U.PORTO

FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

Modelo formal do StackOverflow em VDM++

Mestrado Integrado em Engenharia Informática e Computação

Métodos Formais em Engenharia de Software

Grupo:
Mário Fernandes - up201201705
Tiago Filipe - up201610655

03 de Janeiro de 2018

Índice

1. Descrição do sistema informal e lista de requisitos

1.1 Descrição do sistema informal

O sistema consiste na página web do StackOverflow que é um portal que apresenta perguntas e respostas em uma grande quantidade de tópicos de programação de computadores. Nesta plataforma qualquer utilizador pode perguntar ou responder a uma pergunta.

Para além disso, existe também um sistema de login e de reputação, em que cada utilizador pode aumentar ou diminuir a importância de uma pergunta ou de uma resposta.

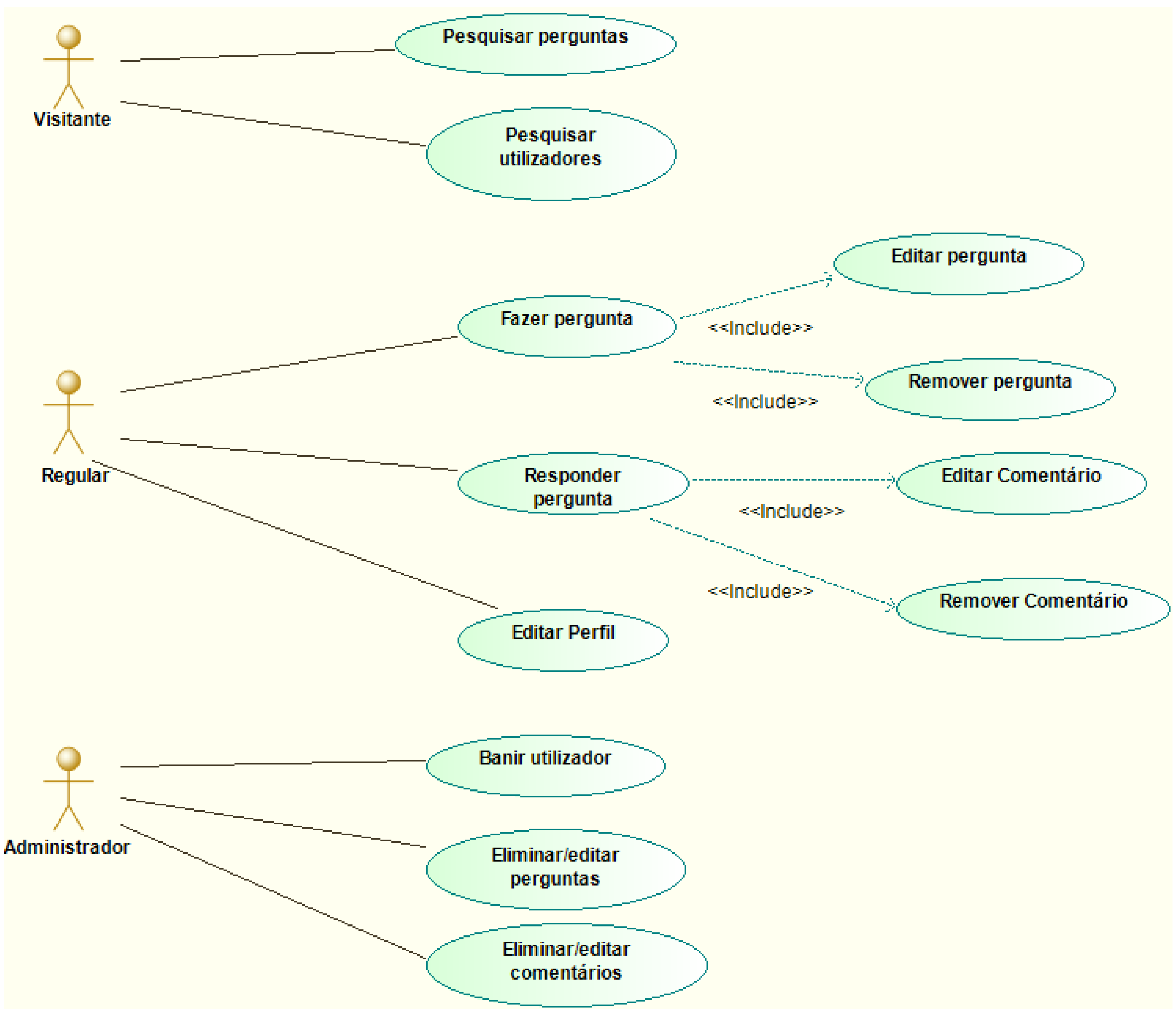
1.2 Lista de requisitos

Id	Prioridade	Descrição
R1	Obrigatório	Um utilizador pode registar-te.
R2	Obrigatório	Um utilizador registado pode escrever uma pergunta.
R3	Obrigatório	Um utilizador registado pode escrever uma resposta a uma pergunta.
R4	Obrigatório	Um utilizador pode pesquisar perguntas que tenham sido colocadas.
R5	Obrigatório	Um utilizador pode pesquisar utilizadores que estejam registados.
R6	Obrigatório	Um administrador pode banir um utilizador.
R7	Obrigatório	Um administrador pode remover uma pergunta ou um comentário.
R8	Obrigatório	Um utilizador registado pode editar ou remover uma pergunta ou comentário que ele tenha feito.
R9	Obrigatório	Um utilizador registado pode editar as informações do perfil.

Os requisitos descritos acima, são transformados em casos de uso e descritos com mais detalhe no capítulo seguinte.

2. Modelo Visual UML

2.1 Modelo de Casos de Uso



Relativamente ao diagrama de casos de uso apresentado em cima, um utilizador Administrador executa todos os casos de uso e um utilizador Regular executa os seus casos de uso e os de um Visitante.

De seguida serão descritos os principais casos de uso:

Cenário	**Registar**
Descrição	Cenário normal para um utilizador se registar no site.
Pré-condições	1. O utilizador não pode estar logado no sistema.

Pós-condições	1. O utilizador fica guardado no sistema.
Procedimento	1. O utilizador insere os dados para registar (nome, email, idade, género). 2. O sistema valida os dados, associa a data actual ao registo e guarda no sistema.
Exceções	1. O utilizador insere dados inválidos (volta para o passo 1).

Cenário	**Fazer pergunta**
Descrição	Cenário normal para um utilizador colocar uma pergunta no site.
Pré-condições	1. O utilizador estar registado no sistema.
Pós-condições	1. A pergunta é guardada no sistema. 2. A pergunta fica associada ao utilizador que a colocou.
Procedimento	1. O utilizador insere o assunto da pergunta, a pergunta e palavras-chave (tags) da pergunta. 2. O sistema valida os dados e guarda a pergunta no sistema.
Exceções	1. O utilizador insere dados inválidos (volta para o passo 1). 2. O utilizador pretende alterar a sua pergunta (volta para o passo 1). 3. O utilizador pretende remover a sua pergunta. O sistema elimina a pergunta. O caso de uso termina.

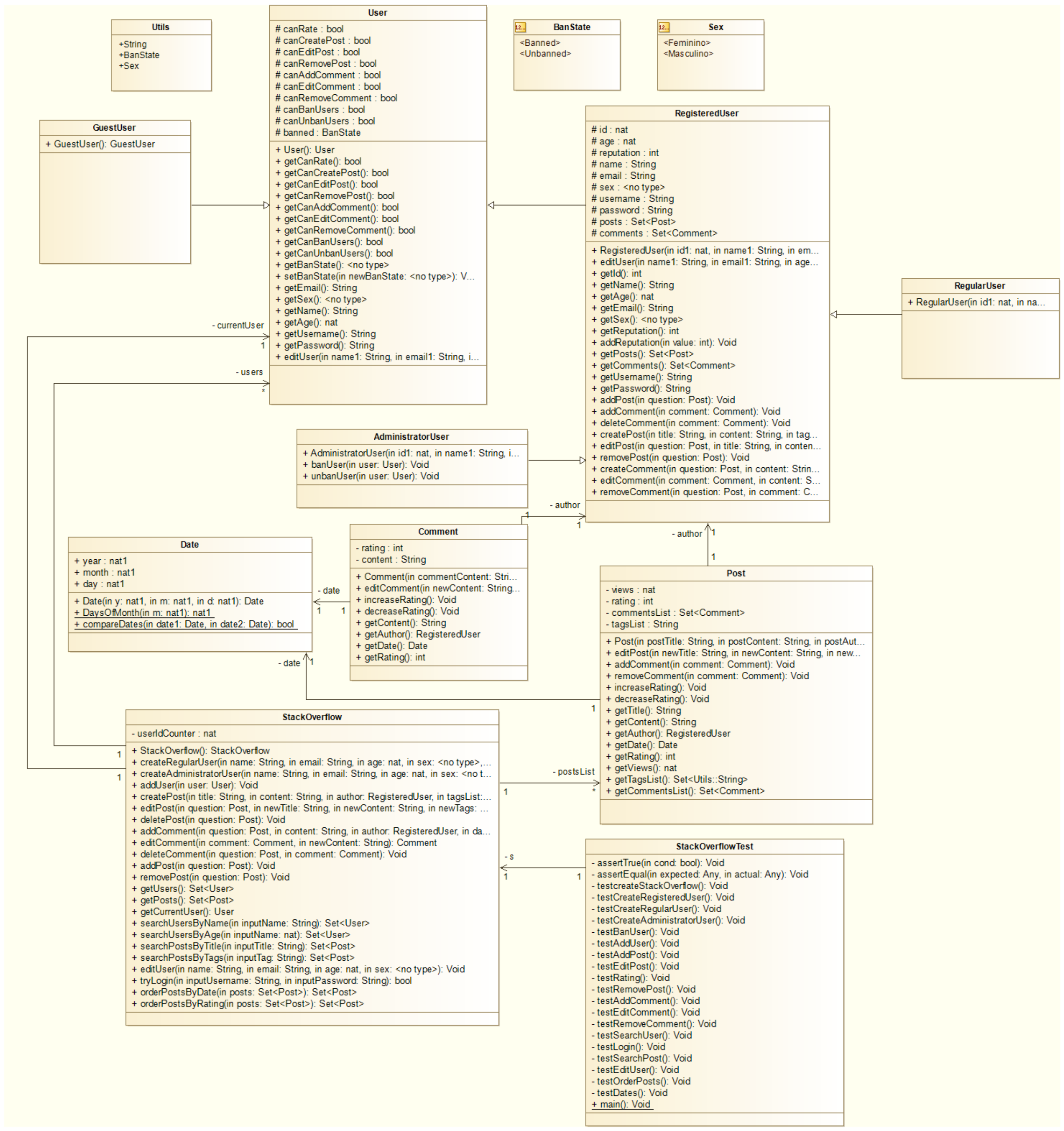
Cenário	**Responder a pergunta**
Descrição	Cenário normal para um utilizador responder a uma pergunta colocada no site.
Pré-condições	1. O utilizador estar registado no sistema.
Pós-condições	1. A resposta é associada à pergunta. 2. A resposta é associada ao utilizador que a fez.

Procedimento	1. O utilizador selecciona uma pergunta. 2. O utilizador insere o texto da resposta. 3. O sistema valida o input e guarda a resposta.
Exceções	1. O utilizador insere um texto vazio (volta para o passo 1). 2. O utilizador pretende alterar a sua resposta (volta para o passo 2). 3. O utilizador pretende remover o comentário escrito. O sistema elimina o comentário. O caso de uso termina.

Cenário	**Pesquisar perguntas**
Descrição	Cenário normal para um utilizador pesquisar perguntas colocadas no site.
Pré-condições	1. Existirem perguntas guardadas no sistema.
Pós-condições	1. Mostra lista com as perguntas encontradas ordenadas por rating ou data.
Procedimento	1. O utilizador insere um parâmetro de pesquisa (título ou palavra-chave) 2. O sistema procura os posts que enquadram com esse parâmetro de pesquisa e mostra o resultado ordenado por rating ou data.
Exceções	(nenhuma)

Cenário	**Pesquisar utilizadores**
Descrição	Cenário normal para um utilizador procurar outros utilizadores.
Pré-condições	1. Existirem utilizadores guardados no sistema.
Pós-condições	1. Mostra lista com os utilizadores encontrados.
Procedimento	1. O utilizador insere um parâmetro de pesquisa (nome ou idade). 2. O sistema procura os utilizadores que enquadram com esse parâmetro de pesquisa e mostra o resultado.
Exceções	(nenhuma)

2.2 Modelo de Classes



Classe	Descrição
User	Define as permissões de um utilizador.
GuestUser	Subclasse de User que define um visitante que não está registado num site.
RegisteredUser	Subclasse de User que define um utilizador registado através dos seus dados pessoais e de autenticação.

RegularUser	Subclasse de RegisteredUser que define um utilizador com privilégios mínimos.
AdministratorUser	Subclasse de RegisteredUser que define um utilizador com todos os privilégios.
Post	Define uma pergunta colocada por um utilizador.
Comment	Define uma resposta escrita por um utilizador a uma pergunta.
Date	Define uma data e tem funções para trabalhar com essas datas.
Utils	Define types específicos deste projecto.
StackOverflow	Define as operações disponíveis para os utilizadores.
StackOverflowTest	Define cenários de teste para o site StackOverflow e testa os mesmos.

3. Modelo Formal VDM++

3.1 Classe AdministratorUser

```
class AdministratorUser is subclass of RegisteredUser
  /*
    Defines an authenticated user with maximum privileges
  */

operations
        public AdministratorUser : nat * Utils‘String * Utils‘String * nat * Utils‘Sex * int *
            Utils‘ String * Utils‘String ==> AdministratorUser
    AdministratorUser(id1, name1, email1, age1, sex1, reputation1, username1, password1)
    == (
    canRate := true;
    canCreatePost := true;
    canEditPost := true;
    canRemovePost := true;
    canAddComment := true;
    canEditComment := true;
    canRemoveComment := true;
    canBanUsers := true;
    canUnbanUsers := true;

    RegisteredUser(id1, name1, email1, age1, sex1, reputation1, username1, password1);
   )
    post canRate = true and canCreatePost = true and canEditPost = true and
      canRemovePost = true and canAddComment = true and canEditComment = true and
      canRemoveComment = true and canBanUsers = true and canUnbanUsers = true and
      banned = <Unbanned> and name = name1 and email = email1 and age = age1 and
```

```
        sex = sex1 and reputation = reputation1 and username = username1 and password =
        password1;

   -- Change the ban state of a user to Banned
   public banUser: User ==> ()


   banUser(user) == (
   user.setBanState(
   <Banned>);
   ) pre canBanUsers = true;


   -- Change the ban state of a user
   to Unbanned public unbanUser:
   User ==> () unbanUser(user) == (
   user.setBanState(<Unbanned>);
    )pre canUnbanUsers = true;

end AdministratorUser
```

3.2 Classe Comment

```
class Comment
   /*
    Defines the details of the comment using types from Utils class and contains
    operations to work with comments.
   */

instance variables
    private content : Utils‘String;
    private author : RegisteredUser;
    private date : Date;
    private rating : int := 0;

operations
     public Comment: Utils‘String * RegisteredUser * nat1 * nat1 * nat1 ==> Comment
      Comment(commentContent, commentAuthor, day, month, year) ==
      ( content := commentContent;
        author := commentAuthor;

      date := new Date(year, month, day);
     )
     post content = commentContent and author = commentAuthor and rating = 0 and
     date.year = year and date.month = month and date.day = day;

     public editComment: Utils‘String ==>
      Comment
      editComment(newContent) ==
     ( content := newContent;
      return self;
     )
     post content = newContent;

      public increaseRating:
        () ==> ()
        increaseRating() ==

      ( rating := rating + 1;
      author.addReputation(3)
     );

      public
        decreaseRating: ()
        ==> ()
        decreaseRating() ==

      ( rating := rating - 1;
      author.addReputation(-2)
     );


    /* Get Operations */
    public getContent: () ==> Utils‘String
getContent() ==

      ( return content;
      )
        pre content <> []
        post RESULT = content;

     public getAuthor: () ==> RegisteredUser


      getAuthor() ==
      ( return author;
      )
        post RESULT = author;

     public getDate: () ==> Date


      getDate() ==
      ( return date;
      )
        post RESULT = date;

     public getRating: () ==> int
      getRating() ==
      ( return rating;
      )
```

```
        post RESULT = rating;

end Comment
```

3.3 Classe Date

```
class Date
  /*
    Defines a Date and contains operations and functions to work with the Date.
  */

instance variables
    public year : nat1;
    public month : nat1;
    public day : nat1;

operations
    public Date:nat1 * nat1 * nat1 ==> Date
    Date(y,m,d) ==
    (year := y; month := m; day := d;
    )pre y > 1900 and m <= 12 and d <= DaysOfMonth(m);


    -- Operation that receives the month and returns the last day of that month.
     public static pure DaysOfMonth: nat1 ==> nat1
     DaysOfMonth(m) ==
     ( if (m = 1 or m = 3 or m = 5 or m = 7 or m = 8 or m = 10 or m = 12) then ( return 31;
     ) else if (m = 2) then ( return 28;

     ) else ( return 30;
     )
    );

functions
    -- Function that check if one date is more recent than the other date
    public static compareDates: Date * Date -> bool
    compareDates(date1, date2) ==
    if(date1.year > date2.year) then true
    elseif (date1.year = date2.year and date1.month > date2.month) then true
        elseif (date1.year = date2.year and date1.month = date2.month and date1.day >
            date2.day) then true
    else false;

end Date
```

3.4 Classe GuestUser

```
class GuestUser is subclass of User
  /*
    Defines a person who is visiting the site without being authenticated.
  */

operations
     public GuestUser:()
     ==> GuestUser
     GuestUser() == (
     User();
     ) post canRate = false and canCreatePost = false and canEditPost = false and
       canRemovePost = false and canAddComment = false and canEditComment = false
       and canRemoveComment = false and canBanUsers = false and canUnbanUsers =
       false and banned = <Unbanned>;

end GuestUser
```

3.5 Classe Post

```
class Post
  /*
   Defines the details of the post using types from Utils class and contains operations to
   work with posts.
  */

instance variables
    private title : Utils‘String;
    private content : Utils‘String;
    private author : RegisteredUser;
    private date : Date;
    private views : nat := 0;
    private rating : int := 0;
    private tagsList : set of Utils‘String;
    private commentsList : set of Comment := {};

operations
    public Post: Utils‘String * Utils‘String * RegisteredUser * set of Utils‘String * nat1 * nat1 *
         nat1 ==> Post
     Post(postTitle, postContent, postAuthor, postTags, day, month, year) ==
     ( title := postTitle;
      content := postContent;
      author := postAuthor;
      tagsList := postTags;
      date := new Date(year, month, day);
      return self;
     )
         post title = postTitle and content = postContent and author = postAuthor and
              tagsList = postTags and views = 0 and rating = 0 and commentsList = {}
              and date.year = year and date.month = month and date.day = day;

     public editPost: Utils‘String * Utils‘String * set of Utils‘String ==>
      Post editPost(newTitle, newContent, newTags) ==
      ( title := newTitle;
      content := newContent;
      tagsList := newTags;
      return self;
     )
     post content = newContent and title = newTitle and tagsList = newTags;

     public addComment: Comment ==> ()
      addComment(comment) ==
      ( commentsList := commentsList union {comment};);

      public removeComment: Comment ==> ()
       removeComment(comment) ==

      ( commentsList := commentsList \ {comment};
     )
     pre comment in set commentsList;

      public increaseRating: () ==> ()
       increaseRating() ==
      ( rating := rating + 1;
      author.addReputation(1)
     );

      public decreaseRating: () ==> ()
       decreaseRating() ==
      ( rating := rating - 1;
      author.addReputation(-1)
     );


    /* Get Operations */
     public getTitle: () ==> Utils‘String
      getTitle() ==
      ( return title;
      )
       pre title <> []
       post RESULT = title;

      public getContent: () ==> Utils‘String
       getContent() ==
      ( return content;
      )
       pre content <> []
       post RESULT = content;

     public getAuthor: () ==> RegisteredUser
      getAuthor() ==
      ( return author;
      )
       post RESULT = author;
```

```
    public getDate: () ==> Date
     getDate() ==
     ( return date;
     )
       post RESULT = date;

    public getRating: () ==> int
     getRating() ==
     ( return rating;
     )
       post RESULT = rating;

    public getViews: () ==> nat
     getViews() ==
     ( return views;
     )
     pre (views >= 0)
       post RESULT = views;

     public getTagsList: () ==> set of Utils‘String
       getTagsList() ==
     ( return tagsList;
     )
       pre tagsList <> {}
       post RESULT = tagsList;

     public getCommentsList: () ==> set of Comment
       getCommentsList() ==
     ( return commentsList;
     )
       post RESULT = commentsList;

end Post
```

3.6 Classe RegisteredUser

```
class RegisteredUser is subclass of User
   /*
    Defines an authenticated user.
   */

    instance variables
      protected id : nat;
    protected name : Utils‘String;
    protected age : nat;
    protected email : Utils‘String;
    protected sex : Utils‘Sex;
    protected reputation : int;
    protected posts : set of Post := {};
    protected comments : set of Comment := {};
    protected username : Utils‘String;
    protected password : Utils‘String;

operations
     public RegisteredUser: nat * Utils‘String * Utils‘String * nat * Utils‘Sex * int * Utils‘String
            * Utils‘String ==> RegisteredUser
     RegisteredUser(id1,name1,email1,age1,sex1,reputation1,username1,password1) ==
            (id := id1;
     name := name1;
     email := email1;
     age := age1;
     sex := sex1;
     reputation := reputation1;
     username := username1;
     password := password1;
     return self;
     )
    post name = name1 and email = email1 and age = age1 and sex = sex1 and reputation
    = reputation1 and username = username1 and password = password1;

     public editUser: Utils‘String * Utils‘String * nat * Utils‘Sex ==> RegisteredUser
     editUser(name1,email1,age1,sex1) ==
      (name := name1;
       email := email1;
       age := age1;
      sex := sex1;
      return self;)
     pre name1 <> "" and email1 <> "" and age1 > 0 and (sex1 = <Masculino> or sex1 =
     <Feminino>)
     post name = name1 and email = email1 and age = age1 and sex = sex1;


    public addReputation: int ==> ()
     addReputation(value) == (reputation := reputation + value;)
     pre value <> 0;


    public addPost: Post ==> ()
    addPost(question) == (posts := posts union {question};);

    public addComment: Comment ==> ()
    addComment(comment) == (comments := comments union {comment};);

    public deleteComment:Comment ==> ()
    deleteComment(comment) == (comments := comments \ {comment};);
```

```
public createPost: Utils‘String * Utils‘String * set of Utils‘String * nat1 * nat1 * nat1 ==>
    Post
 createPost(title,content,tagsList,day,month,year) ==
 (dcl newPost:Post := new Post(title, content, self, tagsList, day, month, year);
 addPost(newPost);
 return newPost;)
pre canCreatePost = true;

public editPost:Post * Utils‘String * Utils‘String * set of Utils‘String ==> Post
editPost(question,title,content,tagsList) ==
(return question.editPost(title, content, tagsList);)
pre canEditPost = true and question in set posts;

public removePost:Post ==> ()
removePost(question) ==
(posts := posts \ {question};)
pre question in set posts;

public createComment:Post * Utils‘String * nat1 * nat1 * nat1 ==> Comment
createComment(question,content,day,month,year) ==
 (dcl newComment:Comment := new Comment(content, self, day, month, year);
  question.addComment(newComment);
 addComment(newComment);
 return newComment;)
pre canAddComment = true;

public editComment:Comment * Utils‘String ==> Comment
editComment(comment,content) == (return comment.editComment(content); )
pre canEditComment = true and comment in set comments;

public removeComment:Post * Comment ==> ()
removeComment(question,comment) ==
(question.removeComment(comment); deleteComment(comment);)
pre comment in set comments;

/* Get Operations */

public getId: () ==> int
 getId() == (return id;)
 pre (id > 0)
 post RESULT = id;

 public getName: () ==>
  Utils‘String getName() ==
  (return name;)
 post RESULT = name;
```

```
        public getAge: () ==> nat
          getAge() == (return age;)
          pre (age > 1)
        post RESULT = age;


        public getEmail: () ==> Utils‘String
          getEmail() == (return email;)
        post RESULT = email;

      public getSex: () ==> Utils‘Sex
        getSex() == (return sex;)
        pre sex = <Masculino> or sex = <Feminino>
        post RESULT = sex;

        public getReputation: () ==> int getReputation()
          == (return reputation;)
        post RESULT = reputation;


        public getPosts: () ==> set of Post
          getPosts() == (return posts;)
        post RESULT = posts;


        public getComments: () ==> set of Comment
          getComments() == (return comments;)
        post RESULT = comments;

        public getUsername: () ==> Utils‘String
          getUsername() == (return username;)
        post RESULT = username;

        public getPassword: () ==> Utils‘String
          getPassword() == (return password;)
        post RESULT = password;

end RegisteredUser
```

3.7 Classe RegularUser

```
class RegularUser is subclass of RegisteredUser
   /*
     Defines an authenticated user with minimum privileges.
   */

operations
            public RegularUser:nat * Utils‘String * Utils‘String * nat * Utils‘Sex * int * Utils‘String
                  * Utils‘String ==> RegularUser
        RegularUser(id1,name1,email1,age1,sex1,reputation1,username1,password1) == (
        canRate := true;
        canCreatePost := true;
        canEditPost := true;
        canAddComment := true;
        canEditComment := true;
        RegisteredUser(id1, name1, email1, age1, sex1, reputation1, username1, password1);
     )
        post canRate = true and canCreatePost = true and canEditPost = true and
         canRemovePost = false and canAddComment = true and canEditComment = true
         and canRemoveComment = false and canBanUsers = false and canUnbanUsers =
         false and banned = <Unbanned> and name = name1 and email = email1 and age =
         age1 and sex = sex1 and reputation = reputation1 and username = username1 and
         password = password1;

end RegularUser
```

3.8 Classe StackOverflow

```
class StackOverflow
  /*
    Contains the core model of the StackOverflow.
    Defines the operations available to the users.
  */

instance variables
    private userIdCounter : nat := 1;
    private users : set of User;
    private postsList : set of Post;
    private currentUser : User;

operations
    public StackOverflow: () ==> StackOverflow StackOverflow() ==
    (currentUser := new GuestUser();
     users := {};
     postsList := {};
     return self;
     )
    post users = {} and userIdCounter = 1 and postsList = {};


       public createRegularUser: Utils‘String * Utils‘String * nat * Utils‘Sex * nat *
           Utils‘String * Utils‘String ==> RegularUser
    createRegularUser(name, email, age, sex, reputation, username, password) ==
    (return new RegularUser(userIdCounter, name, email, age, sex, reputation, username,
    password););


   public createAdministratorUser: Utils‘String * Utils‘String * nat * Utils‘Sex * nat * Utils‘
       String * Utils‘String ==> AdministratorUser
    createAdministratorUser(name, email, age, sex, reputation, username, password) ==
    (return new AdministratorUser(userIdCounter, name, email, age, sex, reputation,
        username, password););

 public addUser: User ==> ()
 addUser(user) ==
  (users := {user} union users;
   userIdCounter := userIdCounter + 1;
  )
 pre (user not in set users)
 post userIdCounter > 1 and user in set users;

public editUser:Utils‘String * Utils‘String * nat * Utils‘Sex ==> ()
editUser(name,email,age,sex) ==
(currentUser := currentUser.editUser(name,email,age,sex);)
pre name <> "" and email <> "" and age > 1 and sex <> nil;

public createPost: Utils‘String * Utils‘String * RegisteredUser * set of Utils‘String * nat1
*nat1 * nat1 ==> Post
 createPost(title, content, author, tagsList, day, month, year) ==
  (dcl newPost:Post := author.createPost(title, content, tagsList, day, month, year);
   addPost(newPost);
  return newPost;);

public editPost: Post * Utils‘String * Utils‘String * set of Utils‘String ==> Post
 editPost(question, newTitle, newContent,
 newTags) == (dcl author:RegisteredUser :=
 question.getAuthor();
  return author.editPost(question, newTitle, newContent, newTags););


 public deletePost:Post ==> ()
  deletePost(question) ==
  (dcl author:RegisteredUser := question.getAuthor(); author.removePost(question);
   removePost(question););


 public addComment:Post * Utils‘String * RegisteredUser * nat1 * nat1 * nat1 ==>
  Comment addComment(question, content, author, day, month, year) ==
 (return author.createComment(question, content, day, month, year););


 public editComment: Comment * Utils‘String ==> Comment
  editComment(comment, newContent) ==
 (dcl author:RegisteredUser := comment.getAuthor();
  return author.editComment(comment, newContent););

public deleteComment:Post * Comment ==> ()
 deleteComment(question, comment) ==
  (dcl author:RegisteredUser := comment.getAuthor();
   author.removeComment(question, comment););

 public addPost: Post ==> ()
 addPost(question) ==
 (postsList := {question} union postsList;)
 pre (question not in set postsList)
 post question in set postsList;


 public removePost: Post ==> ()
 removePost(question) ==
 (postsList := postsList \ {question};)
 pre (question in set postsList)
 post question not in set postsList;

public getUsers: () ==> set of User
getUsers() == (return users;)
post RESULT = users;

public getPosts: () ==> set of Post
getPosts() == (return postsList;) post
RESULT = postsList;

public getCurrentUser: () ==> User
getCurrentUser() == (return currentUser;)
post RESULT = currentUser;

-- Receives a word and search the user’s name by that word.
public searchUsersByName:Utils‘String ==> set of User
searchUsersByName(inputName) ==
(dcl usersTemp : set of User := {};
for all currUser in set users do
(if currUser.getName() = inputName then
  usersTemp := {currUser} union usersTemp;);
return usersTemp;)
pre inputName <> "";


-- Search all the users from a certain age.
 public searchUsersByAge:nat ==> set of User
  searchUsersByAge(inputName) ==
 (dcl usersTemp : set of User := {};
for all currUser in set users do
(if currUser.getAge() = inputName then
  usersTemp := {currUser} union usersTemp;);
return usersTemp;)
pre inputName > 0;


-- Receives a word and search the posts’s title which contains that word.
 public searchPostsByTitle:Utils‘String ==> set of Post
  searchPostsByTitle(inputTitle) ==
    (dcl postsTemp : set of Post := {};
    dcl inputSize : int := len inputTitle;
    dcl counter : nat := 1;
    dcl counter2 : nat := 1;
    dcl flagFound : bool := false;

    for all currPost in set postsList do

    (flagFound := false;
    counter2 := 1;
      while (counter < len currPost.getTitle() ) do
      (if(currPost.getTitle()(counter) = inputTitle(counter2) and flagFound = false) then
       (if(counter2 = inputSize) then
         (postsTemp := postsTemp union {currPost};
         flagFound := true;)
         else (counter2 := counter2 + 1;);)
       else
       (counter2 := 1;);
       counter := counter + 1;
       );
       );
  return postsTemp;)

  pre inputTitle <> "";


-- Receives a tag and search posts containing that tag.
public searchPostsByTags:Utils‘String ==> set of Post
searchPostsByTags(inputTag) ==
  (dcl postsTemp : set of Post := {};

  for all currPost in set postsList do (
  let tagList = currPost.getTagsList() in (
  if(inputTag in set tagList) then
  postsTemp := postsTemp union {currPost}; ));
  return postsTemp;)
  pre inputTag <> "";


-- Verifies if the login received exists in the system
public tryLogin:Utils‘String * Utils‘String ==> bool
  tryLogin(inputUsername,inputPassword) == (
  for all currUser in set users do
  (if currUser.getUsername() = inputUsername and currUser.getPassword() =
  inputPassword then(
  currentUser := currUser;
  return true);
  );

  return false;)
pre inputUsername <> "" and inputPassword <> "";
```

```
    public orderPostsByDate:set of Post ==> set of Post
    orderPostsByDate(posts) ==
    ( dcl tempPosts : set of Post := {};
       dcl currentDate : Date; dcl
       tempDate : Date; dcl
       tempPost : Post;
       for all currPost in set posts do
           (currentDate := currPost.getDate();
               tempDate := currentDate;
               tempPost := currPost;
           for all currPost2 in set posts do
           (if(currPost2 not in set tempPosts) then
               ( if(Date`compareDates(currPost2.getDate(),currentDate) and
                   Date`compareDates(currPost2. getDate(), tempDate)) then
           (tempPost := currPost2;
           tempDate := currPost2.getDate();
           );
           );
           tempPosts := tempPosts union {tempPost};
           );
           );
           return tempPosts;)
       pre (card posts > 0)
       post card posts = card RESULT;

       public orderPostsByRating: set of Post ==> set of Post
       orderPostsByRating(posts) ==
       (dcl tempPosts : set of Post := {};
       dcl currentRating : int;
       dcl tempRating : int;
       dcl tempPost : Post;
       for all currPost in set posts do
       (currentRating := currPost.getRating();
       tempRating := currentRating;
       tempPost := currPost;
           for all currPost2 in set posts do
           (if(currPost2 not in set tempPosts) then
           (if(currPost2.getRating() > currentRating and currPost2.getRating() > tempRating)
           then
           (tempPost := currPost2;
           tempRating := currPost2.getRating();
           );
           );
           tempPosts := tempPosts union {tempPost};
           ););
       return tempPosts;)
       pre (card posts > 0)
       post card posts = card RESULT;

end StackOverflow
```

4. Validação do Modelo

4.1 Classe StackOverflowTest

```
class StackOverflowTest
  /*
  Contains the test cases for the StackOverFlow
  */


instance variables
      s  : StackOverflow := new StackOverflow();

operations
    private assertTrue: bool ==> ()
    assertTrue(cond) == return
    pre cond;

    private assertEqual: ? * ? ==> ()
    assertEqual(expected, actual) ==
         if expected <> actual then (
             IO‘print("Actual value (");
             IO‘print(actual);
         IO‘print(") different from expected (");
         IO‘print(expected);
         IO‘println(")\n")
      )
    post expected = actual;

    private testcreateStackOverflow: () ==> ()
    testcreateStackOverflow() == (
     dcl guest : GuestUser := s.getCurrentUser();
      assertTrue(isofclass(GuestUser,guest));
      assertTrue(guest.getName() = "");
      assertTrue(guest.getEmail() = "");
      assertTrue(guest.getAge() = 1);
      assertTrue(guest.getSex() = <Masculino>);
      assertTrue(guest.getUsername() = "");
      assertTrue(guest.getPassword() = "");

     guest := guest.editUser("Mario","mario.gustavo@hotmail.com", 23, <Masculino>);
     assertTrue(guest.getName() = "");
     assertTrue(guest.getEmail() = "");
     assertTrue(guest.getAge() = 1);
     assertTrue(guest.getSex() = <Masculino>);
     assertTrue(guest.getUsername() = "");
     assertTrue(guest.getPassword() = "");
    );
```

```
private testCreateRegisteredUser: () ==> ()
testCreateRegisteredUser() == (
     dcl newUser:RegisteredUser := new RegisteredUser(1,"Mario",
          "mario.gustavo@hotmail.com", 23, < Masculino>, 0, "mario","12345" );
 assertTrue(newUser.getCanRate() = false);
 assertTrue(newUser.getCanCreatePost() = false );
 assertTrue(newUser.getCanEditPost() = false );
 assertTrue(newUser.getCanRemovePost() = false );
 assertTrue(newUser.getCanAddComment() = false );
 assertTrue(newUser.getCanEditComment() = false );
 assertTrue(newUser.getCanRemoveComment() = false );
 assertTrue(newUser.getCanBanUsers() = false );
 assertTrue(newUser.getCanUnbanUsers() = false );
 assertTrue(newUser.getName() = "Mario");
 assertTrue(newUser.getEmail() = "mario.gustavo@hotmail.com");
 assertTrue(newUser.getAge() = 23);
  assertTrue(newUser.getSex() = <Masculino>);
 assertTrue(newUser.getReputation() = 0);
 assertTrue(newUser.getBanState() = <Unbanned>);
 assertTrue(newUser.getId() = 1);
 assertEqual(newUser.getPosts(), {}); assertTrue(newUser.getUsername() = "mario");
 assertTrue(newUser.getPassword() = "12345");
);

 private testCreateRegularUser: () ==> ()
testCreateRegularUser() ==
(
     dcl newUser:RegularUser := s.createRegularUser("Mario",
          "mario.gustavo@hotmail.com", 23, < Masculino>, 0, "mario","12345" );
 assertTrue(newUser.getCanRate() = true);
 assertTrue(newUser.getCanCreatePost() = true );
 assertTrue(newUser.getCanEditPost() = true );
 assertTrue(newUser.getCanRemovePost() = false );
 assertTrue(newUser.getCanAddComment() = true );
 assertTrue(newUser.getCanEditComment() = true );
 assertTrue(newUser.getCanRemoveComment() = false );
 assertTrue(newUser.getCanBanUsers() = false );
 assertTrue(newUser.getCanUnbanUsers() = false );
 assertTrue(newUser.getName() = "Mario");
 assertTrue(newUser.getEmail() = "mario.gustavo@hotmail.com");
 assertTrue(newUser.getAge() = 23);
 assertTrue(newUser.getSex() = <Masculino>);
 assertTrue(newUser.getReputation() = 0);
 assertTrue(newUser.getBanState() = <Unbanned>);
 assertTrue(newUser.getId() = 1);
 assertEqual(newUser.getPosts(), {});
 assertTrue(newUser.getUsername() = "mario");
 assertTrue(newUser.getPassword() = "12345");
);
```

```
  private
testCreateAdministratorUser: () ==> ()
testCreateAdministratorUser() == (
 dcl newUser:AdministratorUser := s.createAdministratorUser("Mario",
 "mario.gustavo@hotmail.com", 23, <Masculino>, 0, "mario","12345" );
 assertTrue(newUser.getCanRate() = true);
 assertTrue(newUser.getCanCreatePost() = true );
 assertTrue(newUser.getCanEditPost() = true );
 assertTrue(newUser.getCanRemovePost() = true );
 assertTrue(newUser.getCanAddComment() = true );
 assertTrue(newUser.getCanEditComment() = true );
 assertTrue(newUser.getCanRemoveComment() = true );
 assertTrue(newUser.getCanBanUsers() = true );
 assertTrue(newUser.getCanUnbanUsers() = true );
 assertTrue(newUser.getName() = "Mario");
 assertTrue(newUser.getEmail() = "mario.gustavo@hotmail.com");
 assertTrue(newUser.getAge() = 23);
 assertTrue(newUser.getSex() = <Masculino>);
 assertTrue(newUser.getReputation() = 0);
 assertTrue(newUser.getBanState() = <Unbanned>);
);

private testBanUser: () ==> ()
testBanUser() ==
(
 dcl newUser:AdministratorUser := s.createAdministratorUser("Mario",
 "mario.gustavo@hotmail.com", 23, <Masculino>, 0, "mario","12345" );
     dcl newUser2:RegularUser := s.createRegularUser("Mario",
         "mario.gustavo@hotmail.com", 23, < Masculino>, 0, "mario","12345" );
     dcl newUser3:AdministratorUser := s.createAdministratorUser("Mario",
         "mario.gustavo@hotmail.com ", 23, <Masculino>, 0, "mario","12345" );
 assertTrue(newUser2.getBanState() = <Unbanned>);
 assertTrue(newUser3.getBanState() = <Unbanned>);
 newUser.banUser(newUser2);
 newUser.banUser(newUser3);
 assertTrue(newUser2.getBanState() = <Banned>);
 assertTrue(newUser3.getBanState() = <Banned>);
 newUser.unbanUser(newUser2);
 newUser.unbanUser(newUser3);
 assertTrue(newUser2.getBanState() = <Unbanned>);
 assertTrue(newUser3.getBanState() = <Unbanned>);
);

private testAddUser: () ==> ()
testAddUser() ==
(
     dcl newUser:RegularUser := s.createRegularUser("Mario",
         "mario.gustavo@hotmail.com", 23, < Masculino>, 0, "mario","12345" );
 dcl usersLength:nat := card
 s.getUsers();
 s.addUser(newUser);
 assertTrue(usersLength + 1 = card
 s.getUsers()); assertTrue(newUser in set
 s.getUsers());
);

private testAddPost: () ==> ()
testAddPost() ==
(
     dcl user:RegularUser := new RegularUser(1, "User", "email@gmail.com", 47,
          <Feminino>, 0, "mario ","12345");
 dcl tagsList:set of Utils‘String := {"Tag1", "Tag2", "Tag3"};
 dcl postsLength:nat := card s.getPosts();
 dcl newPost:Post := s.createPost("Title", "Content", user, tagsList, 29, 12, 2017);
 assertTrue(postsLength + 1 = card s.getPosts());
 assertTrue(newPost in set s.getPosts());
 assertTrue(newPost in set user.getPosts());
 assertEqual(newPost.getAuthor(), user);
 assertEqual(newPost.getDate().day, 29);
 assertEqual(newPost.getDate().month, 12);
 assertEqual(newPost.getDate().year, 2017);
 assertEqual(newPost.getViews(), 0);
 assertEqual(newPost.getRating(), 0);
);

private testEditPost: () ==> () testEditPost() ==
(
     dcl user:RegularUser := new RegularUser(1, "User", "email@gmail.com", 47,
          <Feminino>, 0, "mario ","12345");
 dcl tagsList:set of Utils‘String := {"Tag1", "Tag2", "Tag3"};
 dcl newPost:Post := s.createPost("Title", "Content", user, tagsList, 29, 12, 2017);
 dcl title:Utils‘String := newPost.getTitle();
 dcl tags:set of Utils‘String := newPost.getTagsList();
 newPost := s.editPost(newPost, title, "New content", tags);
 assertTrue(newPost.getTitle() = "Title");
 assertTrue(newPost.getContent() = "New content");
 assertEqual(newPost.getAuthor(), user);
 assertEqual(newPost.getTagsList(), {"Tag1", "Tag2", "Tag3"});
 assertEqual(newPost.getDate().day, 29);
 assertEqual(newPost.getDate().month, 12);
 assertEqual(newPost.getDate().year, 2017);
);

private testRating: () ==> () testRating() ==
(
     dcl user:RegularUser := new RegularUser(1, "User", "email@gmail.com", 47,
          <Feminino>, 0, "mario ","12345");
 dcl tagsList:set of Utils‘String := {"Tag1", "Tag2", "Tag3"};
 dcl newPost:Post := s.createPost("Title", "Content", user, tagsList, 29, 12, 2017);
 dcl newComment:Comment := s.addComment(newPost, "Content", user, 29,12,2017);
 assertEqual(newPost.getRating(), 0);
 assertEqual(newComment.getRating(), 0);
 assertEqual(user.getReputation(),0);
 newPost.increaseRating();
 newComment.increaseRating();
 assertEqual(newPost.getRating(),1);
 assertEqual(newComment.getRating(), 1);
 assertEqual(user.getReputation(),4);
 newPost.decreaseRating();
 newPost.decreaseRating();
 newComment.decreaseRating();
 newComment.decreaseRating();
 assertEqual(newPost.getRating(),-1);
 assertEqual(newComment.getRating(), -1);
 assertEqual(user.getReputation(),-2);
);

private testRemovePost: () ==> ()
testRemovePost() ==
(
     dcl user:RegularUser := new RegularUser(1, "User", "email@gmail.com", 47,
          <Feminino>, 0, "mario ","12345");
 dcl tagsList:set of Utils‘String := {"Tag1", "Tag2", "Tag3"};
 dcl newPost:Post := s.createPost("Title", "Content", user, tagsList, 29, 12, 2017);
 dcl postsLength:nat := card s.getPosts(); s.deletePost(newPost);
 assertTrue(postsLength - 1 = card s.getPosts());
 assertTrue(newPost not in set s.getPosts());
 assertTrue(newPost not in set newPost.getAuthor().getPosts());
);

private testAddComment: () ==> () testAddComment() ==
(
     dcl user:RegularUser := new RegularUser(1, "User", "email@gmail.com", 47,
          <Feminino>, 0, "mario ","12345");
 dcl tagsList:set of Utils‘String := {"Tag1", "Tag2", "Tag3"};
 dcl newPost:Post := s.createPost("Title", "Content", user, tagsList, 29, 12, 2017);
 dcl newComment:Comment := s.addComment(newPost, "Content", user, 29,12,2017);
 assertTrue(newComment in set newPost.getCommentsList());
 assertTrue(newComment in set user.getComments());
 assertEqual(newComment.getContent(), "Content");
 assertEqual(newComment.getDate().day, 29);
 assertEqual(newComment.getDate().month, 12);
 assertEqual(newComment.getDate().year, 2017);
);

private testEditComment: () ==> ()
testEditComment() ==
(
     dcl user:RegularUser := new RegularUser(1, "User", "email@gmail.com", 47,
          <Feminino>, 0, "mario ","12345");
 dcl tagsList:set of Utils‘String := {"Tag1", "Tag2", "Tag3"};
 dcl newPost:Post := s.createPost("Title", "Content", user, tagsList, 29, 12, 2017);
```

```
  dcl newComment:Comment := s.addComment(newPost, "Content", user, 29,12,2017);

  newComment := s.editComment(newComment, "New Content");
  assertTrue(newComment in set newPost.getCommentsList());
  assertTrue(newComment in set user.getComments());
  assertEqual(newComment.getContent(), "New Content");
  assertEqual(newComment.getDate().day, 29);
  assertEqual(newComment.getDate().month, 12);
  assertEqual(newComment.getDate().year, 2017);
);

private testRemoveComment: () ==> ()
testRemoveComment() ==
(
     dcl user:RegularUser := new RegularUser(1, "User", "email@gmail.com", 47,
          <Feminino>, 0, "mario ","12345");
  dcl tagsList:set of Utils`String := {"Tag1", "Tag2", "Tag3"};
  dcl newPost:Post := s.createPost("Title", "Content", user, tagsList, 29, 12, 2017);
  dcl newComment:Comment := s.addComment(newPost, "Content", user, 29,12,2017);
  s.deleteComment(newPost, newComment);
  assertTrue(newComment not in set newPost.getCommentsList());
  assertTrue(newComment not in set user.getComments());
);
  private testSearchUser: () ==> () testSearchUser() ==
(
     dcl newUser:RegularUser := s.createRegularUser("Rafael",
          "mario.gustavo@hotmail.com", 27, < Masculino>, 0, "mario","12345" );
  s.addUser(newUser);
  assertEqual(s.searchUsersByName("Rafael"), {newUser});
  assertEqual(s.searchUsersByAge(27), {newUser});
);

   private testLogin: () ==> ()
   testLogin() ==
(
     dcl newUser:RegularUser := s.createRegularUser("Jorge",
          "mario.gustavo@hotmail.com", 27, < Masculino>, 0, "jj","12345" );
  dcl tempBool : bool := false;
  s.addUser(newUser);
  tempBool := s.tryLogin("j","12345");
  assertTrue(tempBool = false);
  tempBool := s.tryLogin("jj","12345");
  assertTrue(tempBool); assertEqual(newUser,s.getCurrentUser());
);

   private testSearchPost: () ==> ()
   testSearchPost() ==
(
      dcl user:RegularUser := new RegularUser(1, "User", "email@gmail.com", 47,
           <Feminino>, 0, "mario ","12345");
  dcl tagsList:set of Utils‘String := {"Tag1", "Tag2", "Tag3"};
  dcl tagsList2:set of Utils‘String := {"Tag3"};
  dcl newPost:Post := s.createPost("Title4", "Content", user, tagsList, 29, 12, 2017);
       dcl newPost2:Post := s.createPost("teste para Title7 meio", "Content", user,
            tagsList2, 29, 12, 2017);

  assertEqual(s.searchPostsByTags("Tag1"),{newPost});
  assertEqual(s.searchPostsByTags("Tag3"),{newPost, newPost2});
  assertEqual(s.searchPostsByTitle("Title7"),{newPost2});
);

   private testEditUser: () ==> () testEditUser() ==
(
      dcl newUser:RegularUser := s.createRegularUser("Ana", "ana@hotmail.com", 27,
           <Feminino>, 0, "aa ","12345" );
  dcl tempBool : bool; s.addUser(newUser);
  tempBool := s.tryLogin("aa","12345");
  s.editUser("Mario","mario.gustavo@hotmail.com", 23, <Masculino>);
  assertTrue(s.getCurrentUser().getName() = "Mario");
  assertTrue(s.getCurrentUser().getEmail() = "mario.gustavo@hotmail.com");
  assertTrue(s.getCurrentUser().getAge() = 23);
  assertTrue(s.getCurrentUser().getSex() = <Masculino>);
  s.d ditUser("Ana", "ana@hotmail.com", 27, <Feminino>);
  assertTrue(s.getCurrentUser().getName() = "Ana");
  assertTrue(s.getCurrentUser().getEmail() = "ana@hotmail.com");
  assertTrue(s.getCurrentUser().getAge() = 27); assertTrue(s.getCurrentUser().getSex() =
  <Feminino>);
);

  private testOrderPosts: () ==> () testOrderPosts() ==
(
      dcl user:RegularUser := new RegularUser(1, "User", "email@gmail.com", 47,
           <Feminino>, 0, "mario ","12345");
  dcl tagsList:set of Utils‘String := {"Tag1", "Tag2", "Tag3"};
  dcl newPost:Post := s.createPost("Title", "Content", user, tagsList, 01, 01, 2015);
  dcl newPost2:Post := s.createPost("Title", "Content", user, tagsList, 29, 12, 2017);
  dcl newPost3:Post := s.createPost("Title2", "Content2", user, tagsList, 27, 05, 2017);
  dcl newPost4:Post := s.createPost("Title", "Content", user, tagsList, 31, 03, 2016);
  dcl newPost5:Post := s.createPost("Title", "Content", user, tagsList, 01, 12, 2017);
  dcl newPost6:Post := s.createPost("Title", "Content", user, tagsList, 27, 05, 2017);

  newPost2.increaseRating();
  newPost2.increaseRating();
  newPost3.increaseRating();
  newPost4.decreaseRating();
  newPost5.increaseRating();
  newPost5.increaseRating();
  newPost6.decreaseRating();
  assertEqual(s.getPosts(),{newPost,newPost2,newPost3,newPost4,newPost5,newPost6}
  );
```

```
    assertEqual(s.orderPostsByDate(s.getPosts()),{newPost2, newPost5, newPost3,
    newPost6, newPost4, newPost});
    assertEqual(s.orderPostsByRating(s.getPosts()),{newPost4,newPost6,newPost,newPost
    3,newPost2, newPost5});
   );

   private testDates: () ==> ()
   testDates() ==
   (
    dcl date1:Date := new Date(2017, 5, 23);
    dcl date2:Date := new Date(2017, 5, 29);
    assertTrue(Date‘compareDates(date2, date1));
    assertEqual(Date‘compareDates(date1, date2), false);
    assertEqual(Date‘DaysOfMonth(4), 30);
    assertEqual(Date‘DaysOfMonth(2), 28);
    assertEqual(Date‘DaysOfMonth(1), 31);
   );

   public static main: () ==> ()
   main() ==
   (
    new StackOverflowTest().testcreateStackOverflow();
     new StackOverflowTest().testCreateRegisteredUser();
    new StackOverflowTest().testCreateRegularUser();
    new StackOverflowTest().testCreateAdministratorUser();
    new StackOverflowTest().testBanUser();
    new StackOverflowTest().testAddUser();
    new StackOverflowTest().testAddPost();
    new StackOverflowTest().testEditPost();
    new StackOverflowTest().testRemovePost();
    new StackOverflowTest().testSearchUser();
    new StackOverflowTest().testLogin();
    new StackOverflowTest().testAddComment();
    new StackOverflowTest().testEditComment();
    new StackOverflowTest().testRemoveComment();
    new StackOverflowTest().testRating();
    new StackOverflowTest().testSearchPost();
    new StackOverflowTest().testEditUser();
    new StackOverflowTest().testOrderPosts();
    new StackOverflowTest().testDates();
   )

end StackOverflowTest
```

4.2 Tabelas de Coverage

Tabela de Coverage da classe AdministratorUser:

Function or operation	**Line**	**Coverage**	**Calls**

AdministratorUser	3	100%	3
banUser	34	100%	2
unbanUser	39	100%	2
AdministratorUser.vdmpp		100%	7

Tabela de Coverage da classe Comment:

Function or operation	**Line**	**Coverage**	**Calls**
Comment	9	100%	4
decreaseRating	30	100%	2
editComment	17	100%	1
getAuthor	43	100%	2
getContent	36	100%	2
getDate	49	100%	6
getRating	55	100%	3
increaseRating	24	100%	1
Comment.vdmpp		100%	21

Tabela de Coverage da classe Date:

Function or operation	**Line**	**Coverage**	**Calls**
Date	7	100%	21
DaysOfMonth	14	100%	24
compareDates	24	100%	1
Date.vdmpp		100%	46

Tabela de Coverage da classe GuestUser:

Function or operation	**Line**	**Coverage**	**Calls**
GuestUser	3	100%	19
GuestUser.vdmpp		100%	19

Tabela de Coverage da classe Post:

Function or operation	**Line**	**Coverage**	**Calls**
Post	14	100%	15
addComment	35	100%	4
decreaseRating	52	100%	4
editPost	26	100%	1
getAuthor	72	100%	5
getCommentsList	104	100%	3
getDate	78	100%	36
getRating	84	100%	31
getTagsList	97	100%	6
getTitle	58	100%	46
getViews	90	100%	2
increaseRating	46	100%	6
removeComment	40	100%	1
Post.vdmpp		100%	162

Tabela de Coverage da classe RegisteredUser:

Function or operation	**Line**	**Coverage**	**Calls**
RegisteredUser	14	100%	19
addComment	87	100%	4
addPost	84	100%	15
addReputation	64	100%	26
createComment	110	100%	4
createPost	93	100%	15
deleteComment	90	100%	1
editComment	118	100%	1
editPost	100	100%	1
editUser	27	100%	2

getAge	46	100%	6
getComments	72	100%	3
getEmail	51	100%	3
getId	37	100%	2
getName	42	100%	6
getPassword	80	100%	4
getReputation	60	100%	6
getSex	55	100%	5
getUsername	76	100%	5
removeComment	122	100%	1
removePost	105	100%	1
RegisteredUser.vdmpp		100%	136

Tabela de Coverage da classe RegularUser:

Function or operation	**Line**	**Coverage**	**Calls**
RegularUser	3	100%	15
RegularUser.vdmpp		100%	15

Tabela de Coverage da classe StackOverflow:

Function or operation	**Line**	**Coverage**	**Calls**
StackOverflow	9	100%	19
addComment	49	100%	4
addPost	63	100%	15
addUser	25	100%	4
createAdministratorUser	21	100%	3
createPost	32	100%	15
createRegularUser	17	100%	6
deleteComment	58	100%	1
deletePost	43	100%	1
editComment	53	100%	1

editPost	38	100%	1
editUser	138	100%	2
getCurrentUser	83	100%	10
getPosts	79	100%	9
getUsers	75	100%	3
orderPostsByDate	153	100%	1
orderPostsByRating	173	100%	36
removePost	69	100%	1
searchPostsByTags	127	100%	2
searchPostsByTitle	105	100%	21
searchUsersByAge	96	100%	1
searchUsersByName	87	100%	1
tryLogin	144	100%	3
Stackoverflow.vdmpp		100%	160

5. Verificação do Modelo

5.1 Exemplo de verificação do domínio

Umas das “proof obligations” gerada pelo Overture foi a seguinte:

No.	PO Name	Type
42	StackOver`searchPostsByTitle(Utils`String)	legal sequence application

E o código em análise (com a parte relevante da sequência a sublinhado) é o seguinte:

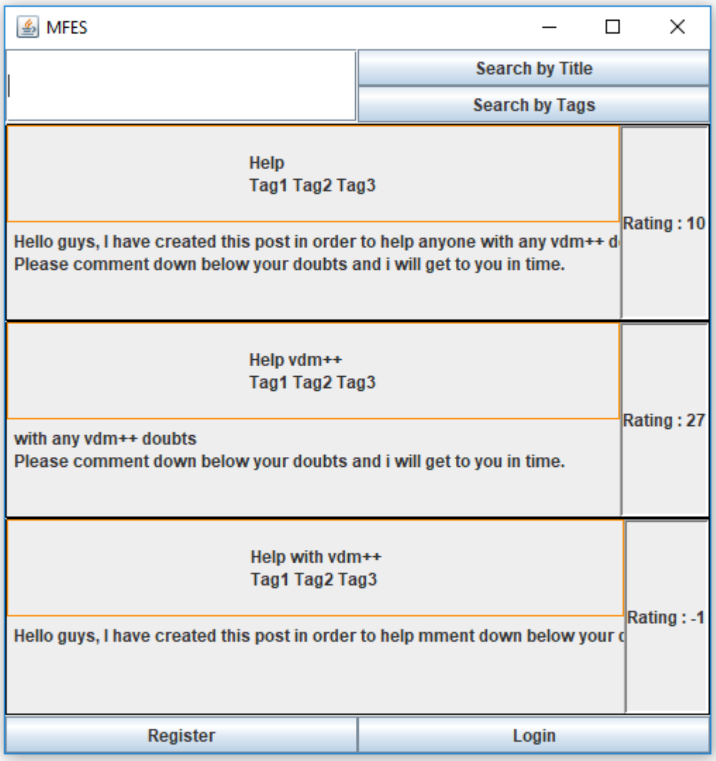
```
-- Receives a word and search the posts's title which contains that word.
       public searchPostsByTitle:Utils`String ==> set of Post
  searchPostsByTitle(inputTitle) ==
       (dcl postsTemp : set of Post := {};
       dcl inputSize : int := len inputTitle;
       dcl counter : nat := 1;
       dcl counter2 : nat := 1;
       dcl flagFound : bool := false;

       for all currPost in set postsList do
              (flagFound := false;
              counter2 := 1;
              while (counter < len currPost.getTitle()  ) do
                     (if(currPost.getTitle()(counter) = inputTitle(counter2) and
flagFound = false) then
                            (if(counter2 = inputSize) then
                                   (postsTemp := postsTemp union {currPost};
                                   flagFound := true;)
                            else (counter2 := counter2 + 1;)
                     ;)
                     else (counter2 := 1;);
                            counter := counter + 1;);
              );
       return postsTemp;)
       pre inputTitle <> "";
```

Como o “currPost.getTitle()(counter)” se encontra dentro do while, em que esse while é válido se variável counter for menor que o tamanho do título do currPost, a sequence utilizada no título do currPost nunca sairá fora do seu domínio, portanto esta proof obligation é trivial.

6. Geração do código

Para gerar o código do Java utilizou-se a funcionalidade do Overture que permite converter o modelo de VDM++ em código java. Com o código java, elaborou-se uma interface gráfica para o utilizador e os resultados foram os seguintes:


MFES

Search by Title

Search by Tags

Help
Tag1 Tag2 Tag3

Hello guys, I have created this post in order to help anyone with any vdm++ d
Please comment down below your doubts and i will get to you in time.

Rating : 10

Help vdm++
Tag1 Tag2 Tag3

with any vdm++ doubts
Please comment down below your doubts and i will get to you in time.

Rating : 27

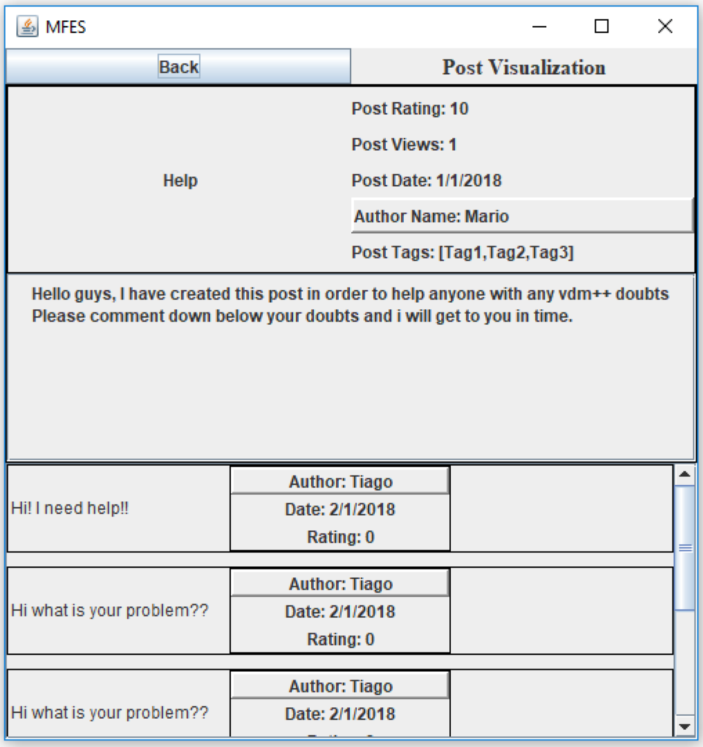
Help with vdm++
Tag1 Tag2 Tag3

Hello guys, I have created this post in order to help mment down below your d

Rating : -1

Register

Login


MFES

Back

Post Visualization

Help

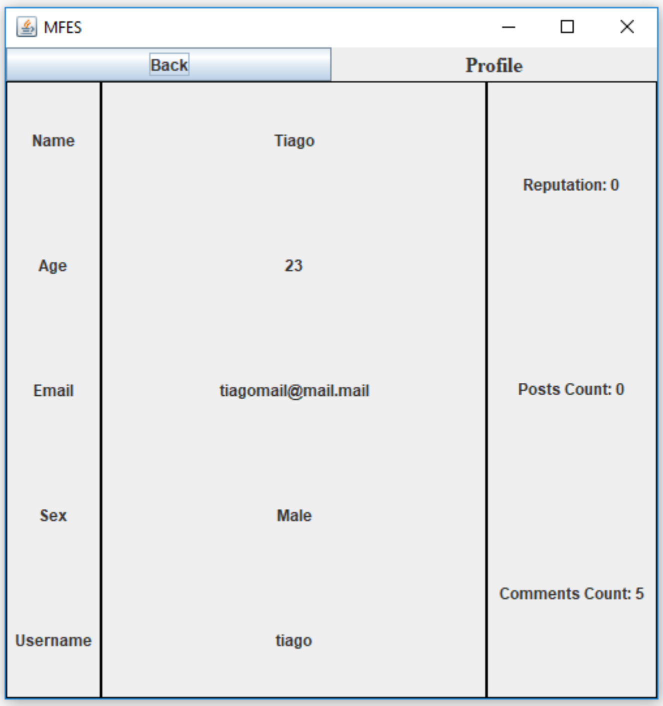
Post Rating: 10
Post Views: 1
Post Date: 1/1/2018
Author Name: Mario
Post Tags: [Tag1,Tag2,Tag3]

Hello guys, I have created this post in order to help anyone with any vdm++ doubts
Please comment down below your doubts and i will get to you in time.

Hi! I need help!!
Author: Tiago
Date: 2/1/2018
Rating: 0

Hi what is your problem??
Author: Tiago
Date: 2/1/2018
Rating: 0

Hi what is your problem??
Author: Tiago
Date: 2/1/2018



Name	Tiago	Reputation: 0
Age	23	
Email	tiagomail@mail.mail	Posts Count: 0
Sex	Male	
Username	tiago	Comments Count: 5

7. Conclusões

Relativamente ao trabalho desenvolvido conclui-se que os resultados foram bastante positivos, visto que o modelo cobre as principais funcionalidades do StackOverflow. Inicialmente existiu alguma dificuldade devido ao facto do VDM++ nos ser desconhecido, mas conforme o trabalho começou a ser desenvolvido verificou-se que era uma ferramenta bastante interessante e o facto de efectuar a conversão do modelo em VDM++ para código java foi sem dúvida um dos aspectos mais cativantes.

O trabalho foi dividido de igual modo por cada elemento do grupo.

Referências

1. StackOverflow web site, http://stackoverflow.com/
2. VDM-10 Language Manual, Peter Gorm Larsen et al, Overture Technical Report Series No. TR-001, March 2014
3. VDM++ Slides MFES, FEUP
4. Overture tool web site, http://overturetool.org