GMP Corporation

Gerenciador de Multi-Projetos

Dicionário de Dados

© 2000 GMP Corporation

Histórico de Revisões

Data	Revisão	Descrição	Autor
18/07/2004	1.0	Versão Inicial do Documento	Bárbara Siqueira Bruno Celso Diego Ribeiro Eduardo Vinicius Marcos Cardoso
26/07/2004	1.1	- Reestruturação do modelo relacional	Bruno Celso

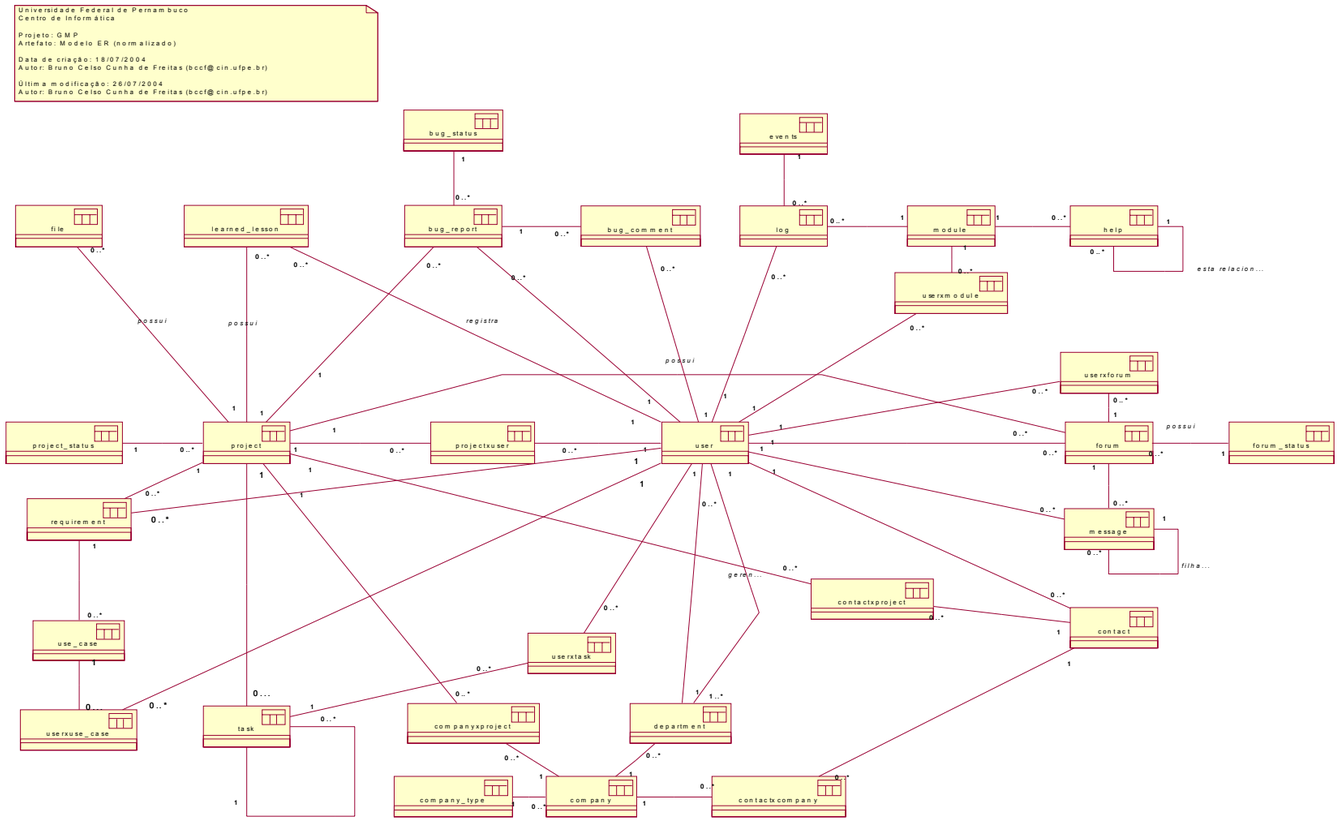
Projeto de Banco de Dados

Bárbara Siqueira Santos
Bruno Celso Cunha de Freitas
Diego Azevedo Ribeiro
Eduardo Vinicius de Figueiredo Salvador
Marcos Cardoso Jr.

{bss,bccf,dar,evfs,mjmcj@cin.ufpe.br}

Tabela de Conteúdos

1. Modelo Entidade-Relacionamento



No modelo ER acima representado, os atributos foram ocultados com o intuito de melhorar a legibilidade do diagrama. Abaixo seguem as entidades observadas no diagrama, seus atributos correspondentes e os tipos que estes assumirão no mapeamento para o esquema relacional.

ENTIDADE	ATRIBUTOS
BUG_REPORT	**bug_id : SMALLINT** subject : VARCHAR(1) date : DATE priority : SMALLINT body : VARCHAR(1)
BUG_STATUS	**bug_status_id : SMALLINT** description : VARCHAR(1)
COMPANY	**company_id : SMALLINT** name : VARCHAR(1) phone1: VARCHAR(1) phone2: VARCHAR(1) fax : VARCHAR(1) address1 : VARCHAR(1) address2 : VARCHAR(1) city : VARCHAR(1) state : VARCHAR(1) zip : VARCHAR(1) url : VARCHAR(1) description : VARCHAR(1) email : VARCHAR(1)
COMPANY_TYPE	**cp_type_id : SMALLINT** description : VARCHAR(1)
CONTACT	**contact_id : SMALLINT** title : SMALLINT first_name : VARCHAR(1) last_name : VARCHAR(1) birthday : DATE email : VARCHAR(1) phone1 : VARCHAR(1) phone2 : VARCHAR(1) mobile : VARCHAR(1)

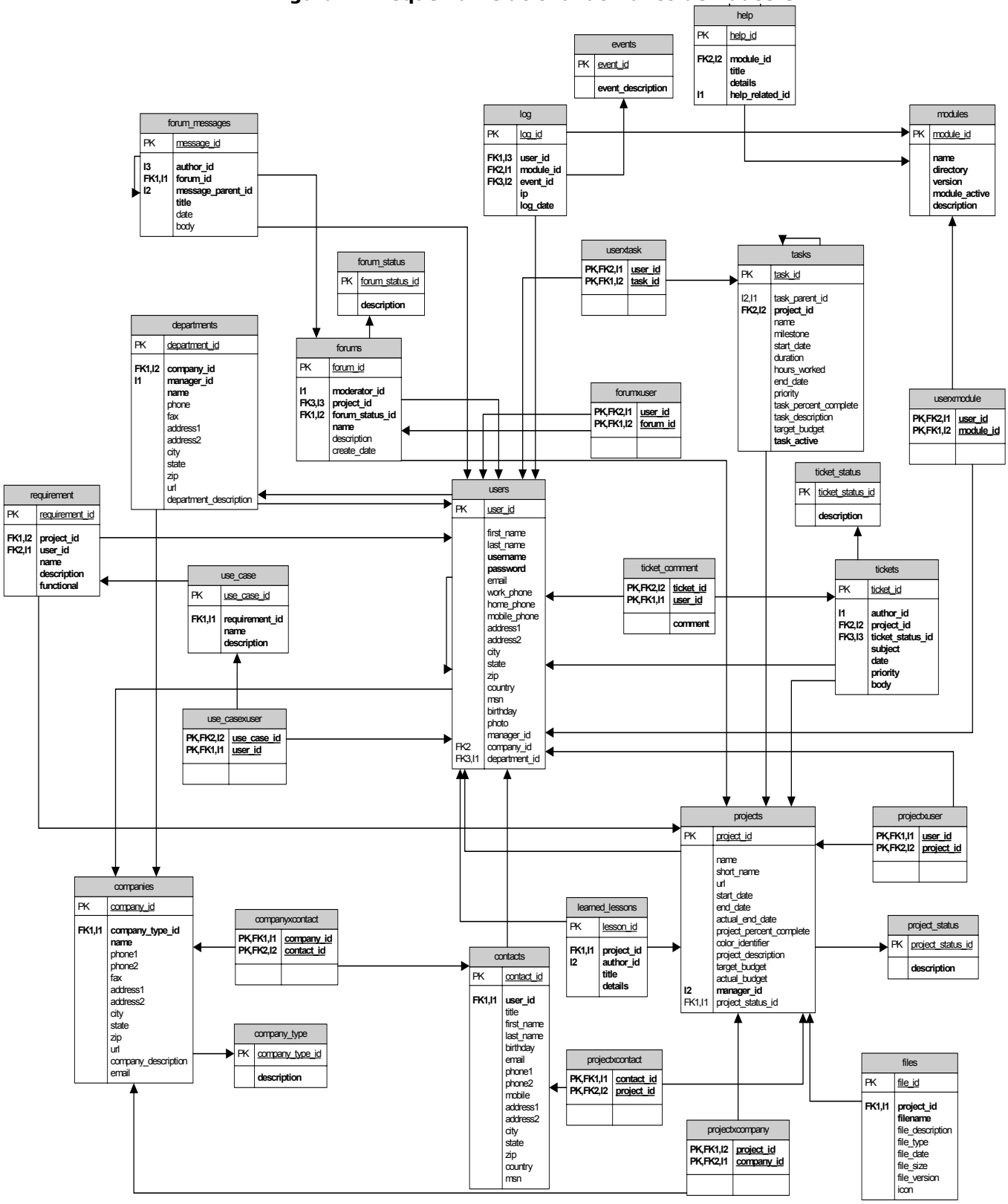
	address1 : VARCHAR(1) address2 : VARCHAR(1) city : VARCHAR(1) state : VARCHAR(1) zip : VARCHAR(1) country : VARCHAR(1) msn : VARCHAR(1)
DEPARTAMENT	**departament_id : SMALLINT** name : VARCHAR(1) phone : VARCHAR(1) fax : VARCHAR(1) address1 : VARCHAR(1) address2 : VARCHAR(1) city : VARCHAR(1) state : VARCHAR(1) zip : VARCHAR(1) url : VARCHAR(1) description : VARCHAR(1)
EVENTS	**event_id : SMALLINT** description : VARCHAR(1)
FILE	**file_id : SMALLINT** filename : TIMESTAMP WITH TIME ZONE description : VARCHAR(1) date : DATE size : INTEGER version : VARCHAR(1) icon : VARCHAR(1)
FORUM	**forum_id : SMALLINT** name : VARCHAR(1) description : VARCHAR(1) create_date : DATE last_post_date : DATE
FORUM_STATUS	**forum_status_id : SMALLINT** description : VARCHAR(1)
HELP	**help_id : SMALLINT** title : VARCHAR(1) details : VARCHAR(1)

LEARNED_LESSON	**lesson_id : SMALLINT** title : VARCHAR(1) details : VARCHAR(1)
LOG	**log_id : SMALLINT** ip : VARCHAR(1) date : DATE
MESSAGE	**message_id : SMALLINT** title : VARCHAR(1) date : DATE body : SMALLINT
MODULE	**module_id : SMALLINT** name : VARCHAR(1) directory : VARCHAR(1) version : VARCHAR(1) active : BIT(1) description : VARCHAR(1)
PROJECT	**project_id : SMALLINT** name : VARCHAR(1) short_name : VARCHAR(1) url : VARCHAR(1) start_date : DATE end_date : DATE actual_end_date : DATE percent_complete : FLOAT color_identifier : VARCHAR(1) description : VARCHAR(1) target_budget : FLOAT actual_budget : FLOAT
PROJECT_STATUS	**proj_status_id : SMALLINT** description : VARCHAR(1)
REQUIREMENT	**requirement_id : SMALLINT** name : VARCHAR(1) description : VARCHAR(1) functional : BIT(1)
TASK	**task_id : SMALLINT** name : VARCHAR(1) milestone : BIT(1)

	start_date : DATE duration : INTEGER hours_worked : INTEGER end_date : DATE priority : SMALLINT percent_complete : FLOAT description : VARCHAR(1) target_buget : FLOAT active : BIT(1)
USE_CASE	**uc_case_id : SMALLINT** name : SMALLINT description : SMALLINT
USER	**user_id : SAMLLINT** first_name : VARCHAR(1) last_name : VARCHAR(1) username : VARCHAR(1) password : VARCHAR(1) email : VARCHAR(1) work_phone : VARCHAR(1) mobile_phone : VARCHAR(1) address1 : VARCHAR(1) address2 : VARCHAR(1) city : VARCHAR(1) state : VARCHAR(1) zip : VARCHAR(1) country : VARCHAR(1) msn : VARCHAR(1) birthday : DATE photo : BIT(1)

2. Esquema Relacional

Figura 1 – Esquema Relacional do Banco de Dados GMP



3. Usuários

Visando garantir a flexibilidade na administração da base de dados e a segurança da aplicação, criamos duas contas de usuários:

Usuário	Descrição
gmp_root	Este usuário será utilizado pelos desenvolvedores do sistema para efetuar qualquer alteração na estrutura da base de dados. Este usuário terá liberdade para realizar qualquer operação na base de dados GMP.
gmp_app	Este usuário será utilizado pela aplicação e, portanto, sua liberdade de atuação é restrita. Apenas as operações de select, insert, update e delete são liberadas para este usuário.

4. Stored Procedures e Triggers

Não foi identificada a necessidade do uso de stored procedures e triggers uma vez que se o controle das regras da aplicação for feito pela própria aplicação, a migração da base de dados atual implementada em MySQL para um outro SGBD será direta, não necessitará de modificações no script de criação do banco.

5. Scripts

```
#
# Universidade Federal de Pernambuco
# Centro de Informática
#
# Projeto: GMP
# Artefato: Script de criação da base de dados
# Versão: 1.1.0
# Data de criação: 18/07/2004
# Autor: Bruno Celso Cunha de Freitas (bccf@cin.ufpe.br)
#
```

```
# Última modificação: 26/07/2004
# Autor: Bruno Celso Cunha de Freitas (bccf@cin.ufpe.br)
#

#---------------------------------------------- CRIANDO O BANCO DE DADOS ----------------------------------------------#

# Excluindo qualquer instancia da base de dados
DROP DATABASE IF EXISTS gmp_teste;

# Criando a base de dados do sistema
CREATE DATABASE gmp_teste;

# Selecionando a base de dados criada para a criação das tabelas
USE gmp_teste;

#---------------------------------------------- CRIANDO AS TABELAS ----------------------------------------------#

#
# Eliminando qualquer instancia atual das tabelas
#
DROP TABLE IF EXISTS companies;
DROP TABLE IF EXISTS company_type;
DROP TABLE IF EXISTS companyxcontact;
DROP TABLE IF EXISTS departments;
DROP TABLE IF EXISTS contacts;
DROP TABLE IF EXISTS events;
DROP TABLE IF EXISTS files;
DROP TABLE IF EXISTS forum_messages;
DROP TABLE IF EXISTS forums;
DROP TABLE IF EXISTS forumxuser;
DROP TABLE IF EXISTS forum_status;
DROP TABLE IF EXISTS userxmodule;
DROP TABLE IF EXISTS projects;
DROP TABLE IF EXISTS projectxuser;
DROP TABLE IF EXISTS projectxcontact;
DROP TABLE IF EXISTS project_status;
DROP TABLE IF EXISTS projectxcompany;
DROP TABLE IF EXISTS tasks;
DROP TABLE IF EXISTS tickets;
DROP TABLE IF EXISTS ticket_status;
DROP TABLE IF EXISTS ticket_comment;
DROP TABLE IF EXISTS users;
DROP TABLE IF EXISTS userxtasks;
DROP TABLE IF EXISTS modules;
DROP TABLE IF EXISTS learned_lessons;
DROP TABLE IF EXISTS log;
DROP TABLE IF EXISTS requirement;
DROP TABLE IF EXISTS use_case;
DROP TABLE IF EXISTS use_casexuser;
```

```
DROP TABLE IF EXISTS help;

#
# Esta tabela armazena os dados inerentes a uma companhia
#
CREATE TABLE companies (
  company_id INT(11) NOT NULL auto_increment,
  company_type_id int(11) NOT NULL,
  name varchar(100) NOT NULL,
  phone1 varchar(30),
  phone2 varchar(30),
  fax varchar(30),
  address1 varchar(50),
  address2 varchar(50),
  city varchar(30),
  state varchar(30),
  zip varchar(11),
  url varchar(255),
  company_description text,
  email varchar(255),
  PRIMARY KEY (company_id)
) TYPE=InnoDB;

#
# Esta tabela armazena os tipos de empresa que podem ser atribuídos
#
CREATE TABLE company_type (
  company_type_id INT(11) NOT NULL auto_increment,
  description VARCHAR(255) NOT NULL,
  PRIMARY KEY (company_type_id)
) TYPE=InnoDB;

#
# Esta tabela armazena o relacionamento entre companhias e contatos
#
CREATE TABLE companyxcontact (
  company_id INT(11) NOT NULL,
  contact_id INT(11) NOT NULL,
  PRIMARY KEY(company_id, contact_id)
) TYPE=InnoDB;

#
# Esta tabela armazena os dados inerentes aos departamentos de uma companhia
#
CREATE TABLE departments (
  department_id int(10) NOT NULL auto_increment,
  company_id int(10) NOT NULL,
  manager_id int(3) NOT NULL,
  name tinytext NOT NULL,
```

```
  phone varchar(30),
  fax varchar(30),
  address1 varchar(30),
  address2 varchar(30),
  city varchar(30),
  state varchar(30),
  zip varchar(11),
  url varchar(25),
  department_description text,
  PRIMARY KEY  (department_id)
) TYPE=InnoDB;

#
# Esta tabela guarda os dados dos contatos para a execução dos projetos
#
CREATE TABLE contacts (
  contact_id int(3) NOT NULL auto_increment,
  user_id int(3) NOT NULL,
  title varchar(50),
  first_name varchar(30),
  last_name varchar(30),
  birthday datetime,
  email varchar(255),
  phone1 varchar(30),
  phone2 varchar(30),
  mobile varchar(30),
  address1 varchar(30),
  address2 varchar(30),
  city varchar(30),
  state varchar(30),
  zip varchar(11),
  country varchar(30),
  msn varchar(50),
  PRIMARY KEY  (contact_id)
) TYPE=InnoDB;

#
# Esta tabela guarda os dados dos eventos do sistema
#
CREATE TABLE events (
  event_id int(3) NOT NULL auto_increment,
  event_description text NOT NULL,
  PRIMARY KEY  (event_id)
) TYPE=InnoDB;

#
# Esta tabela guarda os dados dos arquivos postados pelos usuários
#
CREATE TABLE files (
```

```
  file_id int(3) NOT NULL auto_increment,
  project_id int(3) NOT NULL,
  filename varchar(255) NOT NULL,
  file_description text,
  file_type varchar(100),
  file_date datetime,
  file_size int(11),
  file_version float,
  icon varchar(20),
  PRIMARY KEY  (file_id)
) TYPE=InnoDB;

#
# Esta tabela armazena os dados das mensagens postadas nos foruns.
#
CREATE TABLE forum_messages (
  message_id int(11) NOT NULL auto_increment,
  author_id int(11) NOT NULL,
  forum_id int(11) NOT NULL,
  message_parent_id int(11) NOT NULL,
  title varchar(255) NOT NULL,
  date datetime,
  body text,
  PRIMARY KEY  (message_id)
) TYPE=InnoDB;

#
# Esta tabela armazena os dados inerentes aos foruns de discussao
#
CREATE TABLE forums (
  forum_id int(11) NOT NULL auto_increment,
  moderator_id int(11) NOT NULL,
  project_id int(11) NOT NULL,
  forum_status_id int(11) NOT NULL,
  name varchar(50) NOT NULL,
  description varchar(255),
  create_date datetime,
  PRIMARY KEY  (forum_id)
) TYPE=InnoDB;

#
# Esta tabela armazena os registros de quais usuarios podem acessar quais forums
#
CREATE TABLE forumxuser (
  user_id INT(11) NOT NULL,
  forum_id INT(11) NOT NULL,
  PRIMARY KEY (user_id, forum_id)
) TYPE=InnoDB;
```

```
#
# Esta tabela armazena os status que podem ser atribuídos aos foruns
#
CREATE TABLE forum_status (
  forum_status_id INT(11) NOT NULL auto_increment,
  description VARCHAR(255) NOT NULL,
  PRIMARY KEY (forum_status_id)
) TYPE=InnoDB;

#
# Esta tabela armazena os dados inerentes aos projetos
#
CREATE TABLE projects (
  project_id int(11) NOT NULL auto_increment,
  name varchar(255),
  short_name varchar(10),
  url varchar(255),
  start_date datetime,
  end_date datetime,
  actual_end_date datetime,
  project_percent_complete tinyint(4),
  color_identifier varchar(6),
  project_description text,
  target_budget int(11),
  actual_budget int(11),
  manager_id int(11) NOT NULL,
  project_status_id int(11),
  PRIMARY KEY  (project_id)
) TYPE=InnoDB;

#
# Esta tabela armazena os registros de participação dos usuarios nos diversos projetos cadastrados
#
CREATE TABLE projectxuser (
  user_id int(11) NOT NULL,
  project_id int(11) NOT NULL,
  PRIMARY KEY (user_id, project_id)
) TYPE=InnoDB;

#
# Esta tabela armazena os registros de contatos relacionados com os projetos
#
CREATE TABLE projectxcontact (
  contact_id int(11) NOT NULL,
  project_id int(11) NOT NULL,
  PRIMARY KEY(contact_id, project_id)
) TYPE=InnoDB;

#
```

```
# Esta tabela armazena os status que os projetos podem assumir
#
CREATE TABLE project_status (
  project_status_id INT(11) NOT NULL auto_increment,
  description VARCHAR(255) NOT NULL,
  PRIMARY KEY (project_status_id)
) TYPE=InnoDB;

#
# Esta tabela armazena os registros entre os projetos e as companhias que os patrocinam
#
CREATE TABLE projectxcompany (
  project_id INT(11) NOT NULL,
  company_id INT(11) NOT NULL,
  PRIMARY KEY(project_id,company_id)
) TYPE=InnoDB;

#
# Esta tabela armazena os dados inerentes as tarefas dos projetos
#
CREATE TABLE tasks (
  task_id int(11) NOT NULL auto_increment,
  task_parent_id int(11),
  project_id int(11) NOT NULL,
  name varchar(255),
  milestone tinyint(1),
  start_date datetime,
  duration float unsigned,
  hours_worked float unsigned,
  end_date datetime,
  priority tinyint(4),
  task_percent_complete float,
  task_description text,
  target_budget int(11),
  task_active tinyint(1) NOT NULL,
  PRIMARY KEY  (task_id)
) TYPE=InnoDB;

#
# Esta tabela armazena os dados dos chamados de reparo solicitados pelos membros dos projetos
#
CREATE TABLE tickets (
  ticket_id int(11) NOT NULL auto_increment,
  author_id int(11) NOT NULL,
  project_id int(11) NOT NULL,
  ticket_status_id int(11) NOT NULL,
  subject varchar(100) NOT NULL,
  date datetime NOT NULL,
  priority tinyint(1) unsigned NOT NULL,
```

```
  body text NOT NULL,
  PRIMARY KEY  (ticket_id)
) TYPE=InnoDB;

#
# Esta tabela armazena os status que os registros de reparo podem receber
#
CREATE TABLE ticket_status (
  ticket_status_id INT(11) NOT NULL auto_increment,
  description VARCHAR(255) NOT NULL,
  PRIMARY KEY (ticket_status_id)
) TYPE=InnoDB;

#
# Esta tabela armazena os registros de comentarios dos usuarios acerca de um chamado de reparo
#
CREATE TABLE ticket_comment (
  ticket_id INT(11) NOT NULL,
  user_id INT(11) NOT NULL,
  comment text NOT NULL,
  PRIMARY KEY (ticket_id,user_id)
) TYPE=InnoDB;

#
# Esta tabela armazena os dados inerentes aos usuários
#
CREATE TABLE users (
  user_id int(11) NOT NULL auto_increment,
  first_name varchar(50),
  last_name varchar(50),
  username varchar(20) NOT NULL,
  password varchar(32) NOT NULL,
  email varchar(255),
  work_phone varchar(30),
  home_phone varchar(30),
  mobile_phone varchar(30),
  address1 varchar(30),
  address2 varchar(30),
  city varchar(30),
  state varchar(30),
  zip varchar(11),
  country varchar(30),
  msn varchar(20),
  birthday datetime,
  photo tinyint(1),
  manager_id int(11),
  company_id int(11),
  department_id int(11),
  PRIMARY KEY  (user_id)
```

```
) TYPE=InnoDB;

#
# Esta tabela relaciona as tarefas dos projetos e seus responsaveis
#
CREATE TABLE userxtask (
  user_id int(11) NOT NULL,
  task_id int(11) NOT NULL,
  PRIMARY KEY  (user_id,task_id)
) TYPE=InnoDB;

#
# Esta tabela armazena as permissões que os usuarios do sistema terao aos modulos do mesmo.
#
CREATE TABLE userxmodule (
  user_id int(11) NOT NULL,
  module_id int(11) NOT NULL,
  PRIMARY KEY  (user_id,module_id)
) TYPE=InnoDB;

#
# Esta tabela armazena dados genericos dos modulos funcionais do sistema
#
CREATE TABLE modules (
  module_id int(11) NOT NULL auto_increment,
  name varchar(64) NOT NULL,
  directory varchar(64) NOT NULL,
  version varchar(10) NOT NULL,
  module_active int(1) unsigned NOT NULL,
  description varchar(255) NOT NULL,
  PRIMARY KEY  (module_id)
) TYPE=InnoDB;

#
# Esta tabela armazena as lições aprendidas de um projeto
#
CREATE TABLE learned_lessons (
  lesson_id int(11) NOT NULL auto_increment,
  project_id int(11) NOT NULL,
  author_id int(11) NOT NULL,
  title varchar(100) NOT NULL,
  details text NOT NULL,
  PRIMARY KEY  (lesson_id)
) TYPE=InnoDB;

#
# Esta tabela armazena os registros de log do sistema
#
CREATE TABLE log (
```

```
  log_id INT(11) NOT NULL auto_increment,
  user_id INT(11) NOT NULL,
  module_id int(11) NOT NULL,
  event_id int(11) NOT NULL,
  ip varchar(15) NOT NULL,
  log_date DATETIME NOT NULL,
  PRIMARY KEY  (log_id)
) TYPE=InnoDB;

#
# Esta tabela armazena os requisitos dos projetos
#
CREATE TABLE requirement (
  requirement_id INT(11) NOT NULL auto_increment,
  project_id INT(11) NOT NULL,
  user_id INT(11) NOT NULL,
  name VARCHAR(255) NOT NULL,
  description VARCHAR(255) NOT NULL,
  functional tinyint(1) NOT NULL,
  PRIMARY KEY (requirement_id)
) TYPE=InnoDB;

#
# Esta tabela armazena os casos de uso dos projetos
#
CREATE TABLE use_case (
  use_case_id INT(11) NOT NULL auto_increment,
  requirement_id INT(11) NOT NULL,
  name varchar(255) NOT NULL,
  description varchar(255) NOT NULL,
  PRIMARY KEY (use_case_id)
) TYPE=InnoDB;

#
# Esta tabela armazena os relacionamentos entre os casos de uso e os responsaveis por sua implementacao
#
CREATE TABLE use_casexuser (
  use_case_id INT(11) NOT NULL,
  user_id INT(11) NOT NULL,
  PRIMARY KEY (use_case_id,user_id)
) TYPE=InnoDB;

#
# Esta tabela armazena os dados do help on-line do sistema
#
CREATE TABLE help (
  help_id INT(11) NOT NULL auto_increment,
  module_id INT(11) NOT NULL,
  title VARCHAR(255) NOT NULL,
```

```
  details text NOT NULL,
  help_related_id INT(11) NOT NULL,
  PRIMARY KEY (help_id)
) TYPE=InnoDB;

ALTER TABLE companies ADD INDEX(company_type_id);
ALTER TABLE companies ADD FOREIGN KEY(company_type_id) REFERENCES company_type(company_type_id);

ALTER TABLE departments ADD INDEX(manager_id);
ALTER TABLE departments ADD FOREIGN KEY (manager_id) REFERENCES users(user_id);
ALTER TABLE departments ADD INDEX(company_id);
ALTER TABLE departments ADD FOREIGN KEY (company_id) REFERENCES companies(company_id);

ALTER TABLE companyxcontact ADD INDEX(company_id);
ALTER TABLE companyxcontact ADD FOREIGN KEY (company_id) REFERENCES companies(company_id);
ALTER TABLE companyxcontact ADD INDEX(contact_id);
ALTER TABLE companyxcontact ADD FOREIGN KEY (contact_id) REFERENCES contacts(contact_id);

ALTER TABLE files ADD INDEX(project_id);
ALTER TABLE files ADD FOREIGN KEY (project_id) REFERENCES projects(project_id);

ALTER TABLE forum_messages ADD INDEX (forum_id);
ALTER TABLE forum_messages ADD FOREIGN KEY (forum_id) REFERENCES forums(forum_id);
ALTER TABLE forum_messages ADD INDEX(message_parent_id);
ALTER TABLE forum_messages ADD FOREIGN KEY (message_parent_id) REFERENCES forum_messages(message_id);
ALTER TABLE forum_messages ADD INDEX(author_id);
ALTER TABLE forum_messages ADD FOREIGN KEY (author_id) REFERENCES users(user_id);

ALTER TABLE forums ADD INDEX(moderator_id);
ALTER TABLE forums ADD FOREIGN KEY (moderator_id) REFERENCES users(user_id);
ALTER TABLE forums ADD INDEX(forum_status_id);
ALTER TABLE forums ADD FOREIGN KEY (forum_status_id) REFERENCES forum_status(forum_status_id);
ALTER TABLE forums ADD INDEX(project_id);
ALTER TABLE forums ADD FOREIGN KEY (project_id) REFERENCES projects(project_id);

ALTER TABLE users ADD INDEX(department_id);
ALTER TABLE users ADD FOREIGN KEY (department_id) REFERENCES departments(department_id);

ALTER TABLE tasks ADD INDEX(task_parent_id);
ALTER TABLE tasks ADD FOREIGN KEY (task_parent_id) REFERENCES tasks(task_id);
ALTER TABLE tasks ADD INDEX(project_id);
ALTER TABLE tasks ADD FOREIGN KEY (project_id) REFERENCES projects(project_id);

ALTER TABLE userxmodule ADD INDEX(user_id);
ALTER TABLE userxmodule ADD FOREIGN KEY (user_id) REFERENCES users(user_id);
ALTER TABLE userxmodule ADD INDEX(module_id);
ALTER TABLE userxmodule ADD FOREIGN KEY (module_id) REFERENCES modules(module_id);

ALTER TABLE projects ADD INDEX(project_status_id);
```

```
ALTER TABLE projects ADD FOREIGN KEY (project_status_id) REFERENCES project_status(project_status_id);
ALTER TABLE projects ADD INDEX(manager_id);
ALTER TABLE projects ADD FOREIGN KEY (manager_id) REFERENCES users(user_id);

ALTER TABLE learned_lessons ADD INDEX(project_id);
ALTER TABLE learned_lessons ADD FOREIGN KEY (project_id) REFERENCES projects(project_id);
ALTER TABLE learned_lessons ADD INDEX(author_id);
ALTER TABLE learned_lessons ADD FOREIGN KEY (author_id) REFERENCES users(user_id);

ALTER TABLE use_case ADD INDEX(requirement_id);
ALTER TABLE use_case ADD FOREIGN KEY (requirement_id) REFERENCES requirement(requirement_id);

ALTER TABLE contacts ADD INDEX(user_id);
ALTER TABLE contacts ADD FOREIGN KEY (user_id) REFERENCES users(user_id);

ALTER TABLE log ADD INDEX(module_id);
ALTER TABLE log ADD FOREIGN KEY (module_id) REFERENCES modules(module_id);
ALTER TABLE log ADD INDEX(event_id);
ALTER TABLE log ADD FOREIGN KEY (event_id) REFERENCES events(event_id);
ALTER TABLE log ADD INDEX(user_id);
ALTER TABLE log ADD FOREIGN KEY (user_id) REFERENCES users(user_id);

ALTER TABLE tickets ADD INDEX(author_id);
ALTER TABLE tickets ADD FOREIGN KEY (author_id) REFERENCES users(user_id);
ALTER TABLE tickets ADD INDEX(project_id);
ALTER TABLE tickets ADD FOREIGN KEY (project_id) REFERENCES projects(project_id);
ALTER TABLE tickets ADD INDEX(ticket_status_id);
ALTER TABLE tickets ADD FOREIGN KEY (ticket_status_id) REFERENCES ticket_status(ticket_status_id);

ALTER TABLE userxtask ADD INDEX(user_id);
ALTER TABLE userxtask ADD FOREIGN KEY (user_id) REFERENCES users(user_id);
ALTER TABLE userxtask ADD INDEX(task_id);
ALTER TABLE userxtask ADD FOREIGN KEY (task_id) REFERENCES tasks(task_id);

ALTER TABLE projectxuser ADD INDEX(user_id);
ALTER TABLE projectxuser ADD FOREIGN KEY (user_id) REFERENCES users(user_id);
ALTER TABLE projectxuser ADD INDEX(project_id);
ALTER TABLE projectxuser ADD FOREIGN KEY (project_id) REFERENCES projects(project_id);

ALTER TABLE projectxcontact ADD INDEX(contact_id);
ALTER TABLE projectxcontact ADD FOREIGN KEY (contact_id) REFERENCES contacts(contact_id);
ALTER TABLE projectxcontact ADD INDEX(project_id);
ALTER TABLE projectxcontact ADD FOREIGN KEY (project_id) REFERENCES projects(project_id);

ALTER TABLE forumxuser ADD INDEX(user_id);
ALTER TABLE forumxuser ADD FOREIGN KEY (user_id) REFERENCES users(user_id);
ALTER TABLE forumxuser ADD INDEX(forum_id);
ALTER TABLE forumxuser ADD FOREIGN KEY (forum_id) REFERENCES forums(forum_id);
```

```
ALTER TABLE help ADD INDEX(help_related_id);
ALTER TABLE help ADD FOREIGN KEY (help_related_id) REFERENCES help(help_id);
ALTER TABLE help ADD INDEX(module_id);
ALTER TABLE help ADD FOREIGN KEY (module_id) REFERENCES modules(module_id);

ALTER TABLE use_casexuser ADD INDEX(user_id);
ALTER TABLE use_casexuser ADD FOREIGN KEY (user_id) REFERENCES users(user_id);
ALTER TABLE use_casexuser ADD INDEX(use_case_id);
ALTER TABLE use_casexuser ADD FOREIGN KEY (use_case_id) REFERENCES use_case(use_case_id);

ALTER TABLE requirement ADD INDEX(user_id);
ALTER TABLE requirement ADD FOREIGN KEY (user_id) REFERENCES users(user_id);
ALTER TABLE requirement ADD INDEX(project_id);
ALTER TABLE requirement ADD FOREIGN KEY (project_id) REFERENCES projects(project_id);

ALTER TABLE projectxcompany ADD INDEX(company_id);
ALTER TABLE projectxcompany ADD FOREIGN KEY (company_id) REFERENCES companies(company_id);
ALTER TABLE projectxcompany ADD INDEX(project_id);
ALTER TABLE projectxcompany ADD FOREIGN KEY (project_id) REFERENCES projects(project_id);

ALTER TABLE ticket_comment ADD INDEX(user_id);
ALTER TABLE ticket_comment ADD FOREIGN KEY (user_id) REFERENCES users(user_id);
ALTER TABLE ticket_comment ADD INDEX(ticket_id);
ALTER TABLE ticket_comment ADD FOREIGN KEY (ticket_id) REFERENCES tickets(ticket_id);


#------------------------------------------------ CRIANDO OS USUÁRIOS -------------------------------------------------#

#
# O usuário gmp_root será utilizado pelos desenvolvedores do sistema para efetuar alterações no banco de dados.
#
GRANT ALL ON gmp.* TO gmp_root;

#
# O usuário gmp_app será utilizado pela aplicação, ou seja, pelos usuários do sistema e terá restrições
# (apenas efetuará selects, updates, inserts e deletes) para garantir a segurança da base de dados contra ataques.
#
GRANT SELECT, INSERT, UPDATE, DELETE ON gmp.* TO gmp_app;

#------------------------------------------------ POVOANDO A BASE DE DADOS -------------------------------------------------#
```

6. Referências

[1] Documento de Requisitos do Sistema GMP.

[2] dotProject. Acessível em: http://www.dotproject.net. Último Acesso: 18/07/2004.