Administrator's Guide

Framework NET Genium

Content

1 Basic information

1.1 Framework description

NET Genium is a framework designed for Xtreme Rapid Development, focusing on web applications, which are based on three basic pillars: on the organization, visualization and sharing of data.

Organization

Organization means acquiring, modifying or deleting data, as well as searching, filtering, exporting or importing data, either online in a web browser or offline using a native application.

Visualization

Visualization means displaying data in one of the most commonly used forms, i.e. in a spreadsheet, or in an Outlook calendar, in a map, in a gantt chart, in a multi-source calendar, or in a tree structure, and further means the aggregation and subsequent display of data in either a regular table, a PivotTable, or a chart.

Sharing

Sharing means that users have access to all the information online. Each user works under their own user account, which defines their roles or permissions, and leaves a digital footprint, as any changes to the data they make are recorded in the history.

Processes, workflows and algorithms

Comfort in working with data is a standard of all web applications based on the NET Genium framework. But the most interesting thing for any application administrator is that they can easily implement specific processes, workflows or algorithms into their application that are unique to each end customer. The functionality of the three basic pillars is ensured completely automatically, so the application administrator can fully focus only on activities that provide added value to the entire application.

Customized information systems and functional prototypes

NET Genium is great for creating simple and complex customized information systems, or for preparing functional prototypes of applications that, in our experience, do not differ significantly from the final assignment. NET Genium creates robust web applications that are stable, secure and scalable. These applications can be run in the cloud or on private or office servers. They only require a web browser with javascript support to work.

1.2 Technical parameters of the framework

- Web application written in C# on the .NET platform
- Support for MS SQL Server or Firebird databases
- A multi-user application with its own system of user rights, independent of IIS or Active Directory
- Integrated graphic designer designed to develop applications for users with administrator privileges
- Consistent support for most web browsers and responsive design
- Safety in accordance with the OWASP methodology guaranteed by several independent penetration tests
- Optimized performance for less powerful or office servers

- Optimized performance for working with large amounts of data and full-text search in database records and file attachments
- In-Memory table support
- Single sign-on support in conjunction with Active Directory
- Support for running an application in multiple instances with different IIS security level settings
- Support for running an application in multiple clones – operational, development or archive environment
- Tools for releasing new releases from the development to the production or archive environment
- Tools for moving time-limited and at the same time consistent data to the archive environment
- Tools for exporting, importing or converting large amounts of data

1.3 Technical parameters of development using the NET Genium framework

- Composing and connecting two basic building blocks of applications – edit forms and view pages
- Library of controls for smart tables, classic calendars or planning multi-source calendars, timetables, gantt charts, graphs, maps or tree structures
- Support for multilingualism
- Graphic designer of database queries
- Graphical designer of server-side scripts
- Calling the most frequently used functions for working with data from the function library
- Calling custom functions written in C#
- Tools for running scripts from scheduled tasks
- CodeMirror for writing JavaScript
- Support for printing to XLS, DOC, TXT or HTML printing templates
- Automatic conversion of print reports to PDF
- Sync data with Android or iOS devices
- Symmetric data synchronization (multiple master) with any application based on the NET Genium platform
- API for communication with any application based on the .NET platform

2 Login to NET Genium

- NET Genium supports three basic types of user authorization:
  - **Common user** – works with data, and has only content and functions available based on its permissions
  - **Administrator** – has access to all data, and creates or modifies applications using the integrated designer
  - **Anonymous user** – fills in data in web forms, which become part of web presentations thanks to iframes
- The licensing model is based on the maximum number of users logged in to the application at the same time. The total number of licenses can be expanded at any time.
- Each login requires one user license. The user can log in to the application more than once from one IP address, however, only one license will be drawn. Anonymous users do not need a license.
- The default installation of NET Genium contains two basic user groups:
  - **Users** – regular users
  - **Administrators** – administrators

3 Control area

- The control surface consists of three basic parts:
  - **Header** – the header consists of a thematically tuned color bar with an optional logo, the name of the currently logged in user, the number of total logged in users, the current date, a link to log off the user, and a list of available application groups
  - **Navigation area** – The navigation area consists of a list of available applications and a list of favorites
  - **Main window** – The main window displays three types of pages:
    - Portlets were not the main page or dashboard
    - View pages
    - Edit forms
- A detailed description of the control surface is given in the separate manual “Control surface”.

4 Application

- NET Genium consists of a set of interconnected applications.
- Each application can be created from two basic building blocks of applications – edit forms and view pages.
    - The edit form is used to edit one database record.
    - The view page displays multiple records at once, and is used to visualize, view, filter, and search database records.
    - The view page is also an intermediary for entering the edit form – either in order to create a new record in the database, or in order to edit a specific existing record.
    - Edit forms and view pages can contain buttons that are used to trigger events. Such an event can be the opening of another edit form or view page. The most common way to open an edit form from a view page to create a new database record is to place a button on the view page that opens the edit form.
    - An application that contains only edit forms lacks an interface that allows you to access edit forms. Therefore, the smallest application must contain at least one view page.
- Applications can be created or modified only by administrators, i.e. users belonging to the administrators group. Only one user with the highest authorization can export and import applications – the “Administrator” user with database ID 1.
- Editing applications can only be done in administrator mode. Administrator mode is available only to administrators who can freely switch between user and administrator mode.
- In administrator mode, the web application is displayed almost the same as in user mode, only in addition, navigation elements and control tools are available for editing applications or their parts.
- Editing applications consists of the following activities:
    - creating or modifying edit forms,
    - creating or modifying view pages,
    - creating or modifying controls that make up edit forms and view pages,
    - grouping view pages into categories.
- A detailed description of application settings is given in a separate “Applications” manual.

5 Application groups

- Applications are grouped into application groups for better clarity. Application groups usually contain applications with the same or similar focus.
- At the level of application groups, it is possible to define specific administrators who can modify applications within the application group. Other administrators cannot make any changes to these applications, even though they are in administrator mode.
- NET Genium automatically merges application groups with applications if the application group contains only one application that has the same name as the application group.
- A detailed description of the application group settings is given in the separate “Application group” manual.

6 Edit forms

- The edit form is a graphical interface that is used to edit one database record – either from and to create a new record in the database, or to edit a specific existing record.
- Each edit form has one database table associated with it inside the database, in which the user data filled in by the user in the form is stored. The name of the database table is derived from the name of the edit form, and begins with the prefix "ng_". The process of creating or modifying an associated database table is completely automatic, so administrators never change the database table itself inside the database, but only change the properties of the edit form or its controls.
- Each associated database table contains the following database columns:
    - "id" – the primary key or unique identifier of each record in the database table, which has either the database type "int NOT NULL" or "long NOT NULL", depending on the settings of the edit form.
    - "system" – uses the value "True" to specify a system database record that cannot be deleted. An example is an entry with ID 1 (Administrator) and 2 (Anonymous) in the database table of NET Genium users, which are essential for the functionality of the entire application.
    - "pid" – ID of the parent record
    - "pform" – ID of the edit form in which the parent record is located
    - "userid" – ID of the user who is the owner of the database record, resp. created a record in the database
- A detailed description of edit form settings is given in a separate manual "Edit form".

6.1 Edit form controls

- The edit form consists of database and visual controls.
    - Each database control has an associated one column in the associated database table, in which the values filled in the edit form are stored. For an edit form to make sense, it must contain at least one database control.
    - The visual controls in the form are for information only – they are, for example, the title, view table, button, graphic separator with or without a horizontal line, etc.
    - Each edit form automatically contains an "ActionButtons" control, which consists of three basic buttons needed to work with the form – "Save", "Delete" and "Back". A detailed description of the "ActionButtons" control is given in the separate "ActionButtons – Control in Edit Form" manual.
- The controls are arranged one below the other in the edit form, or they can optically form more columns by snapping the controls to the previous ones. It is possible to jump forward (down) between database controls using the "TAB" key and backward (up) using the "SHIFT + TAB" key combination.
- Database controls include:
    - **TextBox** – a text field of any width and height intended for entering plain text with a maximum length of 250 characters for the Firebird database and with unlimited length for the MSSQL database, without the possibility of inserting a forced transition to a new line – enter. When the width is set to 0 pixels, the text box does not display the typical border, and the text editing option is also missing – the value of the text box is displayed as a regular paragraph with plain text. A detailed description of the control settings is given in the separate manual "TextBox – Control in the edit form". –

- **TextArea** – a text field of any width and height intended for entering plain text with unlimited length with the possibility of inserting a forced transition to a new line – enter. When the width is set to 0 pixels, the text box does not display the typical border, and the text editing option is also missing – the value of the text box is displayed as a regular paragraph with plain text. A detailed description of the control settings is given in the separate manual "TextArea – Control in the edit form".
- **RichTextBox** – a text box of any width and height for entering formatted text of unlimited length. A detailed description of the control settings is given in the separate manual "RichTextBox – Control in the edit form".
- **ComboBox** – a one-line drop-down list of any width with the option of selecting just one value from the list of available values, or with the option of entering your own value similar to a text field. The selected value can have a maximum length of 250 characters for the Firebird database and an unlimited length for the MSSQL database. A detailed description of the control settings is given in the separate manual "ComboBox – Control in the edit form".
- **ListBox** – a multi-line list of any width and height with the option to select just one value from a list of available values. The selected value can have a maximum length of 250 characters for the Firebird database and an unlimited length for the MSSQL database. A detailed description of the control settings is given in the separate manual "ListBox – Control in the edit form".
- **MultiListBox** – a multi-line list of any width and height with the option to select multiple values from a list of available values. The selected values are stored in the database separated by a tab in the form of a text string of unlimited length. A detailed description of the control settings is given in the separate manual "MultiListBox – Control in the edit form".
- **Radio** – a switch with the option to select just one value from the list of available values. The selected value can have a maximum length of 250 characters for the Firebird database and an unlimited length for the MSSQL database. The switch can display the available values either vertically below each other or horizontally next to each other. A detailed description of the control settings is given in a separate manual "Radio – Control in the edit form".
- **CheckBox** – check box, which, when checked, saves a predetermined text value to the database with a maximum length of 250 characters for the Firebird database and with an unlimited length for the MSSQL database. If unchecked, stores an empty text string in the database. A detailed description of the control settings is given in the separate manual "CheckBox – Control in the edit form".
- **ForeignKey** – a one-line drop-down list of any width with the option to select just one value from the list of available values, or a text field displaying exactly one value from the list of available values. ForeignKey uses the database type "int", and stores the primary key (ID) of the selected database record in the database. A detailed description of the control settings is given in the separate manual "ForeignKey – Control in the edit form".
- **Fillet** – a control for inserting a file attachment using the "Browse tlačítka" button. The maximum size of an embedded file attachment can be limited to a predetermined size. The file uses the "int" database type, and stores in the database the ID of the file attachment, which is physically stored on disk in the "Files" directory, and has descriptive data with the file name and size stored in a separate entry in the "sfiles" database table. A detailed description of the control settings is given in a separate manual "File – Control in the edit form".
- **Image** – a control designed to insert an image either with the "Browse…" button or with the "CTRL + V" key from the clipboard. The maximum size of the inserted image can be limited to a predetermined size. Image uses the database type "int", and stores in the database the ID of the file attachment, which is physically stored on disk in the "Files" directory, and has descriptive data with the file name and size stored in a separate entry in the database table "sfiles". A detailed description of the control settings is given in the separate manual "Image – Control in the edit form".
- **DatePicker** – date picker intended for entering the date using drop-down lists with day, month and year, or with a text field for entering the time. DatePicker uses the "datetime" database type. A detailed description of the control settings is given in the separate manual "DatePicker – Control in the edit form".

- **Schedule** – a control designed to set the time period from-to, which consists of either the text boxes “From” and “To” and the check box “From-To”, or the date picker “From” and “To” and the check box “From to”. A detailed description of the control settings is given in the separate manual “Schedule – Control in the edit form”.
- **Watcher** – a control designed to monitor the date of creation of the record and the date of the last change in the record. It consists of four text boxes “Entered when”, “Entered by”, “Changed when” and “Changed by”. A detailed description of the control settings is given in the separate manual “Watcher – Control in the edit form”.
- **AutoSum** – a one-line text field of any width, which is used to automatically evaluate the aggregation function on predefined data from the database. These aggregation functions include number, sum, average, maximum, and minimum. A detailed description of the control settings is given in a separate manual “AutoSum – Control in the edit form”.

- Visual controls include:
  - **Title** – style heading 'H1' or 'H2'. A detailed description of the control settings is given in the separate manual “Title – Control in the edit form”.
  - **Tab** – bookmark. A detailed description of the control settings is given in the separate manual “Tab – Control in the edit form”.
  - **RichText** – formatted text. A detailed description of the control settings is given in a separate manual “RichText – Control in the edit form”.
  - **DataGrid** – view table with the option search and filtering of records, and with the possibility of data exports and imports. A detailed description of the control settings is given in the separate manual “DataGrid – Control in the edit form”.
  - **LiteDataGrid** – view table without the possibility of personalization, search and filter records, and without the ability to export and import data. A detailed description of the control settings is given in the separate manual “LiteDataGrid – Control in the edit form”.
  - **History** – history of changes. A detailed description of the control settings is given in a separate “History” manual.
  - **Calendar** – classic calendar ala Outlook. A detailed description of the control settings is given in the separate manual “Calendar – Control in the edit form”.
  - **TimeTable** – timetable/gantt chart. A detailed description of the control settings is given in a separate manual “TimeTable – Control in the edit form”.
  - **Planner** – multi-source planning calendar. A detailed description of the control settings is given in the separate manual “Planner – Control in the edit form”.
  - **Chart** – graph. A detailed description of the control settings is given in a separate manual “Chart – Control in the edit form”.
  - **Map** – map based on OpenStreetMap maps. A detailed description of the control settings is given in a separate manual “Map – Control in the edit form”.
  - **Tree** – tree structure. A detailed description of the control settings is given in a separate manual “Tree – Control in the edit form”.
  - **Button** – button for call javascript functions in a web browser, scripts on the server side, printing database records to print templates, redirecting the user to a specific edit form or to a specific view page. A detailed description of the control settings is given in the separate “Button” manual.
  - **HTML** – HTML code. A detailed description of the control settings is given in the separate manual “HTML – Control in the edit form”.
  - **JavaScript** – javascript code. A detailed description of the control settings is given in the separate manual “JavaScript – Control in the edit form”.

- **Space** – graphic separator with or without horizontal line. A detailed description of the control settings is given in the separate manual “Space – Control in the edit form”.

- The controls of the edit form can be accessed using javascript, and thus change their properties and behavior. In the case of database controls, it is possible to change the values of text fields, drop-down lists, etc. provided we know the reference to the control in the HTML object model of the document.

6.2 Authorization to enter the edit form

- The edit form can be accessed only using the navigation elements located on the view pages or in the edit forms.
- For the purpose of creating a new record in the database, the user can enter the edit form in the following ways:
  - Using a button located on the view page or in the edit form, which redirects the user to a specific edit form. This method is defined directly in the properties of the “Button” control or in its script using the “SCHEDULEFORM” function.
  - Using a look-up table that redirects the user to the edit form whose data it displays. This method is defined directly in the properties of the look-up table by enabling the display of the “New record icon”.
  - Using other visual controls such as a classic calendar, a multi-source planning calendar, a timetable, a Gantt chart, or a tree structure. This method is defined directly in the properties of the visual control by allowing new records to be entered.
  - Using a view page that redirects users to a specific edit form. This method is defined directly in the properties of the preview page.
- For the purpose of editing a specific existing record in the database, the user can enter the edit form using the following options:
  - Using a look-up table that displays a pencil icon at the beginning of each row with the function of opening the displayed database record.
  - Using other visual controls such as a classic calendar, a multi-source planning calendar, a timetable, a Gantt chart, a map, or a tree structure. These controls either display the pencil icon or allow you to open a database record by clicking on its displayed details.
  - Using a button located on the view page or in the edit form, which redirects the user to a specific database record of the edit form. This method is defined in the script of the “Button” control using the “SCHEDULEFORM” function.
  - Using a preview page that redirects the user to the first record of the edit form. This method is defined directly in the properties of the preview page.

6.3 Permission to view the contents of the edit form

- The edit form alone does not have the option of setting the rights to display the form. The rights to display the content of the edit form is governed by the rights settings of its internal controls. The edit form is displayed when it contains at least one control to which the currently logged in user has the right. Otherwise, only the login window is displayed in the edit form as in the main page.
- Each control of the edit form has the option of setting user groups or specific users who have the rights to the control. Other users do not see the control in the edit form, and do not have the option to fill in its value if it is a database control.
- If the user creates a new record in the database via an edit form, the database value “null” will be filled in all database columns to which he is not entitled.

6.4 Ownership of records

- Each database table contains a column “userid” with the user ID, which is the owner of the database record, respectively. created a record in the database.
- Database records that have been imported or created by an external application may not have the “userid” column value populated. This means that such records do not yet have an owner. The owner then becomes the owner who opens the record in the edit form and saves it first.

6.5 Co-ownership of records

- In addition to its owner, database records can also have several co-owners. These can be both individual users and user groups. The co-owner information is not set for each database record separately as well as for the owner of the record, but is set for the relevant database table (edit form) as a whole.
- Ownership or at least co-ownership of a record is required when editing an existing record in the database and trying to save it in the edit form. Therefore, if the user is not the owner or at least a co-owner of an existing record, he will not be able to change such a record in the database.
- Co-ownership of records does not affect the creation of new records. The owner of the newly created record always becomes the user who saved the record in the edit form.

6.6 Permission to display the value of a control in an edit form

- The visibility of a database control value can be controlled by ownership or co-ownership of a record at the level of individual database records.
- The rights to display the value of a database control is set directly in the properties of the control.
- The control can be set in two different ways:
    - To all users (default setting)
    - Only to the owner or co-owner of the record. In this case, only the owner or co-owner of the database record has permission to view the value. Other users do not see the control in the edit form, and do not have the ability to view or fill in its value.

6.7 Authorization to save the record in the edit form

- The rights to save the database record is conditioned by the rights to edit, resp. ownership or co-ownership of the record. The rights to edit is set directly in the properties of the edit form.
- The edit form can be set up in two different ways:
    - The owner or co-owner of the record has permission to edit
    - Only the co-owner of the record has permission to edit (default setting)

6.8 Authorization to delete an entry in the edit form

- The rights to delete the database record is conditioned by the rights to edit, resp. ownership or co-ownership of the record, and at the same time authorization to all database controls of the edit form.
- The user can delete an entry in the edit form,
    - if he has permission to edit the database record and at the same time

  - has the rights to all database controls of the edit form.

- The user cannot delete an entry in the edit form,
  - if it does not have permission to edit the database record or
  - if there is at least one database control in the edit form to which it does not have the right.

6.9 Identifiers of edit form controls

- The identifier is a reference to the database control of the edit form within the HTML object model of the document, and also specifies the name of the database column in which the values entered through the control in the edit form are stored in the database.
- The identifier is derived from the name of the control. It starts with the prefix “ng_”, and also contains the alphanumeric characters used in the control name (system controls do not include the prefix “ng_”).
- The control identifier is available on the “Administration” tab directly in the control's properties.
- Using the identifier, it is possible to access the database control in javascript, and change the properties, behavior or value of the control.
- It is also possible to access the value stored in the database control using an identifier
  - in javascript in a web browser,
  - in a server-side script,
  - in conditions or joints of database queries,
  - in the “HTML” control
  - in print templates.

6.10 Locking records

- Database records opened in the edit form can be set so that only one user can edit them at a time.
- Database record locks are defined directly in the edit form settings by entering the time interval in minutes during which the lock is to be active. After this interval, the lock is automatically released.
- The user who opens the database record first automatically locks the record. While working in the edit form, the lock automatically extends until the user leaves the edit form. When you exit the edit form, the lock is automatically released.
- Locked records have a lock icon in the lookup table instead of the magnifying glass icon used to open the database record. The lock icon contains a tooltip with information on who locked the record and when the lock is active. Even so, it is possible to open the database record using the lock icon; for other users, the edit form with the locked record is displayed read-only without the possibility of editing.

6.11 History and tracking changes

- Each edit form has an associated additional database table inside the database, in which the changes made by the user in the individual database records are stored. The name of this database table is derived from the name of the edit form, starting with the prefix “ng_”, and ending with the postfix “_history”. The process of creating or modifying this database table is, as with the main one, completely automatic.
- The database structure of the history table is the same as for the main table. In addition, it contains the following columns:
  - “Date” – date of change

- "Userid" – login name of the user who made the change
- "Action" – ID of the script used to make the change
- "Changedid" – ID of the record to which the change relates
- "Created" – value "x" for creating a new database record
- "Deleted" – value "x" for deleting a database record

- Creating a new record in the main database table ensures the creation of a single record in the history table, where all column values are filled in the same way as for the main table. In addition, the values of the "date", "userid", "action", "changedid" and "created" columns are filled in.
- Editing an existing record in the main database table ensures that one record is created in the history table, where only the values of the columns that are being changed are filled. In addition, the values of the "date", "userid", "action" and "changedid" columns are filled in. For other columns, the database value "null" will be filled in the database.
- Deleting an existing record in the main database table ensures the creation of a single record in the history table, where all column values are filled in the same way as for the main table, just after deleting the record. In addition, the values of the "date", "userid", "action", "changedid" and "deleted" columns are filled in.
- By default, log history logging is turned on, but you can turn log history logging off in the edit form settings as needed.

6.12 Synchronization of database tables

- Synchronization is used to maintain a true copy of a database table in several other databases on remote NET Genium installations at the same time, e.g. on another server. It is a symmetric synchronization of data (multiple master), where a change in one copy of the database triggers a request to make the same change in other copies. Requests are queued on the server that initiated the change and executed in the same order on remote servers.
- Data synchronization can be one-way or two-way. One-way synchronization transfers data from the source database to the destination in only one direction. In order to ensure data integrity during one-way synchronization, there must be no record changes in the target database, usually triggered by NET Genium users. If changes also occur in the target database, bidirectional data synchronization must be selected.
- A detailed description of the synchronization settings is given in the "Edit form" manual.

7 View pages

- The view page is a graphical interface that is used to visualize, view, filter and search database records.
- A detailed description of the preview page settings is given in the separate “Preview page” manual.

7.1 Preview page controls

- The view page consists of visual controls that are arranged one below the other.
- Visual controls include:
  - **TextBox** – a one-line text field of any width intended for entering plain text without the possibility of inserting a forced transition to a new line – enter. A detailed description of the control settings is given in the separate manual “TextBox – Control on the view page”.
  - **RichTextBox** – a text box of any width and height for entering formatted text. A detailed description of the control settings is given in the separate manual “RichTextBox – Control on the view page”.
  - **Title** – style heading 'H1' or 'H2'. A detailed description of the control settings is given in the separate manual “Title – Control on the view page”.
  - **Tab** – bookmark. A detailed description of the control settings is given in the separate manual “Tab – Control on the view page”.
  - **RichText** – formatted text. A detailed description of the control settings is given in the separate manual “RichText – Control on the view page”.
  - **DataGrid** – view table with the possibility of searching and filtering records, and with the possibility of exporting and importing data. A detailed description of the control settings is given in the separate manual “DataGrid – Control on the view page”.
  - **DataSet** – multi-source view table with the ability to search and filter records. A detailed description of the control settings is given in the separate manual “DataSet – Control on the view page”.
  - **History** – history of changes. A detailed description of the control settings is given in the separate manual “History – Control on the view page”.
  - **Calendar** – classic calendar ala Outlook. A detailed description of the control settings is given in the separate manual “Calendar – Control on the view page”.
  - **TimeTable** – timetable/gantt chart. A detailed description of the control settings is given in the separate manual “TimeTable – Control on the view page”.
  - **Planner** – multi-source planning calendar. A detailed description of the control settings is given in the separate manual “Planner – Control on the view page”.
  - **Chart** – graph. A detailed description of the control settings is given in a separate manual “Chart – Control on the view page”.
  - **Map** – map based on OpenStreetMap map data. A detailed description of the control settings is given in the separate manual “Map – Control on the view page”.
  - **RDF/RSS News** – RSS feed. A detailed description of the control settings is given in the separate manual “DRF/RSS News – Control on the view page”.
  - **Tree** – tree structure. A detailed description of the control settings is given in the separate manual “Tree – Control on the view page”.
  - **E-mail** – A simple inbox email client that displays (but does not download or store to the database) incoming email messages from a POP3 or IMAP mail server. A detailed description of the control settings is given in the separate manual “E-Mail – Control on the view page”.

- **Button** – a button designed to call javascript functions in a web browser, server-side scripts, print database records to print templates, redirect the user to a specific edit form or to a specific view page. A detailed description of the control settings is given in the separate manual “Button – Control on the view page”.
- **HTML** – HTML code. A detailed description of the control settings is given in the separate manual “HTML – Control on the view page”.
- **JavaScript** – javascript code. A detailed description of the control settings is given in the separate manual “JavaScript – Control on the view page”.
- **Space** – graphic separator with or without horizontal line. A detailed description of the control settings is given in the separate manual “Space – Control on the view page”.

- The view page is also an intermediary for entering the edit form – either for the purpose of creating a new record in the database, or for the purpose of editing a specific existing record.
  - For the purpose of creating a new record in the database, the user can enter the edit form from the view page in the following ways:
    - Using a button that redirects the user to a specific edit form. This method is defined directly in the properties of the “Button” control or in its script using the “SCHEDULEFORM” function.
    - Using a look-up table that redirects the user to the edit form whose data it displays. This method is defined directly in the properties of the look-up table by enabling the display of the “New record icon”.
    - Using other visual controls such as a classic calendar, a multi-source planning calendar, a timetable, a Gantt chart, or a tree structure. This method is defined directly in the properties of the visual control by allowing new records to be entered.
    - Using a view page that redirects users to a specific edit form. This method is defined directly in the properties of the preview page.
  - For the purpose of editing a specific existing record in the database, the user can enter the edit form using the following options:
    - Using a look-up table that displays a pencil icon at the beginning of each row with the function of opening the displayed database record.
    - Using other visual controls such as a classic calendar, a multi-source planning calendar, a timetable, a Gantt chart, a map, or a tree structure. These controls either display the pencil icon or allow you to open a database record by clicking on its displayed details.
    - Using a button that redirects the user to a specific database record of the edit form. This method is defined in the script of the “Button” control using the “SCHEDULEFORM” function.
    - Using a preview page that redirects the user to the first record of the edit form. This method is defined directly in the properties of the preview page.

7.2 Permission to access the view page

- The view page can be accessed from the navigation area with a list of available applications or favourites, as well as using the navigation elements located on the main page, on the view pages or in the edit forms.
- Each view page has the option of setting up user groups or specific users who have the rights to the view page. Other users do not see the view page in the navigation area and are not able to access it.

7.3 Permission to view the content of the view page

- The preview page is displayed when it contains at least one control to which the currently logged in user has the right. Otherwise, only the login window is displayed on the view page as it is on the main page.
- Each view page control has the ability to set up user groups or specific users who have the rights to the control. Other users do not see the control on the preview page.

8 Category

- Landing pages can be grouped into categories for clarity. Categories usually contain view pages with the same or similar focus.
- Categories are defined within the application, and are displayed in the navigation area at the same level as the view pages located directly in the application, i.e. at the same level as view pages that are not categorized.
- Categories appear first in front of view pages, similar to the directory structure on disk for directories and files.
- A detailed description of the category settings is given in the separate “Categories” manual.

9 Portlets

- Portlets are simplified visual controls displayed on the main page, also called a dashboard, and allow you to clearly display different information on a single screen.
- Portlets can be displayed in one, two, or three columns – each NET Genium user has the option to set the portlets at their discretion.
- Portlets display simplified variants of the following controls, located on view pages:
    - **RichText** – formatted text
    - **DataGrid** – view table
    - **Chart** – graph
    - **E-mail** – simple e-mail client
    - **HTML** – HTML code
- Lookup tables or graphs displayed in portlets should respect the limited width of the screen due to the ability to arrange portlets into three columns.
- The default portlet settings are defined by the user with the highest authority – the “Administrator” user with database ID 1. Especially for regular users, and separately for anonymous users.
- A normal user who logs in to NET Genium and does not have portlets set up automatically takes over the portlet settings from the “Administrator” user, resp. those to which he is entitled. You can further customize portlets on your own.
- Anonymous users and users in the “External Users” group do not have the rights to customize their portlets. Only users belonging to the Administrators group can change this setting.
- A detailed description of portlet settings is provided in the separate “Portlets” guide.

10 Favourite items

- Favourites are links to frequently used view pages, and appear in the navigation area so that they are available at any time.
- The default settings of favourites are defined by the user with the highest authorization – the “Administrator” user with database ID 1. Especially for regular users, and separately for anonymous users.
- A regular user who logs in to NET Genium and does not have set favourites, automatically takes over the settings of favourites from the user “Administrator”, resp. those to which he is entitled. He can further customize his favourites completely independently.
- Anonymous users and users in the “External Users” group do not have the rights to customize their favourites. Only users belonging to the Administrators group can change this setting.
- A detailed description of favourite settings is given in the separate “Favourites” manual.

11 Picker

- Picker is a dialog box that displays a look-up table with database records, the values of which can be taken or copied to the currently open edit form. The picker is typically used to copy entire database records or to retrieve a ForeignKey value.
- The link to the pickup dialog box with the lookup table and database records is in the form of a magnifying glass icon, and appears in the edit form to the right of the control to which the picker is associated.
- The picker also takes the form of an ajax whisperer, which starts offering database records as soon as you enter a value in a text field in the edit form. For ajax whispering to work, you must enable searching for values entered in the edit form.
- The picker settings are made in a separate dialog. The link to this dialog is in the form of a magnifying glass icon, and is located in the lower left corner of the control settings dialog to which the picker is associated.
- A detailed description of the picker settings is included in the manual of each database control.

12 Database query

- A database query defines the source of data retrieved from the database, which is then used or displayed in controls on edit forms and view pages, or in scripts.
- The database query determines which records will be displayed in view tables, portlets, calendars, graphs, what values will be displayed in drop-down lists in the edit form, etc.
- The link to the database query is always part of the dialog for setting a specific control, most often used with the name “Data source” or “From query”. Click on the link to display the graphical designer of database queries – “QueryBuilder”.
- A database query is defined:
    - The database table from which the records are read
    - By choosing whether it is a regular database query or an aggregate query that uses grouping
    - The “Sort by” column, by which the database records are sorted, including the sorting method – ascending (ASC) or descending (DESC)
    - Optionally the second column according to which the database records are sorted, including the sorting method – ascending (ASC) or descending (DESC)
    - The “Color by” column, which determines whether and by which column a color rectangle will appear for each record in the lookup table
    - The “Time period by” column, which determines whether and according to which column a filter for selecting the “from-to” time period will be displayed above the look-up table.
    - List of restrictive conditions
    - List of accepted database tables
    - In the case of an aggregation query, it is further defined:
        - The list of columns of the resulting aggregation table
        - A list of headers that appear above the columns in the lookup table
    - In the case of a graph, it is further defined by a list of colors, which will be used to plot the individual columns of the graph
    - By selecting the column according to which duplicate rows in the loaded data set will be evaluated, and these rows will then be removed from this set
- The resulting SQL query composed of “SELECT”, “FROM”, “JOIN”, “WHERE”, “ORDER BY” and “GROUP BY” clauses is always compiled automatically according to the query parameters set in the graphical database query designer.
- The result of a database query is always a set of data retrieved from the database, temporarily stored in an object of type “DataTable”. This data set is then passed to the individual controls for evaluation or visualization. Before passing the “DataTable” object to the control, an external function can be run, which has the option to change the properties of this object – add rows, change values in individual columns, or delete rows.
- A detailed description of the database query designer is provided in the separate “Database Query Designer” guide.

12.1 Common question

- A common query is used for all controls that use data retrieved from the database as the data source.
- A common query is the default setting in the database query designer, and uses only the “SELECT”, “FROM”, “JOIN”, “WHERE”, and “ORDER BY” clauses to build the resulting SQL query. Therefore, it does not use the “GROUP BY” clause.

12.2 Aggregation query

- The aggregation query is used for look-up tables and graphs that display aggregated data retrieved from a database using the “GROUP BY” clause.
- The aggregation query is defined in the database query designer:
  - by checking the box “Aggregation query”
  - a list of query columns according to which the records are grouped – those that appear in the resulting SQL query in the “GROUP BY” clause. Supported types of grouping include:
    - Group by – normal “GROUP BY ng_”
    - Group by hour of the day
    - Group by hour
    - Group by day
    - Group by week
    - Group by month
    - Group by quarter
    - Group by year
  - a list of aggregated columns – those that appear in the resulting SQL query in conjunction with the functions “COUNT”, “SUM”, “AVG”, “MAX” and “MIN”, with the possibility of setting the condition using the “CASE WHEN” clauses. Supported aggregate column types include:
    - COUNT(*)
    - Amount – SUM(ng_)
    - Average – “AVG(ng_)”
    - MAX – MAX(ng_)
    - MIN – “MIN(ng_)”
  - a list of informative columns – those that appear in the resulting SQL query as text or numeric constants, with the possibility of setting the condition using “CASE WHEN” clauses, then evaluated by a lexical analyzer. Supported types of information columns include:
    - Text – “MIN(ng_)”
    - Expression, format according to – “MIN(ng_)”, subsequently evaluated by the lexical analyzer according to the mathematical expression defined as part of the column name
  - a list of columns by which it will be possible to filter in the look-up table or graph
  - a list of headers into which the columns of the lookup table will be grouped
  - a list of colors that will be used in the chart to draw each column of the chart
- Each column of an aggregate query uses the following syntax for calling columns: “Name@mathematical_expression@displayed_value@condition”.
  - The name specifies the name of the column displayed in the lookup table or chart
  - A mathematical expression is an optional data relevant to “Expression, format by” columns, and defines the mathematical formula used to calculate the value displayed in a lookup table or graph.

- The displayed value is an optional data relevant for columns of the “Expression, format by” type, and allows you to hide the calculated value with a specific text string, or hide the entire column in the lookup table using the column name specified as “@@”.
- The condition is an optional data relevant for columns of the “Expression, format by” type, and allows you to display only the rows of the resulting data set that meet the specified condition.

- Examples of column names:
  - “Sum of first and second columns @#c0#+#c1#”
  - “Sum of first and second columns @#c0#+#c1#@Hidden value”
  - “Sum of value on next line@+1”
  - “Sum of value on previous line@-1”
  - “Sum of first and second columns and values on next row@+1#c0#+#c1#”
  - “Sum of values greater than 5@@@>5”
  - Display of the sum of the column in case the sum of the column is not displayed by default – “Column@+ sum”
  - Hide the sum of the column, which is displayed by default with the sum of the column – “Column@-sum”
- Each aggregate query header uses the following syntax for calling headers: “Number_of_merged_cells@header_name”.
- Examples of header names:
  - 1. line: “5@X;1@Y”
  - 2. line: “2@A;3@B;1@C”

X					**Y**
AND		**B**			**C**
Column 1	Column 2	Column 3	Column 4	Column 5	Column 6
…	…	…	…	…	…

12.3 Nested records

- A nested record is a record associated with its parent record using the “pid” and “pform” foreign keys.
- The foreign keys “pid” and “pform” are part of each database table, and for each database record they define the ID of the parent record (parent ID) and the ID of the edit form (parent form) in which the parent record is located.
- The foreign keys “pid” and “pform” most often refer to records in another database table – for example, a job item refers to a job ID.
- The foreign keys “pid” and “pform” can also refer to records in the same database table. This feature is used by the “Tree” control – a tree structure – which visualizes database records based on a recursive method.
- The use of foreign keys “pid” and “pform” is completely automatic. NET Genium automatically populates the value of these two columns when creating database records in the edit form according to where the record originated – from which record the user was redirected to the target edit form in order to create a new record in the database.

- If the user is redirected to the edit form from the view page, the newly saved record will have the value “pid” and “pform” set to “0”.
- If the user is redirected to the edit form from an existing record in the database, the newly saved record will have the value “pid” and “pform” set according to the record ID and the edit form of this parent record.

- From the point of view of the application administrator, the use of the foreign key “pform” is minimal. Only in rare cases is the value of the “pform” column used – when one database table with nested records serves two or more parent tables.
- Foreign “pid” and “pform” keys do not have indexing turned on by default. Whenever the application administrator decides to use these foreign keys, it is necessary to turn on indexing of these columns directly in the settings of the edit form.
- The lookup tables in the edit forms include a “Show only nested records” check box, which simplifies the use of nested records, and automatically appends the “PID = record ID in form” constraint to the database query definition.
- The lookup tables on the lookup pages include a “Hide nested records” check box, which makes it easier to use nested records, and automatically appends the “PID = 0” constraint to the database query definition.

12.4 Query constraints

- A database query can contain any number of constraints that follow a “WHERE” clause. The individual conditions are separated from each other using the operators “AND” and “OR”, including the possibility of using left and right parentheses.
- The resulting form of the query condition in SQL is compiled automatically according to the list of conditions set in the graphical designer. A detailed description of the database query designer is provided in the separate “Database Query Designer” guide.
- The following extension functions can be used in a query condition:
  - The data type “Integer” and “Long” allows writing in the form “Left side of the condition equals 1;2;3”. The resulting form of the condition will then be interpreted as “Left side of condition IN (1;2;3)”.
  - The data type “String” allows writing in the form “The left side of the condition equals (array)A;B;C”. The resulting form of the condition will then be interpreted as “Left side of condition IN ('A','B','C')”.
  - The left side of the condition allows you to select the “MultiListBox” control(tab-separated values) along with a text constant on the right side of the condition. The resulting form of the condition will then be interpreted as “JoinText2(Left side of the condition, Right side of the condition)” for the Firebird database or “dbo.JoinNtext(Left side of the condition, Right side of the condition)” for the MSSQL database. The SQL query then evaluates all records that have at least one value stored in the “MultiListBox” column on the left side of the condition, which is equal to the text constant on the right side of the condition.
  - The left side of the condition allows you to select the “MultiListBox” control(tab-separated values) along with the “MultiListBox” control on the right side of the condition. The resulting form of the condition will then be interpreted as “JoinText2 (Left side of the condition, Right side of the condition)” for the Firebird database or “dbo.JoinNtext(Left side of the condition, Right side of the condition)” for the MSSQL database. The SQL query then evaluates all records that have at least one value stored in the “MultiListBox” column on the left side of the condition, which is stored in the “MultiListBox” column on the right side of the condition.
  - The data type “String” allows writing in the form “Left side of the condition equals(mlb)A#tab#B#tab#C”. The resulting form of the condition will then be interpreted as “JoinText2(Left side of the condition, 'A B C')” for the Firebird database or “dbo.JoinNtext(Left side of the condition, 'A B C')” for the MSSQL database. The SQL query then

evaluates all records that have at least one value stored in the tab-delimited column on the left side of the condition that is part of the text constant on the right side of the condition.

- A query condition can also be defined using SQL syntax that contains both the left and right sides of the condition. The left side of the condition, including the operator, can be arbitrary, the right side of the condition must begin with the word “OK#crlf#”, and must be followed by the SQL wording of the condition, for example:
    - OK#crlf#0=0
    - OK#crlf#0=1
    - OK#crlf#EQUALS(ng_tb, "", 0=0, ng_tb = FORMATSTRINGSQL(#ng_tb#))
    - OK#crlf#EQUALS(ng_tb, "", 0=1, ng_tb = FORMATSTRINGSQL(#ng_tb#))

12.5 Joins

- A database query can contain any number connections that are built using the “JOIN” clause.
- The resulting form of the connection in SQL is compiled automatically according to the list of connections set in the graphic designer. A detailed description of the database query designer is provided in the separate “Database Query Designer” guide.
- The connection can also be defined using SQL syntax. The left side of the connection condition can be arbitrary, the right side of the connection condition must begin with the word “OK#crlf#”, and must be followed by the wording of the SQL connection, for example:
    - OK#crlf#INNER JOIN ng_abc J1 ON J1.pid = ng_form.id
    - The connection definition must respect the aliases used, so the first connection must be named J1, the second J2, and so on.
    - The connection definition must respect the name of the source database table.

13 Variables and identifiers

- A variable is a text identifier that begins and ends with a “#” and stores information about the web application, database, currently logged in user, edit forms, view pages, or controls in general.
- Identifiers are replaced on the server side by specific values expressed by a text string.
- Variables include
    - General variables – such as “#loginname#” or “#today#”
    - Identifiers of database controls
        - Current value entered by the user in the edit form – “#ng_#”
        - Current value entered by the user in the edit form, queried from javascript code – “#ng_:value#”
        - The original historical value, resp. current value in the database – “#ng_:history#”
        - The value of the “MultiListBox” control, where the individual text items are converted to their “id” in the database – “#ng_:ids#”
    - Javascript variables – for example “nID”
    - Script variables – such as “#A#”
    - Variables used in print templates – such as “#ng_#”, “#c0” or “#remove#”
- Variables can only be used by administrators when creating applications. A normal user does not have access to the values of variables, resp. the variable is used only indirectly by applications designed by administrators.
- The identifiers of the database controls in the edit form behave differently inside the javascript code.
    - The use of the variable “#ng_#” is used to access the database control object inside the HTML object model of the edit form document – in this case the variable is replaced by “document.getElementById()” or “document.forms[0].” Depending on used web browser.
    - The value of the database control currently stored in the database is accessed from the javascript code using the variable “#ng_:value#”.
    - If the user does not have the rights to the control associated with the variable, the variable is replaced with the value stored in this control in the database. Therefore, a variable can be replaced by an empty value, which can cause a syntax error in the javascript code, for example “var o = ;”.
- A detailed description of the variables is given in a separate manual “Variables”.

14 Server functions

- Server functions are used to call server-side program code that returns a value expressed by a text string.
- Server functions use parameters enclosed in parentheses, separated by a comma. Parameters can be enclosed in double quotes, for example, in a situation where one of the parameters contains a separator character, or begins or ends with a space. Functions that have no parameters must still contain parentheses when called.
- Server functions can only be used by administrators when creating applications. A normal user does not have access to the call of server functions from the application environment, resp. server functions are called only indirectly using applications designed by administrators.
- A detailed description of the server functions is given in the separate “Server Functions” manual.

15 External functions

- The external function is used to call its own C# program code on the server side, which returns a value expressed by a text string.
- To call external functions, use the server function “ngef(string id, string arg0, string arg1, string arg2,…)”, which contains the required parameter “id” with the identifier of the external function, as well as any number of other optional parameters that are external to functions passed as “string[] args”.
- External functions can only be used by administrators when creating applications. A normal user does not have access to the call of external functions from the application environment, resp. external functions are called only indirectly using applications designed by administrators.
- A detailed description of programming external functions is given in a separate manual “External functions”.

16 Script

16.1 Basic information

- A script is program code that runs on the server side
  - in server events of edit forms,
    - **OnBeforeOpen** – script that will be executed before opening the edit form after clicking on the “Save and new” button
    - **OnAfterOpen** – script that will be executed after opening the edit form
    - **OnBeforeSave** – a script to be executed before saving the record to the database
    - **OnAfterSave** – a script that will be executed after saving the record to the database
    - **OnBeforeDelete** – a script to be executed before deleting a record from the database
    - **OnAfterDelete** – a script that will be executed after deleting a record from the database
  - after pressing the button located in the edit form,
  - after pressing the button located on the view page or
  - when trying to create a new record from the “TimeTable” and “Planner” controls.
- The script link is always part of the dialog for setting a specific control – most often “ActionButtons” in the case of server events of edit forms, or “Button” in the case of buttons located in edit forms or on view pages – used called “Edit script (number of script lines)) ”. Click on the link to display it graphic script designer.
- The script consists of individual commands, which are inserted below each other in the form of numbered lines of the script. Each line of the script contains an instruction on which command to execute. NET Genium goes line by line, and executes the appropriate commands depending on the instructions.
- The basic script commands consist of:
  - Variable declarations
    - The command sets the data type of the script variable.
    - If the variable contains a value, or if the variable has already been declared, the data type of the variable is changed and its value is set to the database value “null”.
  - Assign a value from a query to a variable or control on a form
    - The command reads a value from a database query, and stores it either in a script variable or in a database control if the script is inside an edit form.
    - If the variable contains a value, or if the variable has already been declared, the data type of the variable will be changed and its value will be overwritten.
  - Save the variable back to the database
    - The command saves the array of variable values back to the database from where it was retrieved.
    - The variable must not change the number of its values during the script, e.g. by grouping using the “GROUP(#A#)” function.
  - Assign multiple values from a query to variables or controls on a form

- The command retrieves the values of multiple columns from a database query, and stores them either in a script variable or in database controls if the script is inside an edit form.

- Cycle
  - The command executes the cycle at the end bounded by the “Next…” command as many times as the selected element control variable has. If the variable contains 0 elements, the cycle does not run even once, and the script continues after the “Next…” command.
  - The meaning of the cycle is calculated by traversing the individual elements (even duplications) in the field of values of the control variable. In each iteration of the cycle, the control variable behaves as a single-element array, resp. as one value. At the end of the cycle, it becomes the original variable containing the field of values again.
  - If a variable contains a database value “null” in a certain iteration of the cycle (the value is not defined), the cycle skips to the next possible iteration, when the value of the variable is already defined.
  - If the cycle control variable is loaded together with other variables with the “Assign multiple values from query” command, these variables behave the same as the control variable during cycle traversal. For example, when reading variables “A” and “B” at the same time, it does not matter which of these two variables will be used as the master. The selected control variable should not only contain elements with a database value of “null”, therefore it is recommended to use the primary key “id” as a control variable.
- Assign a value to a variable or control on a form
  - The command assigns a value to a script variable or database control. The command allows you to assign a single value, or a mathematical expression – the sum, difference, multiple or proportion of two values.
  - If the variable is an array of values – for example, the command “[A▼] = [A▼] [+▼] [1]“ – numerical operation is performed on all of its elements.
- Comparison of the values of two variables with each other or comparison of the value entered by the user in the edit form with the variable
- Jump to the line
- Invoking an interrupt
- Sending an e-mail
- Delete a record from the database
  - The command deletes all records that match the conditions of the database query.
- Redirect the user to the edit form to create a new record
  - The command schedules the user to be redirected to the currently open edit form in order to create a new record. The redirection itself will occur until the entire script is successfully completed.
  - The command is only available in the “OnAfterSave”, “OnBeforeOpen” or “OnAfterDelete” script.
- Create a new record in the database
  - The command creates one or more new records in any database table. A database query is used to create several records at once. It tells you how many records will be created and makes the values retrieved from the source database table available when you create them. It goes through the record after the record and creates new records from the values obtained in this way.

        - The last record created by this command in the “OnBeforeSave” script has its unique “id” record stored in the “#nid#” variable, which can be used in the subsequent “OnAfterSave” script.
    - Save the value to the database
        - The command writes the selected value to any place in the database defined by the database query. If the database query results in multiple records, the value will be stored in all of those records in the database.
    - Store multiple values in a database
        - The command writes the selected values to any place in the database defined by the database query. If the database query results in multiple records, the values will be stored in all of those records in the database.
    - Comment
        - The command is used for internal notes of the author of the script, or for calling external functions.
- In the script it is possible to use:
    - Script variables used to store temporary values – such as “#A#”
    - Text variables that use identifiers to refer to control values entered by the user in an edit form – for example, “#ng_#”
    - Text variables that use identifiers to refer to the current values of controls in the database – for example, “#ng_:history#”
    - NET Genium server function calls – for example “SUBSTRING(#A#, 0, 2)”
    - Calling external functions written in C# – for example “ngef(MyFirstFunction, #A#)”
- A script is a programming language that has its own syntax but does not support program blocks. Its operation is provided only by a set of the above-mentioned basic commands inserted into the script designer. Program blocks or more complex algorithms that cannot be serviced using basic commands in conjunction with variables and server functions must be programmed using external functions written in C#.
- The script assigned to the button on the view page can also be run externally directly from the MS Windows operating system using the “RunScript.exe” program. The “RunScript.exe” program is located on the server disk in the “NETGenium\bin” directory.
- Errors or interrupts can occur during script execution, either due to a syntax error in the script, errors in value conversions, when saving data to the database, or when programmatically invoking a script interrupt. In all cases, that means end the execution of the script, return to the edit form or to the view page from which the script was called, and display the error message to the user. For the “OnBeforeSave” and “OnBeforeDelete” scripts, such an interruption means the prevention of the planned save, resp. deleting a record.
- A detailed description of the script designer is given in the separate “Script Designer” manual.

16.2 New record vs existing record in the database

- When evaluating a script, it is often necessary to distinguish whether the user has an open edit form with a new record that has not yet been saved, or whether he is editing an existing record in the database. For these purposes, the primary key “id”, resp. variable “#id#”, which has the value “0” in the case of an unsaved record. A new “id” is generated only after the record has been successfully saved using the “OnBeforeSave” script. The “id” of the newly created record can be accessed in the “OnAfterSave” script, where this “id” is already known.

16.3 Script variables

- When working with script variables, it is always necessary to declare the given variable first – define the data type of the variable. Supported data types include:
    - String – a text string
    - Integer – integer
    - Long – large integer
    - Double – decimal number
    - Date – date
- You can declare a variable in the following ways:
    - Use a script command to declare a variable
    - Using a script command to retrieve a value from a query – the data type of a variable is automatically taken from the type of the database column from which the value is retrieved into the variable
    - Using the “String(abc)”, “Integer(1)”, “Long(1)”, “Double(1)” or “Date(#today#)” commands
- Variables that are retrieved from a query automatically become an array of values. These values can be browsed using the script command “Cycle” – interpretation of the command “foreach (var item in items)” used in C# – or evaluated using the functions “AVERAGE(#A#)”, “COUNT(#A#)”, “FIRST(#A#)”, “GROUP(#A#)”, “LAST(#A#)” and “SUM(#A#)”. If the result of the query is 0 records, it is an array with zero number of elements.
- Fields of values can also be declared manually using the following server functions:
    - “StringArray(semicolon-separated list of values)”
    - “IntegerArray”
    - “LongArray(number of values)”
    - “DoubleArray(semicolon-separated list of values)”
    - “DateArray(semicolon-separated list of values)”
- Even manually generated variables of the value field type can be traversed using the “Cycle” script command.
- Variables use a database value of “null”. When a basic declaration of a variable that is not an array of values, its value is always set to the database value “null”. At any time later, it is possible to store the database value “null” either by using the variable “#null#” or by assigning another variable that carries the database value “null” – for example, it has just been declared and has not yet been stored. It is possible to work with variables similarly to the values of database columns, and test them using the operators “is defined” and “not defined”.

16.4 Save the value of a variable to a database control

- The value of the variable can be assigned to the database control, e.g. after filling in the edit form by the user and then running the script using the button. If the variable is an array of values, only its first value in the sequence is assigned to the control. If the variable is not defined or its field contains 0 elements, the database value “null” is assigned to the control.

16.5 Comparing the values of variables

- It is possible to compare the values of two variables in a script. If one of the variables is an array of values, only the first values of those arrays are compared. It is also possible to verify if the value field contains a certain value. The “contains”

operator is used for this test. However, it is important not to confuse the function of this operator by using a test on a common variable declared manually and carrying a text string. This variable is not the result of a query, and therefore is not an array of values. The “contains” operator is then used to verify if a string contains a particular substring.

16.6 Convert values to another data type

- Writing a variable in the input field of a script command can be used to convert the value of a variable to the required data type.
- Example of converting a numeric value to a text string:
    - [A▼] = [String▼]
      [B▼] = [Integer▼]
      [B▼] = [2]
      [A▼] = [#B#]
- Example of converting a text string to a numeric value:
    - [A▼] = [Integer▼]
      [B▼] = [String▼]
      [B▼] = [2]
      [A▼] = [#B#]

17 Javascript

- Javascript is program code that runs on the client side in a web browser
  - from the “JavaScript” control in the edit form,
  - from the “JavaScript” control on the preview page
  - after pressing the button located in the edit form,
  - after pressing the button located on the view page or
  - when invoking a javascript event of database controls in the edit form – “OnBlur”, “OnChange”, “OnClick” and “OnFocus”.
- In javascript it is possible to use:
  - NET Genium text variables – for example “#today#”
  - Text variables that use identifiers to refer to the objects of the database controls of the edit form inside the object model of the HTML document – “#ng_#”
  - Text variables that use identifiers to refer to control values entered by the user in the edit form – “#ng_:value#”
  - Text variables that use identifiers to refer to the current values of controls in the database – “#ng_:history#”
  - Extended text variables that use identifiers to refer to a table with a control in an edit form – “#ng_:Table#”
    - The controls are arranged one below the other in the edit form, and are located in tables – <table>…</table>.
    - Each database control is located in a separate table that can be hidden or shown – for example #ng_:Table#.style.display = 'none';
  - Extended text variables that use identifiers to refer to the label with the name of the control in the edit form – “#ng_:Name#”
    - The names of the controls are displayed in the edit form to the left of the control itself, and are located in labels – <label>…</label>.
    - Each database control has its own label, which can be used to change the displayed value – for example #ng_:Name#.innerHTML = 'abc';
  - Calling javascript functions NET Genium – for example “control_Enable(#ng_#);”
  - NET Genium server function calls – for example “SQL(SELECT ng_ FROM ng_)” or “SQLARRAY(SELECT * FROM ng_)”
  - Calling external functions written in C# – for example “ngef(MyFirstFunction, #today#)”
- Javascript is most often used on
  - read or set the value of database controls,
  - validation of values entered by the user into database controls,
  - changing the properties or behavior of database controls
  - hiding or showing parts of the edit form,
  - retrieving or saving values from a database using ajax.
- A detailed description of javascript functions is given in a separate manual “Javascript functions”.

18 Printing

- The buttons located in the edit form or on the view page allow printing in the printing template. Supported print templates include “xls”, “xlsx”, “doc”, “docx”, “html”, “txt”, “csv” and “pdf”. The “xlsx” and “docx” print templates support automatic conversion to “pdf”.
- Print templates must be saved in the “NETGenium\Templates” directory. It is possible to copy files to this directory directly from the server desktop, or via the dialog for NET Genium settings on the “Print templates” tab. It is then possible to select the selected print template directly from these files in the button properties.
- Only database control identifiers and script variables can be used in print templates. Script variables can be filled in before printing using a script.
- Database control identifiers or variable names are inserted where the values need to be substituted in the template. NET Genium opens a print template, searches for occurrences of all identifiers, and replaces them with the appropriate values.
    - Substituting database control values into a print template from the currently open record in the edit form uses abbreviated identifiers in the format “#ng_control#”.
    - Printing lookup table entries requires the use of full-form identifiers in either format
        - “#ng_form#.#ng_control#” or
        - “#datagrid1#.#ng_control#”, “#datagrid2#.#ng_control#”, etc.
          “#calendar1#.#ng_control#”, “#calendar2#.#ng_control#”, etc.
          “#timetable1#. #ng_control#”, “#timetable2#.#ng_control#”, etc.
          “#planner1#.#ng_control#”, “#planner2#.#ng_control#”, etc.
    - A second full identifier format with the name of the visual control type and its sequence number is recommended because it takes into account situations where multiple view tables with the same data source are placed in the edit form or on the view page, with only different constraints.
    - NET Genium recognizes the occurrence of identifiers in full form, and ensures automatic insertion of the values of individual records below each other.
        - If the identifiers are placed side by side in one row, the print output is a table, where each table entry has its own row in the print template.
        - If the identifiers are placed side by side in two rows, the print output is a table, where each table entry consists of two rows in a print template.
    - Printing from a lookup page that contains only one lookup table does not require full-form identifiers, simplified form identifiers are sufficient.
    - Printing a statistical look-up table with aggregated columns uses column identifiers “c0”, “c1”, “c2”, etc., i.e. according to their order in the table header, numbered from zero.
- Print templates created in MS Word must have identifiers located in the “MergeField” field.
- The values in the print template can be formatted according to the templates using the syntax “#identifier@format#”.
- After the successful creation of a print report, the external function “NETGenium.Print” starts automatically, in which it is possible to influence both the name of the resulting file and the content or format of the file.
- A detailed description of programming printing templates in “pdf” format is given in a separate manual “Programming printing templates in PDF format”.

19 “Settings” application

- NET Genium contains a default set of edit forms and their view pages, which are absolutely necessary for the proper operation of the entire application. These edit forms are located in the “Settings” application.
- The “Settings” application cannot be deleted or moved to another application group.
- Edit forms and view pages in this application can be performed by only one user with the highest authorization – the “Administrator” user with database ID 1.
- The “Settings” application contains the following edit forms and their associated view pages:
    - **User** – database table “susers” with a list of users
        - Full name – the full name of the user
        - Last name – last name of the user
        - Name – user name
        - Title – the title of the user
        - Login name – user login name
        - Password – user password
        - Language – language settings of the user (cs, de, en, fr, sk)
        - Windows account – the name of the user's domain account in Active Directory, used to ensure automatic login to NET Genium without the need to enter a login name and password
        - Email – user's email address
        - M-Password – password for the account on the Android mobile device
        - Authorization group – a foreign key to the authorization group code list, which defines the list of user groups in which the user is a member. Assigning a user to a permission group is the preferred method of assigning rights, however, it is also possible to manually assign a list of user groups, see the next “User Groups” control.
        - User groups – list of user groups in which the user is a member. Manually assigning user groups to each user separately is not the preferred method of assigning rights; the recommended method is to select a permission group, see the previous control.
    - **User group** – database table “susergroups” with a list of user groups
        - User group – name of the user group
        - Description – description of the user group
    - **Authorization group** – database table “srightsgroups” with a list of authorization groups
        - Authorization group – name of the authorization group
        - Description – description of the permission group
        - User groups – a list of user groups that define a permission group
    - **Public holiday** – database table “sholiday” with a list of current public holidays. The content of this table is generated automatically according to the language settings of NET Genium.
        - Holiday name – the name of the national holiday
        - Date – the date of the national holiday
        - Half-day – check box for the purpose of selecting half-day public holidays in Germany

- **Appearance** – database table "slayout" with a list of used graphic skins NET Genium
  - Name – the name of the skin
  - Save as current appearance – check box for the purpose of marking the appearance as current, and for the purpose of subsequent regeneration of CSS styles.
  - Font name
  - Text size
  - Font color
  - Header – Logo
  - Header – Background image
  - …
- **Invalid login** – database table "sinvalidlogins" with a list of invalid logins to NET Genium
  - Date – the date of the invalid login
  - IP address – IP address from which the login attempt was made
  - Login name – used login name
- **Statistics** – database table "sstatistics" with statistics on the use of view pages NET Genium
  - View page – visited view page
  - User – login name of the user who visited the page
  - Date – date of visit
- **Statistics 2**- database table "sstatistics2" with statistics on the use of NET Genium view pages
  - View page – visited view page
  - Session – number of visits
- **Statistics 3**- database table "sstatistics3" with statistics on the use of NET Genium view pages
  - View page – visited view page
  - User – login name of the user who visited the page
  - Session – number of visits
- **Synchronization** – the "ssynchro" database table with a list of errors encountered during synchronization
  - Error date
  - Edit form – the name of the edit form in which the error occurred
  - Record ID – The ID of the database record for which the error occurred
  - Error – description of the error

20 NET Genium settings

- NET Genium settings consist of one part stored in a database, configured using a web application in administrator mode, and configuration files, which are stored on disk in the “NETGenium\Config” directory.

20.1 NET Genium settings configured in the web application

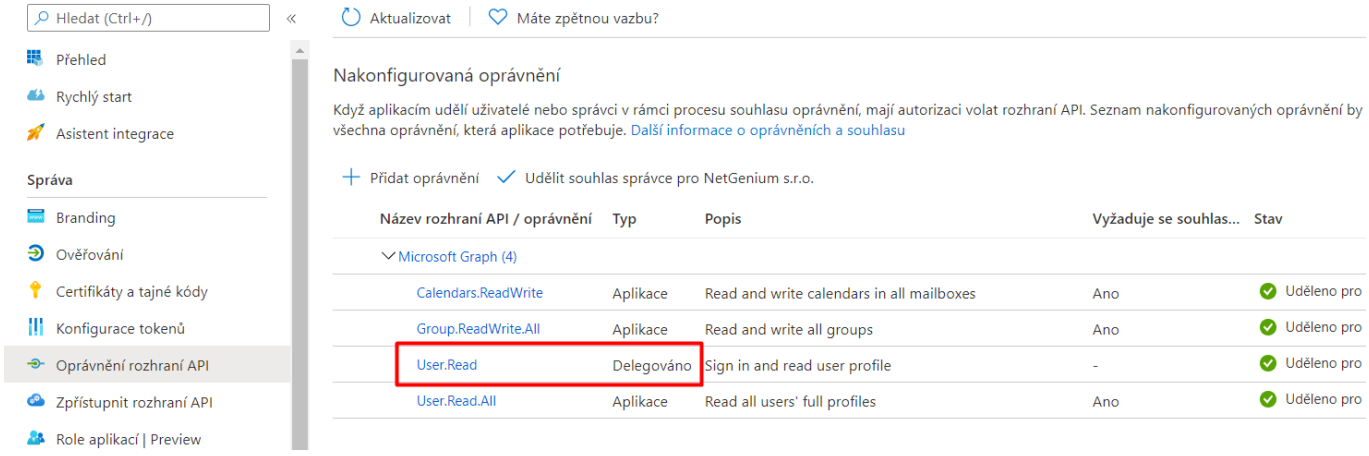
- The icon for invoking the NET Genium settings dialog is located in the toolbar in the navigation area in administrator mode. The toolbar is displayed by clicking on the gear icon or on the word NET Genium located to the right of this icon.
- A detailed description of NET Genium settings is given in a separate manual “NET Genium settings”.

20.2 Configuration files

- Configuration files contain data that is required for the web application to run properly, but is not part of the database. It is located in the “NETGenium\Config” directory in the form of text files.
- **ActiveLayout.txt** – the database ID of the skin that will be used as the default, regardless of the active skin settings in the database records of the configured NET Genium skins in the “Settings” application
- **BlindFriendly.txt** – the existence of this file ensures that lookup tables do not have a dynamically displayed toolbar, but a static one, which is still displayed under the lookup table title
- **ConnectionString.txt** – connection string to the NET Genium database
- **Copyright.txt** – placeholder HTML code for copyright displayed in the navigation area. If the file does not exist, the text string “© 20xx NetGenium” is displayed in the navigation area.
- **CustomIcons.txt** – A list of placeholder icons that replace the default icons used in the application. Each line of the file contains a single icon definition in the following format:
  Images/Picker.gif:Images/Custom/Picker.svg
- **DefaultLanguage.txt** – the two-letter abbreviation of the default language in which the basic application “Settings” is located and which was set for the users “Administrator” and “Anonymous” when they were created (“cs”, “de”, “en”, “fr”, “sk”)
- **Demo.txt** – The existence of this file will ensure that it will not be possible to save or delete anything in the web application. The file can optionally contain a list of enter-separated database tables in which changes will be allowed. For example, “susers_vf” will ensure that customization of lookup tables is possible.
- **DenyCookieless.txt** – The existence of this file will ensure that the web application will require the use of cookies on the client's side. Web application subsequently
  - will not use embedding SessionID in the URL of each web page (…/netgenium/(S(33cipeb0js15bo1t1xhkupr4))/default.aspx) and
  - will not allow logging in in one web browser under different user accounts on different tabs – the web browser merges individual sessions.
- **DenyExport.txt** – The existence of this file will ensure that the export of applications will be prohibited
- **DenyLogin.txt** – the existence of this file will ensure that it will be forbidden to log in to the application from any place other than the server desktop, resp. from the IP address “127.0.0.1”. The file can optionally contain a URL to which the user will be redirected when trying to log in to the application.
- **DenySynchro.txt** – The existence of this file will ensure that data synchronization is disabled

- **DenyTrimStream.txt** – the existence of this file will ensure that the automatic truncation of the HTML code of all pages of the application to a minimum will be prohibited – blank spaces between the end of one and the beginning of the other tag will be removed from the HTML code.
- **Developer.txt** – the existence of this file will ensure that errors generated in the application will be displayed in their native form, without a user-friendly form
- **EncryptionKey.txt** – encryption key for encrypting the "TextArea" and "RichTextBox" columns. If this file does not exist, the default encryption key is used to encrypt the data.
- **ExecutionTimeout.txt** – maximum length of request processing, resp. a server-side Web page in hours that is used to increase the default length of 5 hours, typically when you need to run long-running scripts. The maximum length of the request processing affects the value of the "executionTimeout" attribute in the "Web.config" file, which is set by the "FinishUpdate.exe" program after each update of NET Genium to a new version.
- **Framework.txt** – a file created automatically with the detected version of the .NET Framework, which is set in the properties of the application pool associated with the NET Genium web application
- **Framework4.txt** – the existence of this file will ensure that the update program "Update.exe" will download from the Internet new versions of NET Genium designed for the .NET Framework version 4
- **GoogleTranslationKey.txt** – API key to the Google Translation API service, used in richtext boxes for text translation
- **GoogleTranslationLanguages.txt** – a list of two-letter abbreviations of languages separated by a semicolon, which are offered in richtext boxes for text translation
- **Holidays.txt** – the condition according to which the loaded database records of public holidays will be filtered into date picker and into the functions "ADDWORKDAYS", "MEASUREWORKDAYS", "MEASUREWORKHOURS" or "jsMeasureWorkDays" – for example "ng_country = 'USERDETAILS(ng_countryforholidays)'"
- **License.txt** – NET Genium license key
- **LoginByIdentity.txt** – The existence of this file will ensure that the login window to NET Genium will include a check box for automatic user login in "Active Directory" using integrated Windows authentication without the need to manually enter the login name and password. Along with the creation of this file, it is necessary to configure the "LoginByIdentity.aspx" web page in IIS to support only integrated Windows authentication, see NET Genium installation guide.
- **LoginByMicrosoft.txt** – The existence of this file will ensure that the NET Genium login window will include a check box designed to automatically log the user in "Azure Active Directory" using integrated Microsoft authentication without having to manually enter the login name and password. Along with the creation of this file, it is necessary to configure NET Genium on the portal "https://portal.azure.com":
    - Application registration "**NET Genium**" at https://portal.azure.com/#blade/Microsoft_AAD_IAM/ActiveDirectoryMenuBlade/RegisteredApps
        - portal.azure.com / Azure Active Directory / App registrations
        - **NET Genium** as the application name
        - Finding **Application (client) ID** which is automatically created when the application is registered
            - Application (client) ID
        - Setting NET Genium URL to "https://{netgenium_url}/LoginByMicrosoft.aspx"
            - portal.azure.com / Azure Active Directory / App registrations / NET Genium / Authentication / Redirect URIs

- Authorization **access tokens**
  - portal.azure.com / Azure Active Directory / App registrations / NET Genium / Authentication / Implicit grant and hybrid flows / Access tokens (used for implicit flows)
- Authorization **Token ID**
  - portal.azure.com / Azure Active Directory / App registrations / NET Genium / Authentication / Implicit grant and hybrid flows / ID tokens (used for implicit and hybrid flows)
- Creation **client secret code**
  - portal.azure.com / Azure Active Directory / App registrations / NET Genium / Certificates & secrets / New client secret
- Setting API permissions for **Microsoft Graph**
  - portal.azure.com / Azure Active Directory / App registrations / NET Genium / API permissions / Add a permission / Microsoft Graph
  - API name "**Microsoft Graph – User.Read**", type "**Delegated**"



  - Creating a configuration file "**MicrosoftOAuth.json**"
    - Set the contents of the file to: "{"web":{"client_id":"**Application (client) ID**","auth_uri":"https://login.microsoftonline.com/organizations/oauth2/v2.0/authorize","token_uri":"https://login.microsoftonline.com/organizations/oauth2/v2.0/token","client_secret":"**client secret code**"}}"

- **MaxRequestCount.txt** – maximum number of requests of one user from one IP address per second. The increase of the default limit of 100 requests per second is current especially for anonymous web forms, where all users act in the context of one unlogged user.
- **MaxRequestLength.txt** – the maximum size of the user's request stated in MB, which is used to increase the default limit of 50 MB, typically when regular uploads of large files to the "File" and "Image" database controls in the edit form are required. The size of the request affects the values of the "maxRequestLength" and "maxAllowedContentLength"

attributes in the “Web.config” file, which the “FinishUpdate.exe” program sets after each update of NET Genium to a new version.

- **MetaTags.txt** – a list of HTML tags that are inserted in each edit form in the header of the web page between the tags “<head>”And“</head>”
- **NoFrames.txt** – the existence of this file will ensure that the frameless version of NET Genium will be used – the header and navigation area will be part of the HTML code of portlets, view pages or edit forms
- **NoScript.txt** – a text message that is automatically inserted inside the “<noscript>”Into each edit form, and specifies placeholder text for users who do not have javascript enabled in their web browser. For example:
  “Dear users, unfortunately your web browser does not support javascript, which is necessary for the correct display of these pages.
  Please turn it on.
  Thank you”
- **OCRServer.txt** – The URL of the remote NET Genium that provides OCR services in conjunction with the “#ocrserver#” variable and the “OCR2TEXT” or “OCR2FILE” functions.
- **ResponseHeaders.txt** – a list of headers separated by enter, which are automatically inserted into each response. Each line of the file contains a definition of one header in the following format:
  “Key:value”
- **SecureCookies.txt** – The existence of this file will ensure that each cookie has the “Secure” parameter set. With this configuration file, it is necessary to control the behavior of the “Secure” parameter only for web servers that have SSL termination set on the firewall (for example, F5). Web servers that have an SSL certificate set up directly in IIS do not need this configuration file, and the “Secure” parameter is set automatically for each cookie.
- **SessionTimeout.txt** – length in minutes, after which the session of the currently logged in user will be terminated due to his inactivity
- **SingleLogin.txt** – The existence of this file ensures that the user will always have only the last valid login to the application active. Previous sessions will be automatically logged out.
- **SimpleClick.txt** – The existence of this file ensures that a single click is used to open records from lookup tables. The default setting for opening records from lookup tables requires double-clicking.
- **SSL.txt** – The existence of this file will ensure that each request is automatically redirected to “https” via the insecure “http” channel. The file can optionally contain the URL of the page to which the user will be redirected if trying to view NET Genium via the “http” channel.
- **SysLog.txt** – IP addresses of “SysLog” servers separated by a semicolon, to which all log messages will be sent automatically, see. chapter “Logging”
- **Tester.txt** – The existence of this file will ensure that the update program “Update.exe” downloads from the Internet new test versions of NET Genium, which are intended for partners or developers of NetGenium in order to debug new features. Test versions may contain bugs or be unstable!
- **UnderConstruction.txt** – The existence of this file will ensure that it will not be possible to use the web application, and each request will be automatically redirected to the “UnderConstruction.aspx” web page with a message that the application is unavailable. The file can optionally contain a text message that appears instead of the default unavailability message.
- **UnderConstruction2.txt** – The existence of this file will ensure that no one other than the “Administrator” with the highest privileges can log in to the application, and any requests from all other users will be automatically redirected to the “UnderConstruction.aspx” website with a message that the application is unavailable. The file can optionally contain a text message that appears instead of the default unavailability message.
- **Version.txt** – current version number of NET Genium

- **XFrameOptions.txt** – the value of the “X-Frame-Options” header, which is inserted into each response on the main page, view pages, or in edit forms. The “X-Frame-Options” header allows you to tell the web browser whether the web page can be displayed in a frame. By default, the above pages cannot be displayed in a frame; by setting the value to “*”, other domains can be allowed to display in a frame.

20.3 File attachments

- File attachments are stored on disk in the “Files” directory. Each file attachment has file name information stored in the database in the “sfiles” table, and is stored on disk under a changed file name. The file name on the disk without the extension is defined as the record ID of the file attachment in the “sfiles” table, the extension is then retained. An example of the first file attachment stored in NET Genium may be “1.pdf”.
- File attachments are created by uploading a file using the “File” or “Image” controls in the edit form, creating a file attachment with a server function in a script, etc.
- The directory with file attachments can be redirected to another location on the disk using the configuration file “Files” without extension, which is stored directly in the directory of the same name, similar to other file attachments. The contents of the “Files” file then define the path on the disk to an alternative directory in which the file attachments are to be saved.

21 Reports

- The icon for invoking a dialog with a menu of available reports is located in the toolbar in the navigation area in administrator mode. The toolbar is displayed by clicking on the gear icon or on the word NET Genium located to the right of this icon.
- A detailed description of the reports is given in a separate manual “Reports”.

22 Logging

22.1 Incident logging

- Incident logging is provided by two basic operations:
  - Saving log messages to disk files in the “NETGenium\Logs” directory
  - Simultaneous sending of the same log messages to the “SysLog” server

22.1.1 Log files

- Log messages are stored in the “NETGenium\Logs” directory in the following files:
  - Auth-year-month.log – List of successful and unsuccessful logins and logouts
  - Download-year-month.log – List of downloaded files
  - Errors.log – List of errors encountered during user work
  - Export-year-month.log – List of user exported data
  - FinishUpdate.log – List of errors encountered during the application update to the new version
  - Fulltext-year-month.log – List of errors encountered during full-text search of file attachments
  - ResendEmlFiles-year-month.log – List of commands for communication with the SMTP server when trying to resend e-mail messages from the disk queue, resp. from the “NETGenium\Logs” directory
  - RunScript-year-month.log – List of executed scripts via “RunScript.exe”
  - Scripts60sec-year-month.log – List of scripts that lasted longer than 60 seconds
  - SMTP-year-month.log – List of successfully and unsuccessfully sent e-mails
  - System.log – List of informative messages about application startup (often related to starting or restarting the application pool)
  - UnsafeTags.log – List of originally entered and resulting HTML code that was cleaned up when entering formatted texts for “RichTextBox” controls
  - Version.log – List of database structure changes related to the new version of the application
  - Warnings.log – List of alerts
- The directory with log files can be redirected to another location on the disk using the configuration file “Logs” without an extension, which is stored directly in the directory of the same name, similar to other log files. The contents of the “Logs” file then define the path on the disk to an alternative directory in which the log files are to be saved.

22.1.2 SysLog

- Sending log messages can be set to multiple “SysLog” servers at the same time. The settings are made using the configuration file “SysLog.txt” located in the “NETGenium\Config” directory.
- If the configuration file “SysLog.txt” is created and contains one IP address of the “SysLog” server, or more IP addresses of “SysLog” servers separated by a semicolon, all log messages will be automatically sent to the specified “SysLog” servers, and at the same time will be saved in the “NETGenium\Logs” directory.
- In the absence of the “SysLog.txt” configuration file, logging is performed only to files in the “NETGenium\Logs” directory.

- Messages sent to the “SysLog” server have the following format, which uses a space as a delimiter for each value:“Date time application_server_domain category category event_type login_name ip_user_user message”.
    - Categories – Auth, Download, Errors, etc.
    - Event type – VERBOSE, NOTICE, WARNING, ERROR, FATAL

22.2 Web application speed logging

- NET Genium logs the following events to the database:
    - Speed of loading edit forms – into the database table “stables_stats”
    - Loading page loading speed – into the database table “sviewpages_stats”
    - Database query speed – to the “squerybuilder_stats” database table
    - Speed of external functions – to the database table “sngef_stats”
- Enabling or disabling logging is performed in a separate dialog under the link “Edit NET Genium” on the “Other” tab:
    - Off – logging is off
    - To database – logging takes place only to the database
    - On disk – logging takes place only on disk in the “Logs” directory
    - To database and disk – logging takes place to the database and to disk in the “Logs” directory
- Display of aggregated statistics from logs created in the database is possible using reports, see chapter “Reports”.

23 Installation

- A typical NET Genium installation consists of the following steps:
  - Firebird or MSSQL database server installation
    - A detailed description of the Firebird database server installation is given in the separate “Firebird Installation” manual.
    - A detailed description of the installation of the MSSQL database server is given in the separate manual “Installation of MSSQL”.
  - NET Genium installation – a detailed description of NET Genium installation is given in a separate manual “NET Genium installation”.
  - SMTP server installation – a detailed description of SMTP server installation is given in a separate manual “SMTP installation”.
- Deploying NET Genium for mission-critical applications should differentiate between development and operational environments, running ideally on two different servers.
- A detailed description of managing the development environment and releasing new releases is provided in a separate “Development Environment” guide.

24 Utility

- NET Genium includes a number of utilities located on disk in the “Backup”, “bin” and “Config” directories:
  - Backup
    - BackupServerService.exe
    - BackupServer.exe
    - FileBackup.exe
    - SqlBackup.exe
  - bin
    - ApplicationManager.exe
    - FileUpload.exe
    - FinishUpdate.exe
    - LogService.exe
    - OnlineUsers.exe
    - PrintPdf.exe
    - ResendEmlFiles.exe
    - Restart.exe
    - RunningQueries.exe
    - RunScript.exe
    - Setup.exe
  - Config\Tools
    - Deactivate.exe
    - Activate.exe
    - GrantLogin.sql
    - GrantLogin.bat.txt
    - MemoryDumps.txt
    - SSL.reg
    - SSL-ie6.reg
    - TracingRequests.txt
    - TuningQueries.sql
    - TuningQueries.txt
    - WALTU.exe
- A detailed description of all utilities is given in a separate “Utilities” manual.

25 Backup

- A detailed description of NET Genium backup is given in a separate “Backup” manual.