OtherId | int |  |  | X |  |  |
| Brief | nvarchar(100) |  |  |  |  |  |
| SubmitDate | smalldatetime |  |  |  |  |  |
| Unread | bit |  |  |  |  |  |
| NewCount | tinyint |  |  |  |  | Default: 0 |

### Create Command

CREATE TABLE [dbo].[MessageLists](

[MessageListId] [bigint] IDENTITY(1,1) NOT NULL,

[OwnerId] [int] NOT NULL,

[OtherId] [int] NOT NULL,

[Brief] [nvarchar](100) NOT NULL,

[SubmitDate] [smalldatetime] NOT NULL CONSTRAINT [DF\_MessagesLists\_SubmitDate] DEFAULT (getdate()),

[Unread] [bit] NOT NULL CONSTRAINT [DF\_MessagesLists\_Unread] DEFAULT ((1)),

[NewCount] [tinyint] NOT NULL CONSTRAINT [DF\_MessageLists\_NewCount] DEFAULT ((1)),

CONSTRAINT [PK\_MessageLists] PRIMARY KEY CLUSTERED

(

[MessageListId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

### Indexes

* PK\_MessageLists
* ix\_MessageLists\_OwnerId

### Store Procedures

* sp\_messageListDelete
* sp\_messageLists
* sp\_messages
* sp\_messagesSendAdd

## Messages

### Goal

Storing users’ login information

### Columns

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Null** | **PK** | **FK** | **IX** | **Descriptions** |
| MessageId | bigint |  | X |  | X |  |
| OwnerId | int |  |  | X | X |  |
| OtherId | int |  |  | X |  |  |
| Sender | bit |  |  |  |  | 1: owner / 0: other |
| SubmitDate | smalldatetime |  |  |  |  |  |
| Message | nvarchar(4000) |  |  |  |  |  |
| Unread | bit |  |  |  |  |  |

### Create Command

CREATE TABLE [dbo].[Messages](

[MessageId] [bigint] IDENTITY(1,1) NOT NULL,

[OwnerId] [int] NOT NULL,

[OtherId] [int] NOT NULL,

[Sender] [bit] NOT NULL,

[SubmitDate] [smalldatetime] NOT NULL CONSTRAINT [DF\_Messages\_SubmitDate] DEFAULT (getdate()),

[Message] [nvarchar](4000) NOT NULL,

[Unread] [bigint] NOT NULL CONSTRAINT [DF\_Messages\_Unread] DEFAULT ((1)),

CONSTRAINT [PK\_Messages] PRIMARY KEY CLUSTERED

(

[MessageId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

### Indexes

* PK\_Messages
* ix\_Messages\_OwnerId

### Store Procedures

* sp\_messageDelete
* sp\_messages
* sp\_messagesRead
* sp\_messagesSendAdd

## Notifications

### Goal

Storing website’s settings

### Columns

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Null** | **PK** | **FK** | **IX** | **Descriptions** |
| NotificationId | bigint |  | X |  | X |  |
| UserId | int |  |  | X | X |  |
| NotificationType | smallint |  |  |  |  |  |
| ItemId | bigint |  |  |  |  |  |
| Unread | bit |  |  |  |  | Default: 1 |
| SubmitDate | smalldatetime |  |  |  |  |  |

### Create Command

CREATE TABLE [dbo].[Notifications](

[NotificationId] [bigint] IDENTITY(1,1) NOT NULL,

[UserId] [int] NOT NULL,

[NotificationType] [smallint] NOT NULL,

[ItemId] [bigint] NOT NULL CONSTRAINT [DF\_Notifications\_ItemId] DEFAULT ((0)),

[Unread] [bit] NOT NULL CONSTRAINT [DF\_Notifications\_Unread] DEFAULT ((1)),

[SubmitDate] [smalldatetime] NOT NULL,

CONSTRAINT [PK\_Notifications] PRIMARY KEY CLUSTERED

(

[NotificationId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

### Indexes

* PK\_Notifications
* ix\_Notifications\_UserId

### Triggers

* tr\_Notifications\_ForInsert

### Store Procedures

* sp\_garbageCollector
* sp\_notificationAdd
* sp\_notificationDelete
* sp\_notifications

## Reports

### Goal

Storing website’s settings

### Columns

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Null** | **PK** | **FK** | **IX** | **Descriptions** |
| ReportId | bigint |  | X |  | X |  |
| UserId | int |  |  | X |  |  |
| ItemType | smallint |  |  | X |  | 1 user / 2 event / 3 Error |
| ItemId | bigint |  |  | X |  |  |
| ReportType | smallint |  |  |  |  |  |
| Message | nvarchar(500) | X |  |  |  |  |
| SubmitDate | smalldatetime |  |  |  |  | Default: GETDATE() |

### Create Command

CREATE TABLE [dbo].[Report](

[ReportId] [bigint] IDENTITY(1,1) NOT NULL,

[UserId] [int] NOT NULL,

[ItemType] [smallint] NOT NULL,

[ItemId] [bigint] NOT NULL,

[ReportType] [smallint] NOT NULL,

[Message] [nvarchar](500) NULL,

[SubmitDate] [smalldatetime] NOT NULL,

CONSTRAINT [PK\_Report] PRIMARY KEY CLUSTERED

(

[ReportId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

### Indexes

* PK\_Reports

### Store Procedures

* sp\_reportError
* sp\_reportEvent
* sp\_reportUser

## ReviewRequests

### Goal

Storing website’s settings

### Columns

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Null** | **PK** | **FK** | **IX** | **Descriptions** |
| ReviewRequestId | bigint |  | X |  | X |  |
| OwnerId | int |  |  | X | X |  |
| UserId | int |  |  | X |  |  |
| EventId | bigint |  |  | X |  |  |
| SubmitDate | smalldatetime |  |  |  |  | Default: GETDATE() |

### Create Command

CREATE TABLE [dbo].[ReviewRequests](

[ReviewRequestId] [bigint] IDENTITY(1,1) NOT NULL,

[OwnerId] [int] NOT NULL,

[UserId] [int] NOT NULL,

[EventId] [bigint] NOT NULL,

[SubmitDate] [smalldatetime] NOT NULL,

CONSTRAINT [PK\_ReviewRequests] PRIMARY KEY CLUSTERED

(

[ReviewRequestId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

### Indexes

* PK\_ReviewRequests
* ix\_ReviewRequests\_OwnerId

### Store Procedures

* sp\_reviewAdd
* sp\_reviewEdit
* sp\_reviewRequestDelete
* sp\_reviewRead

## Reviews

### Goal

Storing website’s settings

### Columns

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Null** | **PK** | **FK** | **IX** | **Descriptions** |
| ReviewId | bigint |  | X |  | X |  |
| OwnerId | int |  |  | X |  |  |
| UserId | int |  |  | X | X |  |
| EventId | bigint |  |  | X |  |  |
| Rate | smallint |  |  |  |  |  |
| Comment | nvarchar(50) | X |  |  |  |  |
| SubmitDate | smalldatetime |  |  |  |  | Default: GETDATE() |

### Create Command

CREATE TABLE [dbo].[ReviewRequests](

[ReviewRequestId] [bigint] IDENTITY(1,1) NOT NULL,

[OwnerId] [int] NOT NULL,

[UserId] [int] NOT NULL,

[EventId] [bigint] NOT NULL,

[SubmitDate] [smalldatetime] NOT NULL,

CONSTRAINT [PK\_ReviewRequests] PRIMARY KEY CLUSTERED

(

[ReviewRequestId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

### Indexes

* PK\_Reviews
* ix\_Reviews\_UserId

### Store Procedures

* sp\_reviewAdd
* sp\_reviewCheck
* sp\_reviewEdit
* sp\_reviews

## Settings

### Goal

Storing website’s settings

### Columns

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Null** | **PK** | **FK** | **IX** | **Descriptions** |
| LoginAllow | bit |  |  |  |  |  |
| RegisterAllow | bit |  |  |  |  |  |
| ActivitiesAllow | bit |  |  |  |  |  |
| Status | smallint |  |  |  |  | 1 online / 2 maintenance |

### Create Command

CREATE TABLE [dbo].[Settings](

[LoginAllow] [bit] NOT NULL,

[RegisterAllow] [bit] NOT NULL,

[ActivitiesAllow] [bit] NOT NULL,

[Status] [smallint] NOT NULL

) ON [PRIMARY]

### Indexes

### Store Procedures

* sp\_adminSettings
* sp\_adminSettingsEdit
* sp\_settings

## Statistics

### Goal

Storing website’s settings

### Columns

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Null** | **PK** | **FK** | **IX** | **Descriptions** |
| LoginAllow | bit |  |  |  |  |  |
| RegisterAllow | bit |  |  |  |  |  |
| ActivitiesAllow | bit |  |  |  |  |  |
| Status | smallint |  |  |  |  | 1 online / 2 maintenance |

### Create Command

CREATE TABLE [dbo].[Statistics](

[StatisticId] [int] IDENTITY(1,1) NOT NULL,

[SubmitDate] [smalldatetime] NOT NULL CONSTRAINT [DF\_Statistics\_SubmitDate] DEFAULT (getdate()),

[Admins] [int] NULL,

[Bookmarks] [bigint] NULL,

[BookmarkDistinctUsers] [int] NULL,

[Events] [bigint] NULL,

[EventsAvailable] [bigint] NULL,

[EventsDistinctOwners] [int] NULL,

[EventsParticipantsAverage] [int] NULL,

[EventsParticipantsAcceptedAverage] [int] NULL,

[EventsDistinctLocations] [int] NULL,

[EventsType1] [bigint] NULL,

[EventsType2] [bigint] NULL,

[EventsType3] [bigint] NULL,

[EventsType4] [bigint] NULL,

[EventsType5] [bigint] NULL,

[EventsType6] [bigint] NULL,

[EventsType7] [bigint] NULL,

[EventsType8] [bigint] NULL,

[EventsView] [bigint] NULL,

[EventsRequests] [bigint] NULL,

[EventsBoardsMessages] [bigint] NULL,

[Followers] [bigint] NULL,

[Reviews] [bigint] NULL,

[ReviewsRateAverage] [float] NULL,

[Users] [int] NULL,

[UsersGenderMale] [int] NULL,

[UsersGenderFemale] [int] NULL,

[UsersGenderUnknown] [int] NULL,

[UsersHasPhoto] [int] NULL,

[UsersMessages] [bigint] NULL

) ON [PRIMARY]

### Indexes

### Store Procedures

* sp\_adminSettings
* sp\_adminSettingsEdit
* sp\_settings

## Users

### Goal

Store accounts information.

### Columns

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Null** | **PK** | **FK** | **IX** | **Descriptions** |
| UserId | int |  | X |  | X |  |
| FbId | int |  |  |  |  | Default: 0 |
| Email | varchar(320) |  |  |  |  |  |
| Mobile | bigint | X |  |  |  |  |
| Password | nvarchar(128) |  |  |  |  |  |
| Username | varchar(50) | X |  |  | X |  |
| FirstName | nvarchar(30) | X |  |  |  |  |
| LastName | nvarchar(30) | X |  |  |  |  |
| FailedPasswordAttemptCount | smallint |  |  |  |  | Default: 0 |
| FailedPasswordAttemptDate | smalldatetime | X |  |  |  |  |
| BirthDate | date | X |  |  |  |  |
| MemberSince | smalldatetime |  |  |  |  | Default: GETDATE() |
| LastLogin | smalldatetime |  |  |  |  | Default: GETDATE() |
| Gender | tinyint | X |  |  |  | 1 Male 2 Female 3 Unknown |
| Language | varchar(2) |  |  |  |  | Default: en |
| LocationId | int | X |  |  |  | Default: 0 |
| Status | tinyint |  |  |  |  | 0 Not completed register / 1 active / 2 disabled / 3 user deactivated / 4 FB completion |
| EmailVerified | bit |  |  |  |  | Default: 0 |
| MobileVerified | bit |  |  |  |  | Default: 0 |
| ProfileVerified | bit |  |  |  |  |  |
| HasPhoto | bit |  |  |  |  |  |
| RateScore | int |  |  |  |  | Default: 0 |
| RateCount | int |  |  |  |  | Default: 0 |
| FollowingCount | int |  |  |  |  | Default: 0 |
| FollowersCount | int |  |  |  |  | Default: 0 |
| NotificationsCount | int |  |  |  |  | Default: 0 |
| RequestsCount | int |  |  |  |  | Default: 0 |
| MessagesCount | int |  |  |  |  | Default: 0 |
| InvitesCount | int |  |  |  |  | Default: 0 |
| InvitesSuccessfulCount | int |  |  |  |  | Default: 0 |
| InviteRegisteredCount | int |  |  |  |  | Default: 0 |
| ReportCount | int |  |  |  |  | Default: 0 |
| ReportSubmitCount | int |  |  |  |  | Default: 0 |
| About | nvarchar(255) | X |  |  |  |  |
| FriendsOnly | bit |  |  |  |  | Default: False |
| PublicProfile | bit |  |  |  |  | Default: True |
| EventsCreated | int |  |  |  |  | Default: 0 |
| EventsRequested | int |  |  |  |  | Default: 0 |
| EventsAccepted | int |  |  |  |  | Default: 0 |
| Notifications | bit |  |  |  |  | Default: 0 |
| Sound | bit |  |  |  |  | Default: 0 |
| LastRecommendation | bigint |  |  |  |  | Default: 0 |
| ExploreStatus | bit |  |  |  |  | Default: 1 |
| CoverId | tinyint |  |  |  |  | Default: 1 |

### Create Command

CREATE TABLE [dbo].[Users](

[UserId] [int] IDENTITY(1,1) NOT NULL,

[Email] [varchar](320) NOT NULL,

[Mobile] [bigint] NULL,

[Password] [nvarchar](128) NOT NULL,

[Username] [varchar](50) NULL,

[FirstName] [nvarchar](30) NULL,

[LastName] [nvarchar](30) NULL,

[LastPasswordChanged] [smalldatetime] NOT NULL,

[FailedPasswordAttemptCount] [smallint] NOT NULL,

[FailedPasswordAttemptDate] [smalldatetime] NULL,

[BirthDate] [date] NULL,

[MemberSince] [smalldatetime] NOT NULL,

[LastLogin] [smalldatetime] NOT NULL,

[Gender] [tinyint] NULL,

[Language] [varchar](2) NOT NULL,

[LocationId] [int] NOT NULL,

[Status] [tinyint] NOT NULL,

[EmailVerified] [bit] NOT NULL,

[MobileVerified] [bit] NOT NULL,

[ProfileVerified] [bit] NOT NULL,

[HasPhoto] [bit] NOT NULL,

[RateScore] [int] NOT NULL,

[RateCount] [int] NOT NULL,

[FollowingCount] [int] NOT NULL,

[FollowersCount] [int] NOT NULL,

[NotificationsCount] [int] NOT NULL,

[RequestsCount] [int] NOT NULL,

[MessagesCount] [int] NOT NULL,

[InvitesCount] [int] NOT NULL,

[InvitesSuccessfulCount] [int] NOT NULL,

[InviteRegisteredCount] [int] NOT NULL,

[ReportCount] [int] NOT NULL,

[ReportSubmitCount] [int] NOT NULL,

[About] [nvarchar](255) NULL,

[FriendsOnly] [bit] NOT NULL,

[PublicProfile] [bit] NOT NULL,

[EventsCreated] [int] NOT NULL,

[EventsRequested] [int] NOT NULL,

[EventsAccepted] [int] NOT NULL,

[Notifications] [bit] NOT NULL,

[Sound] [bit] NOT NULL,

[LastRecommendation] [bigint] NOT NULL,

[CoverId] [int] NOT NULL,

CONSTRAINT [PK\_Users] PRIMARY KEY CLUSTERED

(

[UserId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

SET ANSI\_PADDING OFF

GO

ALTER TABLE [dbo].[Users] ADD CONSTRAINT [DF\_Users\_LastPasswordChanged] DEFAULT (getdate()) FOR [LastPasswordChanged]

GO

ALTER TABLE [dbo].[Users] ADD CONSTRAINT [DF\_Users\_FailedPasswordAttemptCount] DEFAULT ((0)) FOR [FailedPasswordAttemptCount]

GO

ALTER TABLE [dbo].[Users] ADD CONSTRAINT [DF\_Users\_MemberSince] DEFAULT (getdate()) FOR [MemberSince]

GO

ALTER TABLE [dbo].[Users] ADD CONSTRAINT [DF\_Users\_LastLogin] DEFAULT (getdate()) FOR [LastLogin]

GO

ALTER TABLE [dbo].[Users] ADD CONSTRAINT [DF\_Users\_Language] DEFAULT ('en') FOR [Language]

GO

ALTER TABLE [dbo].[Users] ADD CONSTRAINT [DF\_Users\_LocationId] DEFAULT ((0)) FOR [LocationId]

GO

ALTER TABLE [dbo].[Users] ADD CONSTRAINT [DF\_Users\_Status] DEFAULT ((0)) FOR [Status]

GO

ALTER TABLE [dbo].[Users] ADD CONSTRAINT [DF\_Users\_EmailVerified] DEFAULT ((0)) FOR [EmailVerified]

GO

ALTER TABLE [dbo].[Users] ADD CONSTRAINT [DF\_Users\_MobileVerified] DEFAULT ((0)) FOR [MobileVerified]

GO

ALTER TABLE [dbo].[Users] ADD CONSTRAINT [DF\_Users\_ProfileVerified] DEFAULT ((0)) FOR [ProfileVerified]

GO

ALTER TABLE [dbo].[Users] ADD CONSTRAINT [DF\_Users\_HasPhoto] DEFAULT ((0)) FOR [HasPhoto]

GO

ALTER TABLE [dbo].[Users] ADD CONSTRAINT [DF\_Users\_RateScore] DEFAULT ((0)) FOR [RateScore]

GO

ALTER TABLE [dbo].[Users] ADD CONSTRAINT [DF\_Users\_RateCount] DEFAULT ((0)) FOR [RateCount]

GO

ALTER TABLE [dbo].[Users] ADD CONSTRAINT [DF\_Users\_FollowCount] DEFAULT ((0)) FOR [FollowingCount]

GO

ALTER TABLE [dbo].[Users] ADD CONSTRAINT [DF\_Users\_FollowerCount] DEFAULT ((0)) FOR [FollowersCount]

GO

ALTER TABLE [dbo].[Users] ADD CONSTRAINT [DF\_Users\_NotificationsCount] DEFAULT ((0)) FOR [NotificationsCount]

GO

ALTER TABLE [dbo].[Users] ADD CONSTRAINT [DF\_Users\_RequestsCount] DEFAULT ((0)) FOR [RequestsCount]

GO

ALTER TABLE [dbo].[Users] ADD CONSTRAINT [DF\_Users\_MessagesCount] DEFAULT ((0)) FOR [MessagesCount]

GO

ALTER TABLE [dbo].[Users] ADD CONSTRAINT [DF\_Users\_InviteCount] DEFAULT ((0)) FOR [InvitesCount]

GO

ALTER TABLE [dbo].[Users] ADD CONSTRAINT [DF\_Users\_InvitesSuccessfulCount] DEFAULT ((0)) FOR [InvitesSuccessfulCount]

GO

ALTER TABLE [dbo].[Users] ADD CONSTRAINT [DF\_Users\_InviteRegisteredCount] DEFAULT ((0)) FOR [InviteRegisteredCount]

GO

ALTER TABLE [dbo].[Users] ADD CONSTRAINT [DF\_Users\_ReportCount] DEFAULT ((0)) FOR [ReportCount]

GO

ALTER TABLE [dbo].[Users] ADD CONSTRAINT [DF\_Users\_ReportSubmitCount] DEFAULT ((0)) FOR [ReportSubmitCount]

GO

ALTER TABLE [dbo].[Users] ADD CONSTRAINT [DF\_Users\_FriendsOnly] DEFAULT ((0)) FOR [FriendsOnly]

GO

ALTER TABLE [dbo].[Users] ADD CONSTRAINT [DF\_Users\_PublicProfile] DEFAULT ((1)) FOR [PublicProfile]

GO

ALTER TABLE [dbo].[Users] ADD CONSTRAINT [DF\_Users\_EventsCreated] DEFAULT ((0)) FOR [EventsCreated]

GO

ALTER TABLE [dbo].[Users] ADD CONSTRAINT [DF\_Users\_EventsRequested] DEFAULT ((0)) FOR [EventsRequested]

GO

ALTER TABLE [dbo].[Users] ADD CONSTRAINT [DF\_Users\_EventsAccepted] DEFAULT ((0)) FOR [EventsAccepted]

GO

ALTER TABLE [dbo].[Users] ADD CONSTRAINT [DF\_Users\_Notifications] DEFAULT ((0)) FOR [Notifications]

GO

ALTER TABLE [dbo].[Users] ADD CONSTRAINT [DF\_Users\_Sound] DEFAULT ((0)) FOR [Sound]

GO

ALTER TABLE [dbo].[Users] ADD CONSTRAINT [DF\_Users\_LastRecommendation] DEFAULT ((0)) FOR [LastRecommendation]

GO

ALTER TABLE [dbo].[Users] ADD CONSTRAINT [DF\_Users\_CoverId] DEFAULT ((1)) FOR [CoverId]

GO

### Indexes

* PK\_Users
* ix\_Users\_Username

### Store Procedures

* sp\_adminInfo
* sp\_admins
* sp\_adminUsersInfoByUserId
* sp\_adminUsersInfoByUsername
* sp\_adminUsersStatusEdit
* sp\_completion
* sp\_eventBoardMessages
* sp\_eventDelete
* sp\_eventInviteAdd
* sp\_eventInfoGet
* sp\_eventParticipants
* sp\_eventRecommend
* sp\_eventRequests
* sp\_exploreStatusSet
* sp\_feedCityAvailableEvents
* sp\_feedFollowingEvents
* sp\_followers
* sp\_following
* sp\_getNotificationsNumberByUserId
* sp\_getProfileInfoByUserId
* sp\_getProfileInfoByUsername
* sp\_getUserIdByEmail
* sp\_getUserFullNameByUserId
* sp\_getUserLocationByUserId
* sp\_getUserLocationInfoByUserId
* sp\_getUsernameByUserId
* sp\_getUserStatusById
* sp\_lastRecommendationSet
* sp\_loginEmail
* sp\_loginMobile
* sp\_messageLists
* sp\_messagesAllRead
* sp\_messagesSenderReceiverInfo
* sp\_notificationsAllRead
* sp\_register
* sp\_reportEvent
* sp\_reportUser
* sp\_requestsAllRead
* sp\_requestsInfo
* sp\_reviewAdd
* sp\_reviewEdit
* sp\_reviewRead
* sp\_searchByUsername
* sp\_settingsAccountEmailEdit
* sp\_settingsAccountGet
* sp\_settingsAccountMobileEdit
* settingsAccountPassword
* sp\_settingsAccountPasswordEdit
* sp\_getSettingsDeactivate
* sp\_settingsInformationGet
* sp\_settingsInformationEdit

sp\_settingsLocationEdit

* sp\_settingAboutSet
* sp\_settingBirthdateSet
* sp\_settingCoverSet
* sp\_settingFirstnameSet
* sp\_settingGenderSet
* sp\_settingLastnameSet
* sp\_settingLocationSet
* sp\_settingPhotoSet
* sp\_settingUsernameSet
* sp\_getSettingsPhoto
* sp\_settingsPhotoEdit
* sp\_settingsPreferences
* sp\_settingsPereferencesEdit
* sp\_settingsPrivacyGet
* sp\_settingsPrivacyEdit
* sp\_userInfoMaster
* sp\_usernameCheckExists
* sp\_verifyEmail

## VerificationEmail

### Goal

Storing website’s settings

### Columns

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Null** | **PK** | **FK** | **IX** | **Descriptions** |
| RecordId | int |  |  |  |  |  |
| UserId | int |  |  | X |  |  |
| VerificationCode | nvarchar(MAX) |  |  |  |  |  |
| SubmitDate | smalldatetime |  |  |  |  | GETDATE() |

### Create Command

CREATE TABLE [dbo].[VerificationEmail](

[RecordId] [int] IDENTITY(1,1) NOT NULL,

[UserId] [int] NOT NULL,

[VerificationCode] [nvarchar](max) NOT NULL,

[SubmitDate] [smalldatetime] NOT NULL

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

### Indexes

### Store Procedures

* sp\_

# Indexes

## ix\_Admins\_UserId

### Table

Admins

### Source Code

CREATE INDEX ix\_Admins\_UserId

ON Admins (UserId)

## ix\_Bookmarks\_UserId

### Table

Bookmarks

### Source Code

CREATE INDEX ix\_Bookmarks\_UserId

ON Bookmarks (UserId DESC)

## ix\_Events\_OwnerId

### Table

Events

### Source Code

CREATE INDEX ix\_Events\_OwnerId

ON Events (OwnerId ASC)

## ix\_Events\_OwnerId

### Table

Events

### Source Code

CREATE INDEX ix\_Events\_Date

ON Events (Date ASC)

## ix\_EventsBoardsMessages\_EventId

### Table

EventsBoardsMessages

### Source Code

CREATE INDEX ix\_EventsBoardsMessages\_EventId

ON EventsBoardsMessages (EventId ASC)

## ix\_EventsInvites\_ReceiverId

### Table

EventsInvites

### Source Code

CREATE INDEX ix\_EventsInvites\_ReceiverId

ON EventsInvites (ReceiverId ASC)

## ix\_EventsRecommendations\_UserId

### Table

EventsRecommendatons

### Source Code

CREATE INDEX ix\_EventsRecommendations\_UserId

ON EventsRecommendations (UserId ASC)

## ix\_EventsRequests\_OwnerId

### Table

EventsRequests

### Source Code

CREATE INDEX ix\_EventsRequests\_OwnerId

ON EventsRequests (OwnerId ASC)

## ix\_EventsRequestsAccepted\_EventId

### Table

EventsRequestsAccepted

### Source Code

CREATE INDEX ix\_EventsRequestsAccepted\_EventId

ON EventsRequestsAccepted (EventId ASC)

## ix\_Feed\_OwnerId

### Table

Followers

### Source Code

CREATE INDEX ix\_Feed\_OwnerId

ON Feeds (OwnerId ASC)

## ix\_Followers\_FollowerId

### Table

Followers

### Source Code

CREATE INDEX ix\_Followers\_FollowerId

ON Followers (FollowerId ASC)

## ix\_Followers\_UserId

### Table

Followers

### Source Code

CREATE INDEX ix\_Followers\_UserId

ON Followers (UserId ASC)

## ix\_Messages\_OwnerId

### Table

Messages

### Source Code

CREATE INDEX ix\_Messages\_OwnerId

ON Messages (OwnerId ASC)

## ix\_MessageLists\_OwnerId

### Table

MessageLists

### Source Code

CREATE INDEX ix\_MessageLists\_OwnerId

ON MessageLists (OwnerId ASC)

## ix\_Notifications\_UserId

### Table

Notifications

### Source Code

CREATE INDEX ix\_Notifications\_UserId

ON Notifications (UserId ASC)

## ix\_ReviewRequests\_OwnerId

### Table

ReviewRequests

### Source Code

CREATE INDEX ix\_ReviewRequests\_OwnerId

ON ReviewRequests (OwnerId ASC)

## ix\_Reviews\_UserId

### Table

Reviews

### Source Code

CREATE INDEX ix\_Reviews\_UserId

ON ReviewRequests (OwnerId ASC)

## ix\_Users\_Username

### Table

Users

### Source Code

CREATE INDEX ix\_Users\_Username

ON Users (Username ASC)

# Triggers

## tr\_Events\_ForDelete

### Source Code

CREATE TRIGGER [dbo].[tr\_Events\_ForDelete]

ON Events

FOR DELETE

AS

BEGIN

DECLARE @UserId int

DECLARE @EventId int

DECLARE @LocationId int

SELECT @UserId = OwnerId FROM deleted

SELECT @EventId = EventId FROM deleted

SELECT @LocationId = LocationId FROM deleted

UPDATE Users

SET EventsCreated = EventsCreated - 1

WHERE (UserId = @UserId)

UPDATE Locations

SET EventsCount = EventsCount - 1

WHERE (LocationId = @LocationId)

DELETE

FROM Bookmarks

WHERE (EventId = @EventId)

DELETE

FROM EventsBoardsMessages

WHERE (EventId = @EventId)

DELETE

FROM EventsInvites

WHERE (EventId = @EventId)

DELETE

FROM EventsRecommendations

WHERE (EventId = @EventId)

DELETE

FROM EventsRequests

WHERE (EventId = @EventId)

DELETE

FROM EventsRequestsAccepted

WHERE (EventId = @EventId)

DELETE

FROM Feed

WHERE (EventId = @EventId)

DELETE

FROM Hashtags

WHERE (EventId = @EventId)

DELETE FROM Report

WHERE (ItemType = 2)

AND (ItemId = @EventId)

END

### Table

* Events

## tr\_Events\_ForInsert

### Source Code

CREATE TRIGGER [dbo].[tr\_Events\_ForInsert]

ON Events

FOR INSERT

AS

BEGIN

DECLARE @UserId int

SELECT @UserId = OwnerId FROM inserted

DECLARE @LocationId int

SELECT @LocationId = LocationId FROM inserted

UPDATE Users

SET EventsCreated = EventsCreated + 1

WHERE (UserId = @UserId)

UPDATE Locations

SET EventsCount = EventsCount + 1

WHERE (LocationId = @LocationId)

END

### Table

* Events

## tr\_Events\_ForInsert\_Hashtag

### Source Code

CREATE TRIGGER [dbo].[tr\_Events\_ForInsert\_Hashtag]

ON Events

FOR INSERT

AS

BEGIN

DECLARE @EventId bigint

SELECT @EventId = EventId FROM inserted

DECLARE @LocationId int

SELECT @LocationId = LocationId FROM inserted

DECLARE @Descriptions nvarchar(500)

SELECT @Descriptions = Descriptions FROM inserted

DECLARE @ExpireDate smalldatetime

SELECT @ExpireDate = [Date] FROM inserted

DECLARE @table1 TABLE (EventId bigint, LocationId int, Descriptions nvarchar(500), [ExpireDate] smalldatetime)

INSERT INTO @table1 (EventId, LocationId, Descriptions, [ExpireDate]) VALUES

(@EventId, @LocationId, @Descriptions, @ExpireDate)

;WITH base AS (

SELECT eventID, locationID, LEFT(RIGHT(descriptions, LEN(descriptions)+1-CHARINDEX('#',descriptions)),CHARINDEX(' ',RIGHT(descriptions, LEN(descriptions)+1-CHARINDEX('#',descriptions)))) AS Hashtag, descriptions,

RIGHT(descriptions, LEN(descriptions)-CHARINDEX('#',descriptions)) AS remainingDescriptions, [ExpireDate]

FROM @table1

), rCTE AS (

SELECT a.eventID, locationID, Hashtag, a.Descriptions, remainingDescriptions, [ExpireDate]

FROM base a

UNION ALL

SELECT a.eventID, a.locationID,

LEFT(RIGHT(r.remainingDescriptions, LEN(r.remainingDescriptions)+1-CHARINDEX('#',r.remainingDescriptions)),COALESCE(NULLIF(CHARINDEX(' ',RIGHT(r.remainingDescriptions, LEN(r.remainingDescriptions)+1-CHARINDEX('#',r.remainingDescriptions))),0),LEN(RIGHT(r.remainingDescriptions, LEN(r.remainingDescriptions)+1-CHARINDEX('#',r.remainingDescriptions))))) AS Hashtag, r.descriptions,

RIGHT(r.remainingDescriptions, LEN(r.remainingDescriptions)-CHARINDEX('#',r.remainingDescriptions)) AS remainingDescriptions, a.[ExpireDate]

FROM base a

INNER JOIN rCTE r

ON a.eventID = r.EventId

--AND r.Hashtag + 1 = a.Hashtag

WHERE LEN(r.remainingDescriptions)-LEN(REPLACE(r.remainingDescriptions,'#','')) > 0

)

INSERT INTO Hashtags

(EventId,

LocationId,

Hashtag,

[ExpireDate])

SELECT

eventID,

locationID,

Hashtag,

[ExpireDate]

FROM rCTE

END

### Table

* Events

## tr\_EventsRequests\_ForDelete

### Source Code

CREATE TRIGGER [dbo].[tr\_EventsRequests\_ForDelete]

ON EventsRequests

FOR DELETE

AS

BEGIN

DECLARE @UserId int

SELECT @UserId = OwnerId FROM deleted

DECLARE @EventId int

SELECT @EventId = EventId FROM deleted

UPDATE Events

SET RequestsCount = RequestsCount - 1

WHERE (EventId = @EventId)

UPDATE Users

SET RequestsCount = RequestsCount - 1

WHERE (UserId = @UserId)

END

### Table

* EventsRequests

## tr\_EventsRequests\_ForInsert

### Source Code

CREATE TRIGGER [dbo].[tr\_EventsRequests\_ForInsert]

ON EventsRequests

FOR INSERT

AS

BEGIN

DECLARE @UserId int

SELECT @UserId = OwnerId FROM inserted

DECLARE @EventId int

SELECT @EventId = EventId FROM inserted

UPDATE Events

SET RequestsCount = RequestsCount + 1

WHERE (EventId = @EventId)

UPDATE Users

SET RequestsCount = RequestsCount + 1

WHERE (UserId = @UserId)

END

### Table

* EventsRequests

## tr\_Followers\_ForDelete

### Source Code

CREATE TRIGGER [dbo].[tr\_Events\_ForDelete]

ON Events

FOR DELETE

AS

BEGIN

DECLARE @UserId int

SELECT @UserId = OwnerId FROM deleted

UPDATE Users

SET EventsCreated = EventsCreated - 1

WHERE (UserId = @UserId)

END

### Table

* Followers

## tr\_Followers\_ForDelete

### Source Code

CREATE TRIGGER [dbo].[tr\_Followers\_ForDelete]

ON Followers

FOR DELETE

AS

BEGIN

DECLARE @UserId int

DECLARE @FollowerId int

SELECT @UserId = UserId FROM deleted

SELECT @FollowerId = FollowerId FROM deleted

UPDATE Users

SET FollowersCount = FollowersCount - 1

WHERE (UserId = @UserId)

UPDATE Users

SET FollowingCount = FollowingCount - 1

WHERE (UserId = @FollowerId)

DELETE

FROM Feeds

WHERE (OwnerId = @FollowerId)

AND (UserId = @UserId)

END

### Table

* Followers

## tr\_Followers\_ForInsert

### Source Code

CREATE TRIGGER [dbo].[tr\_Followers\_ForInsert]

ON Followers

FOR INSERT

AS

BEGIN

DECLARE @UserId int

DECLARE @FollowerId int

SELECT @UserId = UserId FROM inserted

SELECT @FollowerId = FollowerId FROM inserted

UPDATE Users

SET FollowersCount = FollowersCount + 1

WHERE (UserId = @UserId)

UPDATE Users

SET FollowingCount = FollowingCount + 1

WHERE (UserId = @FollowerId)

INSERT INTO Feeds

(OwnerId,

EventId,

UserId,

[ExpireDate])

SELECT

@FollowerId AS OwnerId,

EventId,

OwnerId,

[Date]

FROM Events

WHERE (OwnerId = @UserId)

AND ([Date] > GETDATE())

END

### Table

* Followers

## tr\_LoginSessions\_ForInsert

### Source Code

CREATE TRIGGER [dbo].[tr\_LoginSessions\_ForInsert]

ON LoginSessions

FOR INSERT

AS

BEGIN

DECLARE @UserId int

SELECT @UserId = UserId FROM inserted

UPDATE Users

SET LastLogin = GETDATE()

WHERE (UserId = @UserId)

END

### Table

* LoginSessions

## tr\_MessageLists\_ForDelete

### Source Code

CREATE TRIGGER [dbo].[tr\_MessageLists\_ForDelete]

ON MessageLists

FOR DELETE

AS

BEGIN

DECLARE @OwnerId int

DECLARE @OtherId int

SELECT @OwnerId = OwnerId FROM deleted

SELECT @OtherId = OtherId FROM deleted

DELETE FROM [Messages]

WHERE

(OtherId = @OtherId)

AND (OwnerId = @OwnerId)

END

### Table

* Followers

## tr\_Notifications\_ForInsert

### Source Code

CREATE TRIGGER [dbo].[tr\_Notifications\_ForInsert]

ON Notifications

FOR INSERT

AS

BEGIN

DECLARE @UserId int

SELECT @UserId = UserId FROM inserted

UPDATE Users

SET NotificationsCount = NotificationsCount + 1

WHERE (UserId = @UserId)

END

### Table

* Notifications

## tr\_Users\_ForInsert

### Source Code

CREATE TRIGGER [dbo].[tr\_Users\_ForInsert]

ON Users

FOR INSERT

AS

BEGIN

DECLARE @LocationId int

SELECT @LocationId = LocationId FROM inserted

UPDATE Locations

SET UsersCount = UsersCount + 1

WHERE (LocationId = @LocationId)

END

### Table

* Users

# Store Procedures

## sp\_adminAdd

### Goal

Add an admin and the premissions.

### Source Code

CREATE PROCEDURE [dbo].[sp\_adminAdd]

@UserId int,

@Status int,

@PermAdmins bit,

@PermBlog bit,

@PermContent bit,

@PermComments bit,

@PermEvents bit,

@PermLocations bit,

@PermNewsletter bit,

@PermSettings bit,

@PermStats bit,

@PermSupport bit,

@PermUsers bit

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO Admins

(UserId,

[Status],

PermAdmins,

PermBlog,

PermContent,

PermComments,

PermEvents,

PermLocations,

PermNewsletter,

PermSettings,

PermStats,

PermSupport,

PermUsers)

VALUES

(@UserId,

@Status,

@PermAdmins,

@PermBlog,

@PermContent,

@PermComments,

@PermEvents,

@PermLocations,

@PermNewsletter,

@PermSettings,

@PermStats,

@PermSupport,

@PermUsers)

END

GO

### Files

## sp\_adminBlogEntryAdd

### Goal

Add an admin and the premissions.

### Source Code

CREATE PROCEDURE [dbo].[sp\_adminBlogEntryAdd]

@Title varchar(200),

@BrowserTitle varchar(200),

@Body nvarchar(MAX)

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO Blog

(Title,

BrowserTitle,

Body)

VALUES

(@Title,

@BrowserTitle,

@Body)

END

GO

### Files

## sp\_adminBlogEntryDelete

### Goal

Add an admin and the premissions.

### Source Code

CREATE PROCEDURE [dbo].[sp\_adminBlogEntryDelete]

@EntryId int

AS

BEGIN

SET NOCOUNT ON;

DELETE

FROM Blog

WHERE (EntryId = @EntryId)

END

GO

### Files

## sp\_adminBlogEntryEdit

### Goal

Add an admin and the premissions.

### Source Code

CREATE PROCEDURE [dbo].[sp\_adminBlogEntryEdit]

@EntryId int,

@Title varchar(200),

@BrowserTitle varchar(200),

@Body nvarchar(MAX)

AS

BEGIN

SET NOCOUNT ON;

UPDATE Blog

SET Title = @Title,

BrowserTitle = @BrowserTitle,

Body = @Body

WHERE (EntryId = @EntryId)

END

GO

### Files

## sp\_adminBlogEntryInfo

### Goal

Add an admin and the premissions.

### Source Code

CREATE PROCEDURE [dbo].[sp\_adminBlogEntryInfo]

@EntryId int

AS

BEGIN

SET NOCOUNT ON;

SELECT TOP(1)

Title,

Body

FROM Blog

WHERE (EntryId = @EntryId)

END

GO

### Files

## sp\_adminCheckExists

### Goal

Add an admin and the premissions.

### Source Code

CREATE PROCEDURE [dbo].[sp\_adminCheckExists]

@UserId int

AS

BEGIN

SET NOCOUNT ON;

SELECT TOP 1 UserId

FROM Admins

WHERE (UserId = @UserId)

END

GO

### Files

## sp\_adminEdit

### Goal

Add an admin and the premissions.

### Source Code

CREATE PROCEDURE [dbo].[sp\_adminEdit]

@UserId int,

@Status int,

@PermAdmins bit,

@PermBlog bit,

@PermComments bit,

@PermContent bit,

@PermEvents bit,

@PermLocations bit,

@PermNewsletter bit,

@PermSettings bit,

@PermStats bit,

@PremSupport bit,

@PremUsers bit

AS

BEGIN

SET NOCOUNT ON;

UPDATE Admins

SET Status = @Status,

PermAdmins = @PermAdmins,

PermBlog = @PermBlog,

PermComments = @PermComments,

PermContent = @PermContent,

PermEvents = @PermEvents,

PermLocations = @PermLocations,

PermNewsletter = @PermNewsletter,

PermSettings = @PermSettings,

PermStats = @PermStats,

PremSupport = @PremSupport,

PremUsers = @PremUsers

WHERE (UserId = @UserId)

END

GO

### Files

## sp\_adminEventDelete

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_adminEventDelete]

@EventId bigint

AS

BEGIN

SET NOCOUNT ON;

DELETE FROM Events

WHERE (EventId = @EventId)

END

GO

### Files

* Classes/.cs

## sp\_adminEventInfo

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_admineventInfo]

@EventId bigint

AS

BEGIN

SET NOCOUNT ON;

SELECT TOP 1

\*

FROM Events

WHERE (EventId = @EventId)

END

GO

### Files

* Classes/.cs

## sp\_adminEventStatusEdit

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_adminEventStatusEdit]

@EventId bigint,

@Status smallint

AS

BEGIN

SET NOCOUNT ON;

UPDATE Events

SET [Status] = @Status

WHERE (EventId = @EventId)

END

GO

### Files

* Classes/.cs

## sp\_adminInfo

### Goal

Add an admin and the premissions.

### Source Code

CREATE PROCEDURE [dbo].[sp\_adminInfo]

@UserId int

AS

BEGIN

SET NOCOUNT ON;

SELECT TOP 1

Admins.AdminDate,

Admins.Status,

Admins.PermAdmins,

Admins.PermBlog,

Admins.PermContent,

Admins.PermComments,

Admins.PermEvents,

Admins.PermLocations,

Admins.PermNewsletter,

Admins.PermSettings,

Admins.PermStats,

Admins.PermSupport,

Admins.PermUsers,

(Users.FirstName + ' ' + Users.LastName) AS FullName

FROM Admins INNER JOIN

Users ON Admins.UserId = Users.UserId

WHERE (Admins.UserId = @UserId)

END

GO

### Files

## sp\_adminModify

### Goal

Add an admin and the permissions.

### Source Code

CREATE PROCEDURE [dbo].[sp\_adminModify]

@UserId int,

@Status int,

@PermAdmins bit,

@PermBlog bit,

@PermContent bit,

@PermComments bit,

@PermEvents bit,

@PermLocations bit,

@PermNewsletter bit,

@PermSettings bit,

@PermStats bit,

@PermSupport bit,

@PermUsers bit

AS

BEGIN

SET NOCOUNT ON;

UPDATE Admins

SET Status = @Status,

PermAdmins = @PermAdmins,

PermBlog = @PermBlog,

PermContent = @PermContent,

PermComments = @PermComments,

PermEvents = @PermEvents,

PermLocations = @PermLocations,

PermNewsletter = @PermNewsletter,

PermSettings = @PermSettings,

PermStats = @PermStats,

PermSupport = @PermSupport,

PermUsers = @PermUsers

WHERE (UserId = @UserId)

END

GO

### Files

## sp\_adminPermissions

### Goal

Add an admin and the premissions.

### Source Code

CREATE PROCEDURE [dbo].[sp\_adminPermissions]

@UserId int

AS

BEGIN

SET NOCOUNT ON;

SELECT TOP 1

UserId,

[Status],

PermAdmins,

PermBlog,

PermContent,

PermComments,

PermEvents,

PremLocations,

PremNewsletter,

PremSettings,

PremStats,

PremSupport,

PremUsers

FROM Admins

WHERE (UserId = @UserId)

END

GO

### Files

## sp\_admins

### Goal

Add an admin and the permissions.

### Source Code

CREATE PROCEDURE [dbo].[sp\_admins]

AS

BEGIN

SET NOCOUNT ON;

SELECT

Admins.UserId,

Admins.AdminDate,

Admins.Status,

Admins.PermAdmins,

Admins.PermBlog,

Admins.PermContent,

Admins.PermComments,

Admins.PermEvents,

Admins.PermLocations,

Admins.PermNewsletter,

Admins.PermSettings,

Admins.PermStats,

Admins.PermSupport,

Admins.PermUsers,

(Users.FirstName + ' ' + Users.LastName) AS FullName

FROM Admins

INNER JOIN Users ON Admins.UserId = Users.UserId

END

GO

### Files

## sp\_adminSettings

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_adminSettings]

AS

BEGIN

SET NOCOUNT ON;

SELECT \*

FROM Settings

END

GO

### Files

* Classes/.cs

## sp\_adminSettingsEdit

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_adminSettingsEdit]

@LoginAllow bit,

@RegisterAllow bit,

@ActivitiesAllow bit,

@Status smallint

AS

BEGIN

SET NOCOUNT ON;

UPDATE Settings

SET LoginAllow = @LoginAllow,

RegisterAllow = @RegisterAllow,

ActivitiesAllow = @ActivitiesAllow,

[Status] = @Status

END

GO

### Files

* Classes/.cs

## sp\_adminStats

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_adminStats]

AS

BEGIN

SET NOCOUNT ON;

SELECT TOP 1

StatisticId,

SubmitDate

FROM [Statistics]

END

GO

### Files

* Classes/.cs

## sp\_adminStatsAdd

### Goal

Add an admin and the permissions.

### Source Code

USE [iagrrsfr\_appdb]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[sp\_adminStatsAdd] Script Date: Apr/26/15 11:29:00 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER PROCEDURE [dbo].[sp\_adminStatsAdd]

AS

BEGIN

SET NOCOUNT ON;

DECLARE @Admins int

SELECT @Admins =

(

SELECT COUNT(\*)

FROM Admins

)

DECLARE @Bookmarks bigint

SELECT @Bookmarks =

(

SELECT COUNT(\*)

FROM Bookmarks

)

DECLARE @BookmarkDistinctUsers int

SELECT @BookmarkDistinctUsers =

(

SELECT COUNT(DISTINCT UserId)

FROM Bookmarks

)

DECLARE @Events bigint

SELECT @Events =

(

SELECT COUNT(\*)

FROM Events

)

DECLARE @EventsAvailable bigint

SELECT @EventsAvailable =

(

SELECT COUNT(\*)

FROM Events

WHERE (Status = 1)

)

DECLARE @EventsDistinctOwners int

SELECT @EventsDistinctOwners =

(

SELECT COUNT(DISTINCT OwnerId)

FROM Events

)

DECLARE @EventsParticipantsAverage int

SELECT @EventsParticipantsAverage =

(

SELECT AVG(Participants)

FROM Events

)

DECLARE @EventsParticipantsAcceptedAverage int

SELECT @EventsParticipantsAcceptedAverage =

(

SELECT AVG(ParticipantsAccepted)

FROM Events

)

DECLARE @EventsDistinctLocations int

SELECT @EventsDistinctLocations =

(

SELECT COUNT(DISTINCT LocationId)

FROM Events

)

DECLARE @EventsType1 bigint

SELECT @EventsType1 =

(

SELECT COUNT(\*)

FROM Events

WHERE (TypeId = 1)

)

DECLARE @EventsType2 bigint

SELECT @EventsType2 =

(

SELECT COUNT(\*)

FROM Events

WHERE (TypeId = 2)

)

DECLARE @EventsType3 bigint

SELECT @EventsType3 =

(

SELECT COUNT(\*)

FROM Events

WHERE (TypeId = 3)

)

DECLARE @EventsType4 bigint

SELECT @EventsType4 =

(

SELECT COUNT(\*)

FROM Events

WHERE (TypeId = 4)

)

DECLARE @EventsType5 bigint

SELECT @EventsType5 =

(

SELECT COUNT(\*)

FROM Events

WHERE (TypeId = 5)

)

DECLARE @EventsType6 bigint

SELECT @EventsType6 =

(

SELECT COUNT(\*)

FROM Events

WHERE (TypeId = 6)

)

DECLARE @EventsType7 bigint

SELECT @EventsType7 =

(

SELECT COUNT(\*)

FROM Events

WHERE (TypeId = 7)

)

DECLARE @EventsType8 bigint

SELECT @EventsType8 =

(

SELECT COUNT(\*)

FROM Events

WHERE (TypeId = 8)

)

DECLARE @EventsView bigint

SELECT @EventsView =

(

SELECT SUM(ViewCount)

FROM Events

)

DECLARE @EventsRequests bigint

SELECT @EventsRequests =

(

SELECT SUM(RequestsCount)

FROM Users

)

DECLARE @EventsBoardsMessages bigint

SELECT @EventsBoardsMessages =

(

SELECT COUNT(\*)

FROM EventsBoardsMessages

)

DECLARE @Followers bigint

SELECT @Followers =

(

SELECT COUNT(\*)

FROM Followers

)

DECLARE @Reviews bigint

SELECT @Reviews =

(

SELECT COUNT(\*)

FROM Reviews

)

DECLARE @ReviewsRateAverage bigint

SELECT @ReviewsRateAverage =

(

SELECT AVG(Rate)

FROM Reviews

)

DECLARE @Users int

SELECT @Users =

(

SELECT COUNT(\*)

FROM Users

)

DECLARE @UsersGenderMale int

SELECT @UsersGenderMale =

(

SELECT COUNT(\*)

FROM Users

WHERE (Gender = 1)

)

DECLARE @UsersGenderFemale int

SELECT @UsersGenderFemale =

(

SELECT COUNT(\*)

FROM Users

WHERE (Gender = 2)

)

DECLARE @UsersGenderUnknown int

SELECT @UsersGenderUnknown =

(

SELECT COUNT(\*)

FROM Users

WHERE (Gender = 3)

)

DECLARE @UsersHasPhoto int

SELECT @UsersHasPhoto =

(

SELECT COUNT(\*)

FROM Users

WHERE (HasPhoto = 1)

)

DECLARE @UsersMessages bigint

SELECT @UsersMessages =

(

SELECT COUNT(\*)

FROM Messages

)

INSERT INTO [Statistics]

(Admins,

Bookmarks,

BookmarkDistinctUsers,

Events,

EventsAvailable,

EventsDistinctOwners,

EventsParticipantsAverage,

EventsParticipantsAcceptedAverage,

EventsDistinctLocations,

EventsType1,

EventsType2,

EventsType3,

EventsType4,

EventsType5,

EventsType6,

EventsType7,

EventsType8,

EventsView,

EventsRequests,

EventsBoardsMessages,

Followers,

Reviews,

ReviewsRateAverage,

Users,

UsersGenderMale,

UsersGenderFemale,

UsersGenderUnknown,

UsersHasPhoto,

UsersMessages)

VALUES

(@Admins,

@Bookmarks,

@BookmarkDistinctUsers,

@Events,

@EventsAvailable,

@EventsDistinctOwners,

@EventsParticipantsAverage,

@EventsParticipantsAcceptedAverage,

@EventsDistinctLocations,

@EventsType1,

@EventsType2,

@EventsType3,

@EventsType4,

@EventsType5,

@EventsType6,

@EventsType7,

@EventsType8,

@EventsView,

@EventsRequests,

@EventsBoardsMessages,

@Followers,

@Reviews,

@ReviewsRateAverage,

@Users,

@UsersGenderMale,

@UsersGenderFemale,

@UsersGenderUnknown,

@UsersHasPhoto,

@UsersMessages)

END

### Files

## sp\_adminStatsDelete

### Goal

Add an admin and the permissions.

### Source Code

CREATE PROCEDURE [dbo].[sp\_adminStatsDelete]

@StatisticId int

AS

BEGIN

SET NOCOUNT ON;

DELETE FROM [Statistics]

WHERE (StatisticId = @StatisticId)

END

GO

### Files

## sp\_adminStatsInfo

### Goal

Add an admin and the permissions.

### Source Code

CREATE PROCEDURE [dbo].[sp\_adminStatsInfo]

@StatisticId int

AS

BEGIN

SET NOCOUNT ON;

SELECT \*

FROM [Statistics]

WHERE (StatisticId = @StatisticId)

END

GO

### Files

## sp\_adminUsersInfoByUserId

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_adminUsersInfoByUserId]

@UserId int

AS

BEGIN

SET NOCOUNT ON;

SELECT TOP 1

\*

FROM Users

WHERE (UserId = @UserId)

END

GO

### Files

* Classes/.cs

## sp\_adminUsersInfoByUsername

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_adminUsersInfoByUsername]

@Username varchar(50)

AS

BEGIN

SET NOCOUNT ON;

SELECT TOP 1

\*

FROM Users

WHERE (Username = @Username)

END

GO

### Files

* Classes/.cs

## sp\_adminUsersStatusEdit

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_adminUsersStatusEdit]

@UserId int,

@Status smallint

AS

BEGIN

SET NOCOUNT ON;

UPDATE Users

SET [Status] = @Status

WHERE (UserId = @UserId)

END

GO

### Files

* Classes/.cs

## sp\_blog

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_blog]

AS

BEGIN

SET NOCOUNT ON;

SELECT

EntryId,

Title,

BrowserTitle,

Body,

SubmitDate,

ViewCount

FROM Blog

ORDER BY SubmitDate

END

GO

### Files

* Classes/.cs

## sp\_blogAdd

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_blogAdd ]

@Title varchar(200),

@BrowserTitle varchar(200),

@Body nvarchar(MAX)

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO Blog

(Title,

BrowserTitle,

Body)

VALUES

(@Title,

@BrowserTitle,

@Body)

END

GO

### Files

* Classes/.cs

## sp\_blogEdit

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_blogEdit]

@EntryId int,

@Title varchar(200),

@BrowserTitle varchar(200),

@Body nvarchar(4000)

AS

BEGIN

SET NOCOUNT ON;

UPDATE Blog

SET Title = @Title,

BrowserTitle = @BrowserTitle,

Body = @Body

WHERE (EntryId = @EntryId)

END

GO

### Files

* Classes/.cs

## sp\_blogInfo

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_blogInfo]

@EntryId int

AS

BEGIN

SET NOCOUNT ON;

SELECT TOP(1)

Title,

BrowserTitle,

Body,

SubmitDate

FROM Blog

WHERE (EntryId = @EntryId)

UPDATE Blog

SET ViewCount = ViewCount + 1

WHERE (EntryId = @EntryId)

END

GO

### Files

* Classes/.cs

## sp\_bookmarkEventCheckExists

### Goal

Get information settings

### Source Code

CREATE PROCEDURE [dbo].[sp\_bookmarkEventCheckExists]

@EventId bigint,

@UserId int

AS

BEGIN

SET NOCOUNT ON;

SELECT TOP 1

BookmarkId

FROM Bookmarks

WHERE (EventId = @EventId)

AND (UserId = @UserId)

END

GO

### Files

* Profile.aspx.cs

## sp\_bookmarkEventAdd

### Goal

Get information settings

### Source Code

CREATE PROCEDURE [dbo].[sp\_bookmarkEventAdd]

@EventId bigint,

@UserId int

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO Bookmarks

(UserId,

EventId)

VALUES

(@UserId,

@EventId)

END

GO

### Files

* Profile.aspx.cs

## sp\_bookmarkEventDelete

### Goal

Get information settings

### Source Code

CREATE PROCEDURE [dbo].[sp\_bookmarkEventDelete]

@EventId bigint,

@UserId int

AS

BEGIN

SET NOCOUNT ON;

DELETE FROM Bookmarks

WHERE (EventId = @EventId)

AND (UserId = @UserId)

END

GO

### Files

* Profile.aspx.cs

## sp\_calendar

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_calendar]

@UserId int

AS

BEGIN

SET NOCOUNT ON;

SELECT

EventId,

Name,

[Date]

FROM

Events

WHERE

(OwnerId = @UserId)

AND ([Date] > GETDATE())

SELECT

EventsRequestsAccepted.EventId,

Events.Name,

Events.Date

FROM

EventsRequestsAccepted INNER JOIN

Events ON

EventsRequestsAccepted.EventId = Events.EventId

WHERE

(EventsRequestsAccepted.UserId = @UserId)

AND ([Date] > GETDATE())

END

GO

### Files

* Classes/SiteSettings.cs

## sp\_completion

### Goal

Get information settings

### Source Code

CREATE PROCEDURE [dbo].[sp\_completion]

@UserId int,

@Username varchar(50),

@FirstName nvarchar(30),

@LastName nvarchar(30),

@BirthDate date,

@Gender tinyint,

@LocationId int,

@HasPhoto bit

AS

BEGIN

SET NOCOUNT ON;

UPDATE Users

SET Username = @Username,

FirstName = @FirstName,

LastName = @LastName,

BirthDate = @BirthDate,

Gender = @Gender,

LocationId = @LocationId,

[Status] = 1,

HasPhoto = @HasPhoto

WHERE

(UserId = @UserId)

END

GO

### Files

* Settings.aspx.cs

## sp\_countries

### Goal

Get location settings

### Source Code

CREATE PROCEDURE [dbo].[sp\_countries]

AS

BEGIN

SET NOCOUNT ON;

SELECT DISTINCT

CountryCode,

CountryName

FROM Locations

ORDER BY CountryName

END

GO

### Files

* Settings.aspx.cs

## sp\_eventAdd

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_eventAdd]

@UserId int,

@Name nvarchar(50),

@Date smalldatetime,

@Participants smallint,

@LocationId int,

@Address nvarchar(255),

@TypeId tinyint,

@CoverId tinyint,

@Languages varchar(50),

@Descriptions nvarchar(500)

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO Events

(OwnerId,

Name,

Date,

Participants,

ParticipantsRemained,

LocationId,

Address,

TypeId,

Languages,

Descriptions,

CoverId

)

VALUES

(@UserId,

@Name,

@Date,

@Participants,

@Participants,

@LocationId,

@Address,

@TypeId,

@Languages,

@Descriptions,

@CoverId

)

END

GO

### Files

* Classes/SiteSettings.cs

## sp\_eventAddHashtagsAdd

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_eventAddHashtagsAdd]

@HashtagsTable TagType READONLY

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO Hashtags

(EventId,

LocationId,

Hashtag,

[ExpireDate])

SELECT

EventId,

LocationId,

Hashtag,

[ExpireDate]

FROM @HashtagsTable

END

GO

### Files

* Classes/SiteSettings.cs

## sp\_eventBoardMessages

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_eventBoardMessages]

@EventId bigint

AS

BEGIN

SET NOCOUNT ON;

SELECT

EventsBoardsMessages.MessageId,

EventsBoardsMessages.UserId,

EventsBoardsMessages.SubmitDate,

EventsBoardsMessages.Message,

Users.HasPhoto,

Users.FirstName,

Users.LastName

FROM

dbo.EventsBoardsMessages INNER JOIN

Users ON

EventsBoardsMessages.UserId = Users.UserId

WHERE

(EventsBoardsMessages.EventId = @EventId)

ORDER BY EventsBoardsMessages.SubmitDate

END

GO

### Files

* Classes/.cs

## sp\_eventBoardMessagesAdd

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_eventBoardMessagesAdd]

@EventId bigint,

@UserId int,

@Message nvarchar(200)

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO EventsBoardsMessages

(EventId,

UserId,

[Message])

VALUES

(@EventId,

@UserId,

@Message)

END

GO

### Files

* Classes/.cs

## sp\_eventCancelParticipation

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_eventCancelParticipation]

@EventId bigint,

@UserId int

AS

BEGIN

SET NOCOUNT ON;

DELETE FROM EventsRequestsAccepted

WHERE (UserId = @UserId)

AND (EventId = @EventId)

UPDATE Events

SET ParticipantsAccepted = ParticipantsAccepted - 1,

ParticipantsRemained = ParticipantsRemained + 1

WHERE (EventId = @EventId)

UPDATE Users

SET EventsAccepted = EventsAccepted - 1

WHERE (UserId = @UserId)

END

GO

### Files

* Classes/SiteSettings.cs

## sp\_eventDelete

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_eventDelete]

@UserId int,

@EventId bigint

AS

BEGIN

SET NOCOUNT ON;

DELETE FROM [Events]

WHERE (EventId = @EventId)

END

GO

### Files

* Classes/SiteSettings.cs

## sp\_eventInviteAdd

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_eventInviteAdd]

@SenderId int,

@ReceiverId int,

@EventId bigint,

@ExpireDate smalldatetime

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO EventsInvites

(SenderId,

ReceiverId,

EventId,

ExpireDate)

VALUES

(@SenderId,

@ReceiverId,

@EventId,

@ExpireDate)

UPDATE Users

SET RequestsCount = RequestsCount + 1

WHERE (UserId = @ReceiverId)

END

GO

### Files

* ForgotPassword.aspx.cs

## sp\_eventInfoGet

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_eventInfoGet]

@EventId bigint

AS

BEGIN

SET NOCOUNT ON;

SELECT TOP 1

Events.EventId,

Events.Name,

Events.OwnerId,

Events.Date,

Events.Participants,

Events.ParticipantsAccepted,

Events.ParticipantsRemained,

Events.LocationId,

Events.Address,

Events.TypeId,

Events.CoverId,

Events.Languages,

Events.Descriptions,

Events.Status,

Users.HasPhoto AS OwnerHasPhoto,

Users.Username,

Users.RateScore,

Users.RateCount,

Users.FirstName + ' ' + Users.LastName AS OwnerName

FROM

dbo.Events INNER JOIN

Users ON

Events.OwnerId = Users.UserId

WHERE

(Events.EventId = @EventId)

END

GO

### Files

* Classes/SiteSettings.cs

## sp\_eventModifyGet

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_eventModifyGet]

@EventId bigint

AS

BEGIN

SET NOCOUNT ON;

SELECT TOP 1

Name,

OwnerId,

[Date],

Participants,

ParticipantsAccepted,

ParticipantsRemained,

LocationId,

[Address],

TypeId,

Languages,

Descriptions,

[Status]

FROM

Events

WHERE

(EventId = @EventId)

END

GO

### Files

* Classes/SiteSettings.cs

## sp\_eventModifySet

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_eventModifySet]

@EventId bigint,

@Name nvarchar(50),

@Date smalldatetime,

@Participants int,

@LocationId int,

@Address nvarchar(255),

@TypeId tinyint,

@Languages varchar(50),

@Descriptions nvarchar(500),

@CoverId smallint

AS

BEGIN

SET NOCOUNT ON;

UPDATE Events

SET Name = @Name,

[Date] = @Date,

Participants = @Participants,

LocationId = @LocationId,

[Address] = @Address,

TypeId = @TypeId,

Languages = @Languages,

Descriptions = @Descriptions,

CoverId = @CoverId

WHERE (EventId = @EventId)

END

GO

### Files

* Classes/SiteSettings.cs

## sp\_eventNameByEventId

### Goal

Add an admin and the premissions.

### Source Code

CREATE PROCEDURE [dbo].[sp\_eventNameByEventId]

@EventId bigint

AS

BEGIN

SET NOCOUNT ON;

SELECT TOP 1 Name

FROM Events

WHERE (EventId = @EventId)

END

GO

### Files

## sp\_eventOwnerIdByEventId

## Goal

Add an admin and the premissions.

### Source Code

CREATE PROCEDURE [dbo].[sp\_eventOwnerIdByEventId]

@EventId bigint

AS

BEGIN

SET NOCOUNT ON;

SELECT TOP 1 OwnerId

FROM Events

WHERE (EventId = @EventId)

END

GO

### Files

## sp\_eventParticipants

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_sp\_eventParticipants]

@EventId int

AS

BEGIN

SET NOCOUNT ON;

SELECT TOP 1

Events.OwnerId AS UserId,

Users.FirstName,

Users.LastName,

Users.HasPhoto

FROM Events INNER JOIN

Users ON Events.OwnerId = Users.UserId

WHERE

(Events.EventId = @EventId)

UNION

SELECT

EventsRequestsAccepted.UserId,

Users.FirstName,

Users.LastName,

Users.HasPhoto

FROM EventsRequestsAccepted INNER JOIN

Users ON EventsRequestsAccepted.UserId = Users.UserId

WHERE

(EventsRequestsAccepted.EventId = @EventId)

### Files

* Classes/.cs

### sp\_eventParticipantsRemained

### Goal

Add an admin and the premissions.

### Source Code

CREATE PROCEDURE [dbo].[sp\_eventParticipantsRemained]

@EventId bigint

AS

BEGIN

SET NOCOUNT ON;

SELECT TOP 1 ParticipantsRemained

FROM Events

WHERE (EventId = @EventId)

END

GO

### Files

## sp\_eventRecommend

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_eventRecommend]

@UserId int

AS

BEGIN

SET NOCOUNT ON;

DECLARE @ExploreStatus bit

SELECT @ExploreStatus =

(

SELECT TOP 1

ExploreStatus

FROM Users

WHERE

(UserId = @UserId)

)

IF (@ExploreStatus = 1)

BEGIN

SELECT TOP 1

EventId

FROM

Events

WHERE (OwnerId <> @UserId)

AND ([Status] = 1)

AND (ParticipantsRemained > 0)

AND ([Date] > GETDATE())

AND (LocationId = (SELECT TOP 1 LocationId FROM Users WHERE (UserId = @UserId)))

AND (EventId > (SELECT TOP 1 LastRecommendation FROM Users WHERE (UserId = @UserId)))

ORDER BY EventId

UPDATE Users

SET ExploreStatus = 0

WHERE (UserId = @UserId)

END

ELSE

BEGIN

SELECT TOP 1

EventId

FROM

Events

WHERE

(EventId = (SELECT TOP 1 LastRecommendation FROM Users WHERE (UserId = @UserId)))

END

END

SELECT TOP 1 EventId

FROM Events

ORDER BY EventId DESC

GO

### Files

* Classes/.cs

## sp\_eventRemoveParticipant

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_eventRemoveParticipant]

@EventId bigint,

@UserId int

AS

BEGIN

SET NOCOUNT ON;

DELETE FROM EventsRequestsAccepted

WHERE (UserId = @UserId)

AND (EventId = @EventId)

UPDATE Events

SET ParticipantsAccepted = ParticipantsAccepted - 1,

ParticipantsRemained = ParticipantsRemained + 1

WHERE (EventId = @EventId)

END

GO

### Files

* Classes/.cs

## sp\_eventRequestAdd

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_eventRequestAdd]

@EventId bigint,

@UserId int,

@Message nvarchar(200)

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO EventsRequests

(UserId,

OwnerId,

EventId,

[Message])

VALUES

(@UserId,

(SELECT TOP 1 OwnerId FROM Events WHERE (EventId = @EventId)),

@EventId,

@Message)

END

GO

### Files

* Classes/SiteSettings.cs

## sp\_eventsAccepted

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_eventsAccepted]

@UserId int

AS

BEGIN

SET NOCOUNT ON;

SELECT

EventId,

Name,

[Date],

TypeId

FROM

Events

WHERE

([Date] > GETDATE())

AND (OwnerId = @UserId)

ORDER BY [Date]

END

GO

### Files

* Classes/.cs

## sp\_eventsBookmarked

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_eventsBookmarked]

@UserId int

AS

BEGIN

SET NOCOUNT ON;

SELECT

Bookmarks.EventId,

Events.Name,

Events.TypeId,

Events.Date

FROM

Bookmarks INNER JOIN

Events ON

Bookmarks.EventId = Events.EventId

WHERE

(Bookmarks.UserId = @UserId)

AND (Events.Date > GETDATE())

ORDER BY Events.Date

END

GO

### Files

* Classes/.cs

## sp\_eventsCreated

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_eventsCreated]

@UserId int

AS

BEGIN

SET NOCOUNT ON;

SELECT

EventId,

Name,

[Date]

FROM

Events

WHERE

(OwnerId = @UserId)

AND ([Date] > GETDATE())

ORDER BY [Date]

END

GO

### Files

* Classes/.cs

## sp\_eventsRequested

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_eventsRequested]

@UserId int

AS

BEGIN

SET NOCOUNT ON;

SELECT

EventsRequests.EventId,

Events.Name,

Events.TypeId,

Events.Date

FROM

EventsRequests INNER JOIN

Events ON

EventsRequests.EventId = Events.EventId

WHERE

(EventsRequests.UserId = @UserId)

AND ([Date] > GETDATE())

ORDER BY Events.Date

END

GO

### Files

* Classes/.cs

## sp\_exploreStatusSet

### Goal

Add an admin and the premissions.

### Source Code

CREATE PROCEDURE [dbo].[sp\_exploreStatusSet]

@UserId int

AS

BEGIN

SET NOCOUNT ON;

UPDATE Users

SET ExploreStatus = 1

WHERE (UserId = @UserId)

END

GO

### Files

## sp\_fbCompletion

### Goal

Add an admin and the premissions.

### Source Code

CREATE PROCEDURE [dbo].[sp\_fbCompletion]

@UserId int

AS

BEGIN

SET NOCOUNT ON;

SELECT TOP 1

FirstName,

LastName,

HasPhoto,

Gender

FROM Users

WHERE (UserId = @UserId)

END

GO

### Files

## sp\_feedbackAdd

### Goal

Add an admin and the premissions.

### Source Code

CREATE PROCEDURE [dbo].[sp\_feedbackAdd]

@UserId int,

@Rate int,

@Message nvarchar(2000)

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO Feedbacks

(UserId,

Rate,

[Message])

VALUES

(@UserId,

@Rate,

@Message)

END

GO

### Files

## sp\_feedCityAvailableEvents

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_feedCityAvailableEvents]

@LocationId int

AS

BEGIN

SET NOCOUNT ON;

SELECT Events.EventId,

Events.Name,

Events.Date,

Events.TypeId,

Events.CoverId,

Events.OwnerId,

Events.Participants,

Events.ParticipantsAccepted,

Events.ParticipantsRemained,

Users.HasPhoto,

Users.RateCount,

Users.RateScore,

Users.FirstName + ' ' + Users.LastName AS OwnerName

FROM Events

INNER JOIN Users

ON Events.OwnerId = Users.UserId

WHERE (Events.Status = 1)

AND (Date > GETDATE())

AND (Events.LocationId = @LocationId)

END

GO

### Files

* Classes/.cs

## sp\_feedFollowingEvents

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_feedFollowingEvents]

@OwnerId int

AS

BEGIN

SET NOCOUNT ON;

SELECT Feeds.EventId,

Feeds.UserId,

Events.Name,

Events.Date,

Events.TypeId,

Events.LocationId,

Events.CoverId,

Events.Participants,

Events.ParticipantsAccepted,

Events.ParticipantsRemained,

Users.HasPhoto,

Users.FirstName,

Users.LastName,

Users.RateCount,

Users.RateScore,

Users.FirstName + ' ' + Users.LastName AS OwnerName

FROM Feeds

INNER JOIN Users

ON Feeds.UserId = Users.UserId

INNER JOIN Events

ON Feeds.EventId = Events.EventId

WHERE (Feeds.OwnerId = @OwnerId)

END

GO

### Files

* Classes/.cs

## sp\_followers

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_followers]

@UserId int

AS

BEGIN

SET NOCOUNT ON;

SELECT

Followers.FollowerId,

Users.Username,

Users.HasPhoto,

Users.FirstName,

Users.LastName

FROM Followers

INNER JOIN Users ON Followers.FollowerId = Users.UserId

WHERE (Followers.UserId = @UserId)

ORDER BY Users.FirstName, Users.LastName

END

GO

### Files

* Classes/.cs

## sp\_following

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_following]

@UserId int

AS

BEGIN

SET NOCOUNT ON;

SELECT

Followers.UserId,

Users.Username,

Users.HasPhoto,

Users.FirstName,

Users.LastName

FROM Followers

INNER JOIN Users ON Followers.UserId = Users.UserId

WHERE (Followers.FollowerId = @UserId)

ORDER BY Users.FirstName, Users.LastName

END

GO

### Files

* Classes/.cs

## sp\_followingEvents

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_followingEvents]

@UserId int,

@EventId bigint

AS

BEGIN

SET NOCOUNT ON;

SELECT TOP 5

Events.EventId,

Events.Name,

Events.Date,

Events.TypeId,

Events.CoverId,

Events.OwnerId,

Events.LocationId,

Events.Participants,

Events.ParticipantsAccepted,

Events.ParticipantsRemained,

Users.HasPhoto,

Users.RateCount,

Users.RateScore,

Users.FirstName + ' ' + Users.LastName AS OwnerName

FROM Events

INNER JOIN Users

ON Events.OwnerId = Users.UserId

WHERE (Events.Status = 1)

AND (Events.Date > GETDATE())

AND (Events.OwnerId <> @UserId)

AND (Events.EventId < @EventId)

AND NOT EXISTS (

SELECT 1

FROM EventsRequests

WHERE (Events.EventId = EventsRequests.EventId AND EventsRequests.UserId = @UserId)

)

AND NOT EXISTS (

SELECT 1

FROM EventsRequestsAccepted

WHERE (Events.EventId = EventsRequestsAccepted.EventId AND EventsRequestsAccepted.UserId = @UserId)

)

AND EXISTS (

SELECT 1

FROM Followers

WHERE (Followers.FollowerId = @UserId AND Events.OwnerId = Followers.UserId)

)

ORDER BY Events.EventId DESC

END

GO

### Files

* Classes/.cs

## sp\_followUser

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_followUser]

@UserId int,

@FollowerId int

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO Followers

(FollowerId,

UserId)

VALUES

(@FollowerId,

@UserId)

END

GO

### Files

* Classes/.cs

## sp\_forgotPasswordEmailByVCode

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_forgotPasswordEmailByVCode]

@VCode varchar(50)

AS

BEGIN

SET NOCOUNT ON;

SELECT TOP 1

Email

FROM ForgotPassword

WHERE (VerificationCode = @VCode)

END

GO

### Files

* ForgotPassword.aspx.cs

## sp\_forgotPasswordExists

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_forgotPasswordExists]

@Email varchar(50)

AS

BEGIN

SET NOCOUNT ON;

SELECT TOP 1

VerificationCode

FROM ForgotPassword

WHERE

(Email = @Email)

END

GO

### Files

* ForgotPassword.aspx.cs

## sp\_forgotPasswordRecover

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_forgotPasswordRecover]

@Password nvarchar(128),

@Email varchar(50),

@VCode nvarchar(MAX)

AS

BEGIN

SET NOCOUNT ON;

UPDATE Users

SET [Password] = @Password

WHERE (UserId = (SELECT UserId FROM ForgotPassword WHERE (Email = @Email) AND (VerificationCode = @VCode)))

DELETE

FROM ForgotPassword

WHERE (VerificationCode = @VCode)

END

GO

### Files

* ForgotPassword.aspx.cs

## sp\_forgotPasswordRequest

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_forgotPasswordRequest]

@Email varchar(50),

@UserId int,

@VCode nvarchar(MAX)

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO ForgotPassword

(Email,

UserId,

VerificationCode)

VALUES

(@Email,

@UserId,

@VCode)

END

GO

### Files

* ForgotPassword.aspx.cs

## sp\_garbageCollector

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_garbageCollector]

AS

BEGIN

SET NOCOUNT ON;

DELETE FROM Feeds

WHERE (ExpireDate < GETDATE())

DELETE FROM Notifications

WHERE (SubmitDate < GETDATE()-31)

DELETE FROM LoginSessions

WHERE (ExpireDate < GETDATE())

DELETE FROM ForgotPassword

WHERE (SubmitDate < GETDATE()-2)

DELETE EventsRequests

FROM EventsRequests

INNER JOIN Events

ON EventsRequests.EventId = Events.EventId

WHERE (Events.Date < GETDATE())

DELETE FROM EventInvites

WHERE (ExpireDate < GETDATE()+15)

DELETE FROM EventRecommendation

WHERE (ExpireDate < GETDATE())

UPDATE Events

SET Status = 3

WHERE (Events.Date < GETDATE())

END

### Files

* ForgotPassword.aspx.cs

## sp\_getNotificationsNumberByUserId

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_getNotificationsNumberByUserId]

@UserId int

AS

BEGIN

SET NOCOUNT ON;

SELECT TOP 1

MessagesCount,

NotificationsCount,

RequestsCount

FROM Users

WHERE (UserId = @UserId)

END

GO

### Files

* Classes/UserInfo.cs

## sp\_getProfileInfoByUserId

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_getProfileInfoByUserId]

@UserId int

AS

BEGIN

SET NOCOUNT ON;

SELECT TOP 1

UserId,

Username,

FirstName,

LastName,

LocationId,

About,

[Status],

HasPhoto,

ProfileVerified,

RateScore,

RateCount,

FollowingCount,

FollowersCount,

CoverId

FROM Users

WHERE (UserId = @UserId)

END

GO

### Files

* Classes/UserInfo.cs

## sp\_getProfileInfoByUsername

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_getProfileInfoByUsername]

@Username varchar(50)

AS

BEGIN

SET NOCOUNT ON;

SELECT TOP 1

UserId,

Username,

FirstName,

LastName,

LocationId,

[Status],

About,

HasPhoto,

ProfileVerified,

RateScore,

RateCount,

FollowingCount,

FollowersCount,

CoverId

FROM Users

WHERE (Username = @Username)

END

GO

### Files

* Classes/UserInfo.cs

## sp\_getUserIdByEmail

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_getUserIdByEmail]

@Email varchar(320)

AS

BEGIN

SET NOCOUNT ON;

SELECT TOP 1

UserId

FROM Users

WHERE (Email = @Email)

END

GO

### Files

* Classes/UserInfo.cs

## sp\_getUserFullNameByUserId

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_getUserFullNameByUserId]

@UserId int

AS

BEGIN

SET NOCOUNT ON;

SELECT TOP 1

FirstName,

LastName

FROM Users

WHERE (UserId = @UserId)

END

GO

### Files

* Classes/UserInfo.cs

## sp\_getUserLastEvent

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_getUserLastEvent]

@UserId int

AS

BEGIN

SET NOCOUNT ON;

SELECT TOP 1

EventId

FROM Events

WHERE (OwnerId = @UserId)

ORDER BY EventId DESC

END

GO

### Files

* Classes/UserInfo.cs

## sp\_getUserLocationByUserId

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_getUserLocationByUserId]

@UserId int

AS

BEGIN

SET NOCOUNT ON;

SELECT TOP 1

LocationId

FROM Users

WHERE (UserId = @UserId)

END

GO

### Files

* Classes/UserInfo.cs

## sp\_getUserLocationInfoByUserId

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_getUserLocationInfoByUserId]

@UserId int

AS

BEGIN

SET NOCOUNT ON;

SELECT TOP 1

Locations.CountryName,

Locations.CityName

FROM Locations INNER JOIN

Users ON Locations.LocationId = Users.LocationId

WHERE (Users.UserId = @UserId)

END

GO

### Files

* Classes/UserInfo.cs

## sp\_getUsernameByUserId

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_getUsernameByUserId]

@UserId int

AS

BEGIN

SET NOCOUNT ON;

SELECT TOP 1

Username

FROM Users

WHERE (UserId = @UserId)

END

GO

### Files

* Classes/UserInfo.cs

## sp\_getUserStatusById

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_getUserStatusById]

@UserId int

AS

BEGIN

SET NOCOUNT ON;

SELECT TOP 1 [Status]

FROM Users

WHERE (UserId = @UserId)

END

GO

### Files

* Classes/UserInfo.cs

## sp\_isUserFollower

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_isUserFollower]

@UserId int,

@FollowerId int

AS

BEGIN

SET NOCOUNT ON;

SELECT TOP 1

FollowId

FROM Followers

WHERE

(UserId = @UserId)

AND (FollowerId = @FollowerId)

END

GO

### Files

* Classes/.cs

## sp\_isUserOwner

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_isUserOwner]

@UserId int,

@EventId bigint

AS

BEGIN

SET NOCOUNT ON;

SELECT TOP 1

EventId

FROM Events

WHERE

(OwnerId = @UserId)

AND (EventId = @EventId)

END

GO

### Files

* Classes/.cs

## sp\_isUserParticipant

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_isUserParticipant]

@UserId int,

@EventId bigint

AS

BEGIN

SET NOCOUNT ON;

SELECT TOP 1

RecordId

FROM EventsRequestsAccepted

WHERE

(UserId = @UserId)

AND (EventId = @EventId)

END

GO

### Files

* Classes/.cs

## sp\_lastRecommendationSet

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_lastRecommendationSet]

@UserId int,

@EventId bigint

AS

BEGIN

SET NOCOUNT ON;

UPDATE Users

SET LastRecommendation = @EventId

WHERE (UserId = @UserId)

END

GO

### Files

* Classes/.cs

## sp\_locationCities

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_locationCities]

@CountryCode varchar(2)

AS

BEGIN

SET NOCOUNT ON;

SELECT

LocationId,

CityName

FROM Locations

WHERE (CountryCode = @CountryCode)

ORDER BY CityName

END

GO

### Files

* Login.aspx

## sp\_locationInfoByCityId

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_locationInfoByCityId]

@LocationId int

AS

BEGIN

SET NOCOUNT ON;

SELECT TOP 1 CityName,

CountryCode,

CountryName

FROM Locations

WHERE (LocationId = @LocationId)

END

GO

### Files

* Login.aspx

## sp\_locationInfoOnlyId

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_locationInfoOnlyId]

@LocationId int

AS

BEGIN

SET NOCOUNT ON;

SELECT TOP 1 CountryCode

FROM Locations

WHERE

(LocationId = @LocationId)

END

GO

### Files

* Login.aspx

## sp\_locationRequestsAdd

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_locationRequestAdd]

@UserId int,

@EventId bigint,

@Date smalldatetime

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO EventsRecommendations

(UserId,

EventId,

[ExpireDate]

)

VALUES

(@UserId,

@EventId,

@Date

)

END

GO

### Files

* Classes/SiteSettings.cs

## sp\_locationUsersCount

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_locationUsersCount]

@LocationId int

AS

BEGIN

SET NOCOUNT ON;

SELECT TOP 1 UsersCount

CountryName,

CityName

FROM Locations

WHERE

(LocationId = @LocationId)

END

GO

### Files

* Login.aspx

## sp\_loginEmail

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_loginEmail]

@Email varchar(320),

@Password nvarchar(128)

AS

BEGIN

SET NOCOUNT ON;

SELECT TOP 1

UserId,

[Status]

FROM Users

WHERE

(Email = @Email)

AND (Password = @Password)

END

GO

### Files

* Login.aspx

## sp\_loginMobile

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_loginMobile]

@Mobile int,

@Password varchar(128)

AS

BEGIN

SET NOCOUNT ON;

SELECT TOP 1

UserId,

[Status]

FROM Users

WHERE

(Mobile = @Mobile)

AND (Password = @Password)

END

GO

### Files

* Login.aspx

## sp\_loginSessionRead

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_loginSessionRead]

@VerificationCode nvarchar(MAX)

AS

BEGIN

SET NOCOUNT ON;

SELECT TOP 1

UserId

FROM LoginSessions

WHERE [ExpireDate] > GETDATE()

AND (VerificationCode = @VerificationCode)

END

GO

### Files

* Classes/LoginSession.cs

## sp\_loginSessionWrite

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_loginSessionWrite]

@UserId int,

@VerificationCode nvarchar(MAX),

@ExpireDate smalldatetime

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO LoginSessions

(UserId,

VerificationCode,

[ExpireDate])

VALUES

(@UserId,

@VerificationCode,

@ExpireDate)

END

GO

### Files

* Classes/LoginSession.cs

## sp\_messageDelete

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_messageDelete]

@MessageId bigint,

@OwnerId int

AS

BEGIN

SET NOCOUNT ON;

DELETE FROM Messages

WHERE (MessageId = @MessageId)

AND (OwnerId = @OwnerId)

END

GO

### Files

* Classes/.cs

## sp\_messageListDelete

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_messageDelete]

@MessageId bigint,

@OwnerId int

AS

BEGIN

SET NOCOUNT ON;

DELETE FROM Messages

WHERE (MessageId = @MessageId)

AND (OwnerId = @OwnerId)

END

GO

### Files

* Classes/.cs

## sp\_messageListDelete

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_messageListDelete]

@MessageListId bigint,

@OwnerId int

AS

BEGIN

SET NOCOUNT ON;

DELETE FROM MessageLists

WHERE (MessageListId = @MessageListId)

AND (OwnerId = @OwnerId)

END

GO

### Files

* Classes/.cs

## sp\_messageLists

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_messageLists]

@UserId int

AS

BEGIN

SET NOCOUNT ON;

SELECT

MessageLists.MessageListId,

MessageLists.OtherId,

Users.FirstName,

Users.LastName,

Users.HasPhoto,

MessageLists.SubmitDate,

MessageLists.Brief,

MessageLists.Unread,

MessageLists.NewCount

FROM MessageLists INNER JOIN

Users ON MessageLists.OtherId = Users.UserId

WHERE

(MessageLists.OwnerId = @UserId)

ORDER BY MessageLists.SubmitDate DESC

END

GO

### Files

* Classes/.cs

## sp\_messages

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_messages]

@OwnerId int,

@OtherId int,

@MessageId bigint

AS

BEGIN

SET NOCOUNT ON;

SELECT

MessageId,

OwnerId,

OtherId,

Sender,

SubmitDate,

[Message],

Unread

FROM [Messages]

WHERE

(OtherId = @OtherId)

AND (OwnerId = @OwnerId)

ORDER BY SubmitDate ASC

UPDATE MessageLists

SET Unread = 0,

NewCount = 0

WHERE

(OtherId = @OtherId)

AND (OwnerId = @OwnerId)

AND (MessageId < @MessageId)

END

GO

### Files

* Classes/.cs

## sp\_messagesAllRead

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_messagesAllRead]

@UserId int

AS

BEGIN

SET NOCOUNT ON;

UPDATE Users

SET MessagesCount = 0

WHERE (UserId = @UserId)

END

GO

### Files

* Classes/.cs

## sp\_messagesRead

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_messagesRead]

@OwnerId int,

@OtherId int

AS

BEGIN

SET NOCOUNT ON;

UPDATE Messages

SET Unread = 0

WHERE

(OtherId = @OtherId)

AND (OwnerId = @OwnerId)

AND (Unread = 1)

END

GO

### Files

* Classes/.cs

## sp\_messagesSendAdd

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_messagesSendAdd]

@SenderId int,

@ReceiverId int,

@Message nvarchar(4000),

@Brief nvarchar(100)

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO [Messages]

(OwnerId,

OtherId,

Sender,

[Message],

Unread)

VALUES

(@SenderId,

@ReceiverId,

1,

@Message,

0)

INSERT INTO [Messages]

(OwnerId,

OtherId,

Sender,

[Message],

Unread)

VALUES

(@ReceiverId,

@SenderId,

0,

@Message,

1)

DECLARE @SenderListId int

SELECT @SenderListId =

(

SELECT TOP 1

(MessageListId)

FROM MessageLists

WHERE (OwnerId = @SenderId)

AND (OtherId = @ReceiverId)

)

IF (@SenderListId is null)

BEGIN

INSERT INTO MessageLists

(OwnerId,

OtherId,

Brief,

Unread,

NewCount)

VALUES

(@SenderId,

@ReceiverId,

@Brief,

0,

0)

END

ELSE

BEGIN

UPDATE MessageLists

SET

Brief = @Brief,

SubmitDate = GETDATE(),

Unread = 0

WHERE

(MessageListId = @SenderListId)

END

DECLARE @ReceiverListId int

SELECT @ReceiverListId =

(

SELECT TOP 1

MessageListId

FROM MessageLists

WHERE

(OwnerId = @ReceiverId)

AND (OtherId = @SenderId)

)

IF (@ReceiverListId is null)

BEGIN

INSERT INTO MessageLists

(OwnerId,

OtherId,

Brief,

Unread,

NewCount)

VALUES

(@ReceiverId,

@SenderId,

@Brief,

1,

1)

END

ELSE

BEGIN

UPDATE MessageLists

SET

Brief = @Brief,

SubmitDate = GETDATE(),

Unread = 1,

NewCount = NewCount + 1

WHERE

(MessageListId = @ReceiverListId)

UPDATE Users

SET MessagesCount = MessagesCount + 1

WHERE (UserId = @ReceiverId)

END

END

GO

### Files

* Classes/.cs

## sp\_messagesSenderReceiverInfo

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_messagesSenderReceiverInfo]

@OwnerId int,

@OtherId int

AS

BEGIN

SET NOCOUNT ON;

SELECT TOP 1

FirstName,

LastName,

HasPhoto

FROM Users

WHERE (UserId = @OwnerId)

SELECT TOP 1

FirstName,

LastName,

HasPhoto

FROM Users

WHERE (UserId = @OtherId)

END

GO

### Files

* Classes/.cs

## sp\_nearbyCityAll

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_nearbyCityAll]

@LocationId int,

@UserId int,

@EventId bigint,

@Days smallint

AS

BEGIN

SET NOCOUNT ON;

SELECT TOP 5

Events.EventId,

Events.Name,

Events.Date,

Events.TypeId,

Events.CoverId,

Events.OwnerId,

Events.Participants,

Events.ParticipantsAccepted,

Events.ParticipantsRemained,

Users.HasPhoto,

Users.RateCount,

Users.RateScore,

Users.FirstName + ' ' + Users.LastName AS OwnerName

FROM Events

INNER JOIN Users

ON Events.OwnerId = Users.UserId

WHERE (Events.Status = 1)

AND (Events.Date > GETDATE())

AND (Events.LocationId = @LocationId)

AND (Events.OwnerId <> @UserId)

AND (Events.EventId < @EventId)

AND NOT EXISTS (

SELECT 1

FROM EventsRequests

WHERE (Events.EventId = EventsRequests.EventId AND EventsRequests.UserId = @UserId)

)

AND NOT EXISTS (

SELECT 1

FROM EventsRequestsAccepted

WHERE (Events.EventId = EventsRequestsAccepted.EventId AND EventsRequestsAccepted.UserId = @UserId)

)

ORDER BY Events.EventId DESC

END

GO

### Files

* Classes/.cs

## sp\_nearbyCityDate

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_nearbyCityDate]

@LocationId int,

@EventId bigint,

@Days smallint

AS

BEGIN

SET NOCOUNT ON;

SELECT TOP 5

Events.EventId,

Events.Name,

Events.Date,

Events.TypeId,

Events.CoverId,

Events.OwnerId,

Events.Participants,

Events.ParticipantsAccepted,

Events.ParticipantsRemained,

Users.HasPhoto,

Users.RateCount,

Users.RateScore,

Users.FirstName + ' ' + Users.LastName AS OwnerName

FROM Events

INNER JOIN Users

ON Events.OwnerId = Users.UserId

WHERE (Events.Status = 1)

AND (Events.Date > GETDATE())

AND (Events.Date < GETDATE() + @Days)

AND (Events.LocationId = @LocationId)

AND (Events.EventId < @EventId)

END

GO

### Files

* Classes/.cs

## sp\_notificationAdd

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_notificationAdd]

@UserId int,

@NotificationType smallint,

@ItemId bigint

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO Notifications

(UserId,

NotificationType,

ItemId)

VALUES

(@UserId,

@NotificationType,

@ItemId)

SELECT SCOPE\_IDENTITY() AS NotificationId

END

GO

### Files

* Classes/.cs

## sp\_notificationDelete

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_notificationDelete]

@NotificationId bigint,

@UserId int

AS

BEGIN

SET NOCOUNT ON;

DELETE

FROM Notifications

WHERE (UserId = @UserId)

AND (NotificationId = @NotificationId)

END

GO

### Files

* Classes/.cs

## sp\_notifications

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_notifications]

@UserId int

AS

BEGIN

SET NOCOUNT ON;

SELECT

NotificationId,

NotificationType,

ItemId,

Unread,

SubmitDate

FROM Notifications

WHERE

(UserId = @UserId)

ORDER BY SubmitDate DESC

UPDATE Notifications

SET Unread = 0

WHERE (UserId = @UserId)

END

GO

### Files

* Classes/.cs

## sp\_notificationsAllRead

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_notificationsAllRead]

@UserId int

AS

BEGIN

SET NOCOUNT ON;

UPDATE Users

SET NotificationsCount = 0

WHERE (UserId = @UserId)

UPDATE Notifications

SET Unread = 0

WHERE (UserId = @UserId)

END

GO

### Files

* Classes/.cs

## sp\_register

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_register]

@Email varchar(320),

@Password nvarchar(128) ,

@VCode nvarchar(MAX)

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO Users

(Email,

Password

)

VALUES

(@Email,

@Password

)

END

GO

### Files

* Classes/SiteSettings.cs

## sp\_reportError

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_reportError]

@UserId int,

@Message nvarchar(500)

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO Reports

(UserId,

ItemType,

ItemId,

ReportType,

[Message])

VALUES

(@UserId,

3,

0,

0,

@Message)

END

GO

### Files

* Classes/.cs

## sp\_reportEvent

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_reportEvent]

@UserId int,

@ItemId bigint,

@ReportType smallint,

@Message nvarchar(500)

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO Reports

(UserId,

ItemType,

ItemId,

ReportType,

[Message])

VALUES

(@UserId,

2,

@ItemId,

@ReportType,

@Message)

UPDATE Events

SET ReportCount = ReportCount + 1

WHERE (EventId = @ItemId)

UPDATE Users

SET ReportSubmitCount = ReportSubmitCount + 1

WHERE (UserId = @UserId)

END

GO

### Files

* Classes/.cs

## sp\_reportUser

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_reportUser]

@UserId int,

@ItemId int,

@ReportType smallint,

@Message nvarchar(500)

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO Reports

(UserId,

ItemType,

ItemId,

ReportType,

[Message])

VALUES

(@UserId,

1,

@ItemId,

@ReportType,

@Message)

UPDATE Users

SET ReportCount = ReportCount + 1

WHERE (UserId = @ItemId)

UPDATE Users

SET ReportSubmitCount = ReportSubmitCount + 1

WHERE (UserId = @UserId)

END

GO

### Files

* Classes/.cs

## sp\_requestAccept

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_requestAccept]

@RequestId bigint,

@OwnerId int,

@EventId bigint,

@UserId int

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO EventsRequestsAccepted

(UserId,

EventId)

VALUES

(@UserId,

@EventId)

UPDATE Events

SET ParticipantsAccepted = ParticipantsAccepted + 1,

ParticipantsRemained = ParticipantsRemained - 1

WHERE (EventId = @EventId)

UPDATE Users

SET EventsAccepted = EventsAccepted + 1

WHERE (UserId = @UserId)

DELETE

FROM EventsRequests

WHERE (RequestId = @RequestId)

AND (OwnerId = @OwnerId)

END

GO

### Files

* Classes/.cs

## sp\_requestCheck

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_requestCheck]

@EventId bigint,

@UserId int

AS

BEGIN

SET NOCOUNT ON;

SELECT

RequestId

FROM EventsRequests

WHERE (UserId = @UserId)

AND (EventId = @EventId)

END

GO

### Files

* Classes/.cs

## sp\_requestsInfo

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[ sp\_requestsInfo]

@EventId bigint,

@UserId int

AS

BEGIN

SET NOCOUNT ON;

SELECT

EventsRequests.RequestId,

EventsRequests.EventId,

EventsRequests.UserId,

EventsRequests.SubmitDate,

EventsRequests.Message,

Users.FirstName,

Users.LastName,

Users.HasPhoto

FROM EventsRequests INNER JOIN

Users ON

EventsRequests.UserId = Users.UserId

WHERE (EventsRequests.OwnerId = @UserId)

AND (EventsRequests.EventId = @EventId)

END

GO

### Files

* Classes/.cs

## sp\_requestReject

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_requestReject]

@RequestId bigint

AS

BEGIN

SET NOCOUNT ON;

DELETE

FROM EventsRequests

WHERE (RequestId = @RequestId)

END

GO

### Files

* Classes/.cs

## sp\_requestsAllRead

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_requestsAllRead]

@UserId int

AS

BEGIN

SET NOCOUNT ON;

UPDATE Users

SET RequestsCount = 0

WHERE (UserId = @UserId)

END

GO

### Files

* Classes/.cs

## sp\_requestsInfo

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_requestsInfo]

@EventId bigint,

@UserId int

AS

BEGIN

SET NOCOUNT ON;

SELECT

EventsRequests.RequestId,

EventsRequests.UserId,

EventsRequests.SubmitDate,

EventsRequests.Message,

Users.FirstName,

Users.LastName,

Users.HasPhoto

FROM EventsRequests INNER JOIN

Users ON

EventsRequests.UserId = Users.UserId

WHERE (EventsRequests.OwnerId = @UserId)

AND (EventsRequests.EventId = @EventId)

END

GO

### Files

* Classes/.cs

## sp\_requestsUserEvents

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_requestsUserEvents]

@UserId int

AS

BEGIN

SET NOCOUNT ON;

SELECT

EventId,

Name,

TypeId,

[Date],

RequestsCount,

[Status]

FROM Events

WHERE ([Date] > GETDATE())

AND (OwnerId = @UserId)

END

GO

### Files

* Classes/.cs

## sp\_reviewAdd

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_reviewAdd]

@ReviewRequestId bigint,

@OwnerId int,

@EventId bigint,

@Rate tinyint,

@Comment nvarchar(50)

AS

BEGIN

SET NOCOUNT ON;

DECLARE @UserId int

SELECT @UserId = (SELECT TOP 1 OwnerId FROM ReviewRequests WHERE (ReviewRequestId = @ReviewRequestId))

DECLARE @OldRate tinyint

SELECT @OldRate = (SELECT TOP 1 Rate FROM Reviews WHERE (UserId = @UserId) AND (OwnerId = @OwnerId))

IF (@OldRate is null)

BEGIN

INSERT INTO Reviews

(OwnerId,

UserId,

EventId,

Rate,

Comment)

VALUES

(@OwnerId,

@UserId,

@EventId,

@Rate,

@Comment)

UPDATE Users

SET RateCount = RateCount + 1,

RateScore = RateScore + @Rate

WHERE (UserId = @OwnerId)

END

ELSE

BEGIN

UPDATE Reviews

SET Rate = @Rate,

Comment = @Comment

WHERE (UserId = @UserId)

AND (OwnerId = @OwnerId)

UPDATE Users

SET RateScore = RateScore + (@Rate - @OldRate)

WHERE (UserId = @OwnerId)

END

DELETE

FROM ReviewRequests

WHERE

(ReviewRequestId = @ReviewRequestId)

INSERT INTO Notifications

(UserId,

NotificationType,

ItemId)

VALUES

(@OwnerId,

6,

@UserId)

END

GO

### Files

* Classes/.cs

## sp\_reviewCheck

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_reviewCheck]

@OwnerId int,

@UserId int

AS

BEGIN

SET NOCOUNT ON;

SELECT TOP 1

ReviewId,

Rate,

Comment

FROM

Reviews

WHERE (OwnerId = @OwnerId)

AND (UserId = @UserId)

END

GO

### Files

* Classes/.cs

## sp\_reviewEdit

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_reviewEdit]

@ReviewRequestId bigint,

@OwnerId int,

@UserId int,

@EventId bigint,

@Rate tinyint,

@RateTotal tinyint,

@Comment nvarchar(50)

AS

BEGIN

SET NOCOUNT ON;

DELETE

FROM ReviewRequests

WHERE (ReviewRequests.ReviewRequestId = @ReviewRequestId)

UPDATE Reviews

SET Rate = @Rate,

Comment = @Comment

WHERE (OwnerId = @OwnerId)

AND (UserId = @UserId)

UPDATE Users

SET RateScore = RateScore + @RateTotal

WHERE (UserId = @UserId)

END

GO

### Files

* Classes/.cs

## sp\_reviewRead

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_reviewRead]

@OwnerId int

AS

BEGIN

SET NOCOUNT ON;

SELECT TOP 1

r.ReviewRequestId,

r.UserId,

r.EventId,

u.FirstName,

u.LastName,

u.HasPhoto,

e.Name AS EventName,

e.TypeId,

e.CoverId,

e.OwnerId AS EventOwnerId,

o.HasPhoto AS EventOwnerHasPhoto

FROM

dbo.ReviewRequests r INNER JOIN

Users u ON

r.UserId = u.UserId

INNER JOIN

Events e ON

r.EventId = e.EventId

INNER JOIN Users o

on e.OwnerId = o.UserId

WHERE

(r.OwnerId = @OwnerId)

END

GO

### Files

* Classes/.cs

## sp\_reviewRequestDelete

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_reviewRequestDelete]

@OwnerId int,

@ReviewRequestId bigint

AS

BEGIN

SET NOCOUNT ON;

DELETE FROM ReviewRequests

WHERE (OwnerId = @OwnerId)

AND (ReviewRequestId = @ReviewRequestId)

END

GO

### Files

* Classes/.cs

## sp\_reviews

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_reviews]

@UserId int

AS

BEGIN

SET NOCOUNT ON;

SELECT

Reviews.UserId,

Reviews.Rate,

Reviews.Comment,

Reviews.SubmitDate,

Users.Username,

Users.HasPhoto,

Users.FirstName,

Users.LastName

FROM Reviews

INNER JOIN Users ON Reviews.UserId = Users.UserId

WHERE (Reviews.OwnerId = @UserId)

ORDER BY Reviews.SubmitDate DESC

END

GO

### Files

* Classes/.cs

## sp\_searchByType

### Goal

Get photo settings

### Source Code

CREATE PROCEDURE [dbo].[sp\_searchByType]

@TypeId tinyint,

@LocationId int

AS

BEGIN

SET NOCOUNT ON;

SELECT EventId,

Name,

[Date],

ParticipantsRemained,

TypeId

FROM Events

WHERE (TypeId = @TypeId)

AND (LocationId = @LocationId)

AND (Status = 1)

END

GO

### Files

* Settings.aspx.cs

## sp\_searchByHashtag

### Goal

Get photo settings

### Source Code

CREATE PROCEDURE [dbo].[sp\_searchByHashtag]

@Hashtag nvarchar(50),

@LocationId int

AS

BEGIN

SET NOCOUNT ON;

SELECT

Events.EventId,

Events.Name,

Events.Date,

Events.ParticipantsRemained,

Events.TypeId

FROM

Events INNER JOIN

Hashtags ON

Events.EventId = Hashtags.EventId

WHERE

(Events.Status = 1)

AND (Hashtags.Hashtag = @Hashtag)

AND (Hashtags.LocationId = @LocationId)

END

GO

### Files

* Settings.aspx.cs

## sp\_searchByUsername

### Goal

Get photo settings

### Source Code

CREATE PROCEDURE [dbo].[sp\_searchByUsername]

@Keyword varchar(50)

AS

BEGIN

SET NOCOUNT ON;

SELECT UserId,

Username,

HasPhoto

FROM Users

WHERE (Username LIKE '%' + @Keyword+ '%')

END

GO

### Files

* Settings.aspx.cs

## sp\_settings

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_settings]

AS

BEGIN

SET NOCOUNT ON;

SELECT

loginallow,

RegisterAllow,

[Status]

FROM Settings

END

GO

### Files

* Classes/.cs

## sp\_settingsAccountEmailEdit

### Goal

Get information settings

### Source Code

CREATE PROCEDURE [dbo].[sp\_settingsAccountEmailEdit]

@UserId int,

@Email varchar(320)

AS

BEGIN

SET NOCOUNT ON;

UPDATE Users

SET Email = @Email,

EmailVerified = 0

WHERE

(UserId = @UserId)

END

GO

### Files

* Settings.aspx.cs

## sp\_settingsAccountGet

### Goal

Get locaton settings

### Source Code

CREATE PROCEDURE [dbo].[sp\_settingsAccountGet]

@UserId int

AS

BEGIN

SET NOCOUNT ON;

SELECT TOP 1

Email,

Mobile,

EmailVerified,

MobileVerified

FROM Users

WHERE (UserId = @UserId)

END

GO

### Files

* Settings.aspx.cs

## sp\_settingsAccountMobileEdit

### Goal

Get information settings

### Source Code

CREATE PROCEDURE [dbo].[sp\_settingsAccountEmailEdit]

@UserId int,

@Mobile varchar(320)

AS

BEGIN

SET NOCOUNT ON;

UPDATE Users

SET Mobile = @Mobile,

MobileVerified = 0

WHERE

(UserId = @UserId)

END

GO

### Files

* Settings.aspx.cs

## sp\_settingsAccountPassword

### Goal

Get photo settings

### Source Code

CREATE PROCEDURE [dbo].[sp\_settingsAccountPassword]

@UserId int

AS

BEGIN

SET NOCOUNT ON;

SELECT TOP 1

[Password]

FROM Users

WHERE (UserId = @UserId)

END

GO

### Files

* Settings.aspx.cs

## sp\_settingsAccountPasswordAdd

### Goal

Get information settings

### Source Code

CREATE PROCEDURE [dbo].[sp\_settingsAccountPasswordAdd]

@UserId int,

@Password nvarchar(128)

AS

BEGIN

SET NOCOUNT ON;

UPDATE Users

SET Password = @Password

WHERE

(Password IS NULL)

AND (UserId = @UserId)

END

GO

### Files

* Settings.aspx.cs

## sp\_settingsAccountPasswordEdit

### Goal

Get information settings

### Source Code

CREATE PROCEDURE [dbo].[sp\_settingsAccountPasswordEdit]

@UserId int,

@Password nvarchar(128)

AS

BEGIN

SET NOCOUNT ON;

UPDATE Users

SET Password = @Password

WHERE

(UserId = @UserId)

END

GO

### Files

* Settings.aspx.cs

## sp\_getSettingsDeactivate

### Goal

Get information settings

### Source Code

CREATE PROCEDURE [dbo].[sp\_settingsDeactivate]

@UserId int

AS

BEGIN

SET NOCOUNT ON;

UPDATE Users

SET Status = 3

WHERE

(UserId = @UserId)

END

GO

### Files

* Settings.aspx.cs

## sp\_settingsInformationGet

### Goal

Get information settings

### Source Code

CREATE PROCEDURE [dbo].[sp\_settingsInformationGet]

@UserId int

AS

BEGIN

SET NOCOUNT ON;

SELECT TOP 1

FirstName,

LastName,

Username,

About,

Birthdate,

Gender

FROM Users

WHERE (UserId = @UserId)

END

GO

### Files

* Settings.aspx.cs

## sp\_settingsInformationEdit

### Goal

Get information settings

### Source Code

CREATE PROCEDURE [dbo].[sp\_settingsInformationEdit]

@UserId int,

@Username varchar(50),

@FirstName nvarchar(30),

@LastName nvarchar(30),

@About nvarchar(255),

@BirthDate date,

@Gender tinyint

AS

BEGIN

SET NOCOUNT ON;

UPDATE Users

SET Username = @Username,

FirstName = @FirstName,

LastName = @LastName,

About = @About,

BirthDate = @BirthDate,

Gender = Gender

WHERE

(UserId = @UserId)

END

GO

### Files

* Settings.aspx.cs

## sp\_settingsAboutSet

### Goal

Get information settings

### Source Code

CREATE PROCEDURE [dbo].[sp\_settingsAboutSet]

@UserId int,

@About nvarchar(255)

AS

BEGIN

SET NOCOUNT ON;

UPDATE Users

SET About = @About

WHERE

(UserId = @UserId)

END

GO

### Files

* Settings.aspx.cs

## sp\_settingsBirthdateSet

### Goal

Get information settings

### Source Code

CREATE PROCEDURE [dbo].[sp\_settingsBirthdateSet]

@UserId int,

@Birthdate smalldatetime

AS

BEGIN

SET NOCOUNT ON;

UPDATE Users

SET Birthdate = @Birthdate

WHERE

(UserId = @UserId)

END

GO

### Files

* Settings.aspx.cs

## sp\_settingsCoverSet

### Goal

Get information settings

### Source Code

CREATE PROCEDURE [dbo].[sp\_settingsCoverSet]

@UserId int,

@CoverId int

AS

BEGIN

SET NOCOUNT ON;

UPDATE Users

SET CoverId = @CoverId

WHERE

(UserId = @UserId)

END

GO

### Files

* Settings.aspx.cs

## sp\_settingsFirstnameSet

### Goal

Get information settings

### Source Code

CREATE PROCEDURE [dbo].[sp\_settingsFirstnameSet]

@UserId int,

@FirstName nvarchar(30)

AS

BEGIN

SET NOCOUNT ON;

UPDATE Users

SET FirstName = @FirstName

WHERE

(UserId = @UserId)

END

GO

### Files

* Settings.aspx.cs

## sp\_settingsGenderSet

### Goal

Get information settings

### Source Code

CREATE PROCEDURE [dbo].[sp\_settingsGenderSet]

@UserId int,

@Gender tinyint

AS

BEGIN

SET NOCOUNT ON;

UPDATE Users

SET Gender = @Gender

WHERE

(UserId = @UserId)

END

GO

### Files

* Settings.aspx.cs

## sp\_settingsLastnameSet

### Goal

Get information settings

### Source Code

CREATE PROCEDURE [dbo].[sp\_settingsLastnameSet]

@UserId int,

@LastName nvarchar(30)

AS

BEGIN

SET NOCOUNT ON;

UPDATE Users

SET LastName = @LastName

WHERE

(UserId = @UserId)

END

GO

### Files

* Settings.aspx.cs

## sp\_settingsPhotoSet

### Goal

Get information settings

### Source Code

CREATE PROCEDURE [dbo].[sp\_settingsPhotoSet]

@UserId int,

@HasPhoto bit

AS

BEGIN

SET NOCOUNT ON;

UPDATE Users

SET HasPhoto = @HasPhoto

WHERE

(UserId = @UserId)

END

GO

### Files

* Settings.aspx.cs

## sp\_settingsUsernameSet

### Goal

Get information settings

### Source Code

CREATE PROCEDURE [dbo].[sp\_settingsUsernameSet]

@UserId int,

@Username varchar(50)

AS

BEGIN

SET NOCOUNT ON;

UPDATE Users

SET Username = @Username

WHERE

(UserId = @UserId)

END

GO

### Files

* Settings.aspx.cs

## sp\_settingsLocationGet

### Goal

Get location settings

### Source Code

CREATE PROCEDURE [dbo].[sp\_settingsLocationGet]

@UserId int

AS

BEGIN

SET NOCOUNT ON;

SELECT TOP 1

LocationId

FROM Users

WHERE (UserId = @UserId)

END

GO

### Files

* Settings.aspx.cs

## sp\_settingsLocationEdit

### Goal

Get information settings

### Source Code

CREATE PROCEDURE [dbo].[sp\_settingsLocationEdit]

@UserId int,

@LocationId int

AS

BEGIN

SET NOCOUNT ON;

UPDATE Users

SET LocationId = @LocationId

WHERE

(UserId = @UserId)

END

GO

### Files

* Settings.aspx.cs

## sp\_getSettingsPhotoGet

### Goal

Get photo settings

### Source Code

CREATE PROCEDURE [dbo].[sp\_getSettingsPhotoGet]

@UserId int

AS

BEGIN

SET NOCOUNT ON;

SELECT HasPhoto,

CoverId

FROM Users

WHERE (UserId = @UserId)

END

GO

### Files

* Settings.aspx.cs

## sp\_settingsPhotoEdit

### Goal

Get information settings

### Source Code

CREATE PROCEDURE [dbo].[sp\_settingsPhotoEdit]

@UserId int,

@HasPhoto bit

AS

BEGIN

SET NOCOUNT ON;

UPDATE Users

SET HasPhoto = @HasPhoto

WHERE

(UserId = @UserId)

END

GO

### Files

* Settings.aspx.cs

## sp\_settingsPreferences

### Goal

Get photo settings

### Source Code

CREATE PROCEDURE [dbo].[sp\_settingsPreferences]

@UserId int

AS

BEGIN

SET NOCOUNT ON;

SELECT TOP 1

Notifications,

Sound

FROM Users

WHERE (UserId = @UserId)

END

GO

### Files

* Settings.aspx.cs

## sp\_settingsPereferencesEdit

### Goal

Get information settings

### Source Code

CREATE PROCEDURE [dbo].[sp\_settingsPereferencesEdit]

@UserId int,

@Notifications bit,

@Sound bit

AS

BEGIN

SET NOCOUNT ON;

UPDATE Users

SET Notifications = @Notifications,

Sound = @Sound

WHERE

(UserId = @UserId)

END

GO

### Files

* Settings.aspx.cs

## sp\_settingsPrivacyGet

### Goal

Get photo settings

### Source Code

CREATE PROCEDURE [dbo].[sp\_settingsPrivacyGet]

@UserId int

AS

BEGIN

SET NOCOUNT ON;

SELECT TOP 1

FriendsOnly,

PublicProfile

FROM Users

WHERE (UserId = @UserId)

END

GO

### Files

* Settings.aspx.cs

## sp\_settingsPrivacyEdit

### Goal

Get information settings

### Source Code

CREATE PROCEDURE [dbo].[sp\_settingsPrivacyEdit]

@UserId int,

@FriendsOnly bit,

@PublicProfile bit

AS

BEGIN

SET NOCOUNT ON;

UPDATE Users

SET FriendsOnly = @FriendsOnly,

PublicProfile = @PublicProfile

WHERE

(UserId = @UserId)

END

GO

### Files

* Settings.aspx.cs

## sp\_unfollowUser

### Goal

Store accounts information.

### Source Code

CREATE PROCEDURE [dbo].[sp\_unfollowUser]

@UserId int,

@FollowerId int

AS

BEGIN

SET NOCOUNT ON;

DELETE FROM Followers

WHERE (FollowerId = @FollowerId)

AND (UserId = @UserId)

END

GO

### Files

* Classes/.cs

## sp\_userInfoMaster

### Goal

Get information settings

### Source Code

CREATE PROCEDURE [dbo].[sp\_userInfoMaster]

@UserId int

AS

BEGIN

SET NOCOUNT ON;

SELECT TOP 1

FirstName,

LastName,

Username,

MessagesCount,

RequestsCount,

NotificationsCount,

HasPhoto

FROM Users

WHERE (UserId = @UserId)

END

GO

### Files

* MasterPageUser.aspx.cs

## sp\_usernameCheckExists

### Goal

Add an admin and the premissions.

### Source Code

CREATE PROCEDURE [dbo].[sp\_usernameCheckExists]

@Username varchar(50)

AS

BEGIN

SET NOCOUNT ON;

SELECT TOP 1 UserId

FROM Users

WHERE (Username = @Username)

END

GO

### Files

## sp\_verifyEmail

### Goal

Get information settings

### Source Code

CREATE PROCEDURE [dbo].[sp\_verifyEmail]

@VCode nvarchar(MAX)

AS

BEGIN

SET NOCOUNT ON;

DECLARE @UserId int

DECLARE @Status smallint

SELECT @UserId =

(

SELECT TOP 1 (UserId)

FROM VerificationEmail

WHERE (VerificationCode = @VCode)

)

IF (@UserId is null)

BEGIN

SELECT @Status = 2

END

ELSE

BEGIN

UPDATE Users

SET EmailVerified = 1

WHERE (UserId = @UserId)

DELETE

FROM VerificationEmail

WHERE (VerificationCode = @VCode)

SELECT @Status = 1

END

SELECT (@Status) AS VerificationStatus

END

GO

### Files

* Verify.aspx.cs

# Classes

# Errors

~/Error/EventNotFound

EventView.aspx.cs

EventsModify.aspx.cs

~/Error/404 (No admin)

Admin.aspx.cs

~/Error/UserNotEventOwner

EventsModify.aspx.cs

~/Error/PRequestNotFound

ForgotPassword.aspx.cs

~/Error/NoProfile

Profile.aspx.cs

~/Error/RedistrictedProfile

Profile.aspx.cs

~/Error/NoProfileForSettings

Settings.aspx.cs

# Notification Codes

Code: 1

Name: Registered successfully

Trigger: After registering

Text: Congratulations! You have successfully registered!

Link: User’s profile

Code: 2

Name: Accepted in an event

Trigger: Owner of the event accepts user’s request

Text: Guess what?! You have been chosen to participate in this event: [EventName]

Link: Event’s page

Code: 3

Name: Event is full

Trigger: All the participants’ spots are filled

Text: Your event is full now! Event: [EventName]

Link: Event’s page

Code: 4

Name: User got followed

Trigger: A user followed this user

Text: [User full name] started following you.

Link: Follower profile

[FUTURE]

Code: 5

Name: Invited person registered

Trigger: Invited person registered

Text:

Link: Friend’s profile

Code: 6

Name: Become an admin

Trigger: User becomes an admin

Text: You are now an admin!

# Automatic Tasks

Deleting hashtags after their lifetime. (1 hour)

Statistics (Daily)

Delete ForgotPassword records after 2 days (Daily)

Delete Expired LoginSessions (1 hour)

Delete Notifications older than 30 days (Daily)

Delete Review Requests after 30 days (Daily)

# Folders

## AdminPages

### Contain

Admin pages user controls

### Files

* Admins.ascx
* Blog.ascx
* Content.ascx
* Events.ascx
* Locations.ascx
* Newsletter.ascx
* Settings.ascx
* Statistics.ascx
* Support.ascx
* Users.ascx

## Classes

### Contain

Project classes

### Files

* AdminPermissions.cs
* LoginSession.cs
* Notifications.cs
* SiteSettings.cs
* UserInfo.cs

## EmailTemplates

### Contain

Templates of the emails which application sends to the users

### Files

* ForgotPassword.txt
* Register.txt

## Files

### Contain

Files that application store from users

### Files



## Images

### Contain

Image files which application uses

### Files



## JS

### Contain

JavaScript files

### Files



## Styles

### Contain

CSS files

### Files



# Pages

# Routing