User Guide

Prepared by:
nuSoftware

Overview

Requirements

Basic knowledge concerning spreadsheets is enough knowledge to commence following this user guide. Having an understanding of spreadsheets will help to grasp the concept of information which is stored in rows and columns.

How to approach this user guide

*If you want to skip all of the background information and just start following the step by step guide, then go to the section call '**Building your first application'.***

We have attempted to structure this user guide by presenting each new concept in a logical sequence. Each concept builds upon the previous. However, with such a range of information being covered, some topics might seem at first to be disconnected. What you will need to do is to hold various ideas together as you progress through each stage of this user guide and then hopefully all of the various concepts will become clearer.

	Throughout this user guide, additional information will be highlighted by Mike - the nuBuilder robot.

```
Code examples will appear like this dialogue box.
```

When any component or key term of nuBuilder is mentioned, like **Screen** or **Button**; it will be highlighted as **bold**.

nuBuilder site

Before you commence following the steps outlined in this user guide you will need to have access to a nuBuilderForte site.

Introduction to nuBuilder

In this section we will learn about the following:

- What is nuBuilder
- Purpose of nuBuilder
- Globeadmin
- Dual login
- Two screen pattern
- Key Terms

What is nuBuilder

nuBuilder is simply a software package which provides two main functions:

1) It is a tool for **Developers** to *build* a custom database application.
2) It is the framework for **End Users** to log into and use their custom *built* application.

Purpose of nuBuilder

nuBuilder is only used for making database applications. In other words you cannot use nuBuilder to build a computer game or a web site or any other type of application other than a database application.

Globeadmin

All development in nuBuilder is done with the default user name which is **globeadmin**.

The **globeadmin** user is a special user built into the nuBuilder system. This user does not reside in the database where all other usernames and passwords are stored. When a nuBuilder site is installed and setup on a server, the **globeadmin** password is placed in a configuration file.

The only way to **develop** in nuBuilder is with the **globeadmin** login, also known as **Developer login**.

	The **globeadmin** default username can changed in the installation configuration file.

Dual login

Both **End Users** and **Developers** login nuBuilder. There is not a separate tool for **Developers** and another for **End Users**, both share that same interface.

Developer login

The **globeadmin** login provides access to everything within nuBuilder. ⟶ *Figure 1* shows the default globeadmin **Home Page**.

User login

When an **End User** logs into a nuBuilder site, they only have permissions to the **Screens** and **Reports** which have been granted to them. ⟶ *Figure 2* shows an example of an end user **Home Page.**

Figure 1.

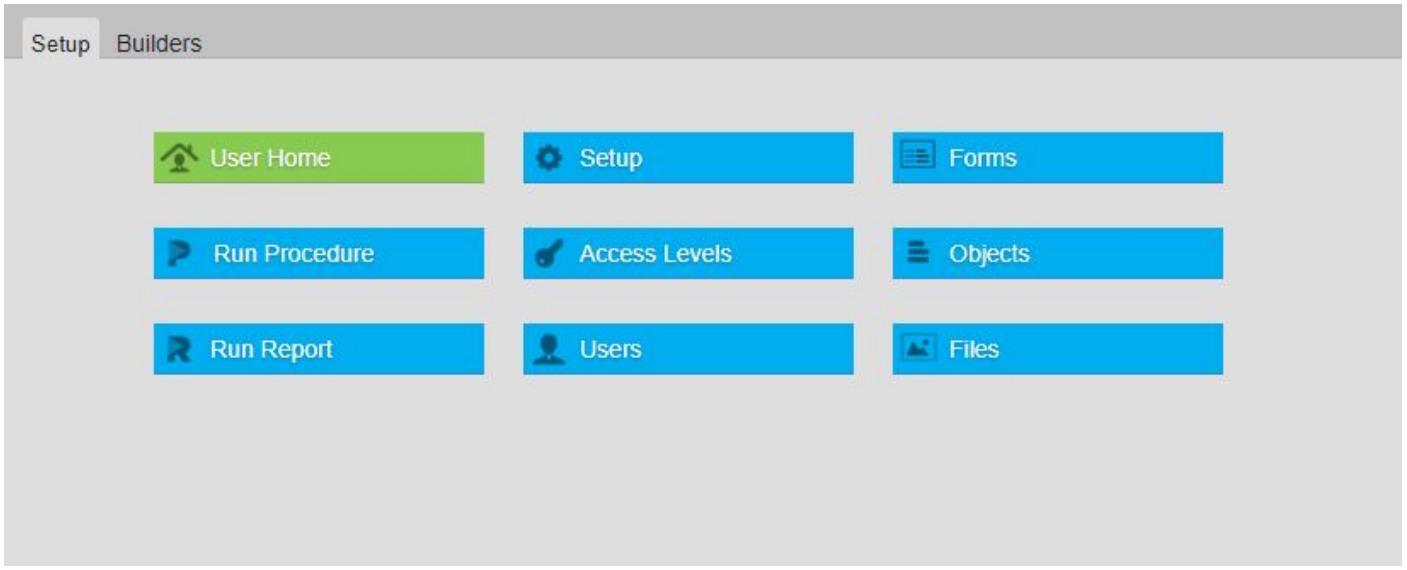
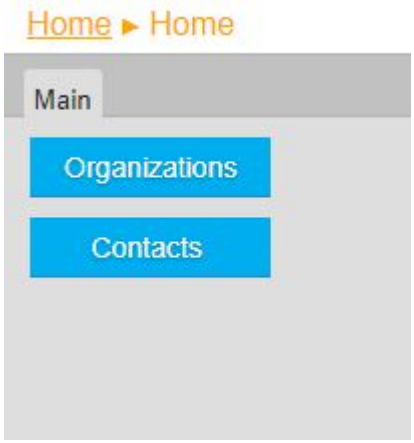


Figure 2.



	**End Users** permissions are defined in groups called **Access Level** ⟶ *(We will cover this in more detail later, see steps to setup an **Access Level**)*

Two screen pattern

nuBuilder is designed with a simple two screen pattern. These two screens are called the **Search Screen** and the **Edit Screen**.

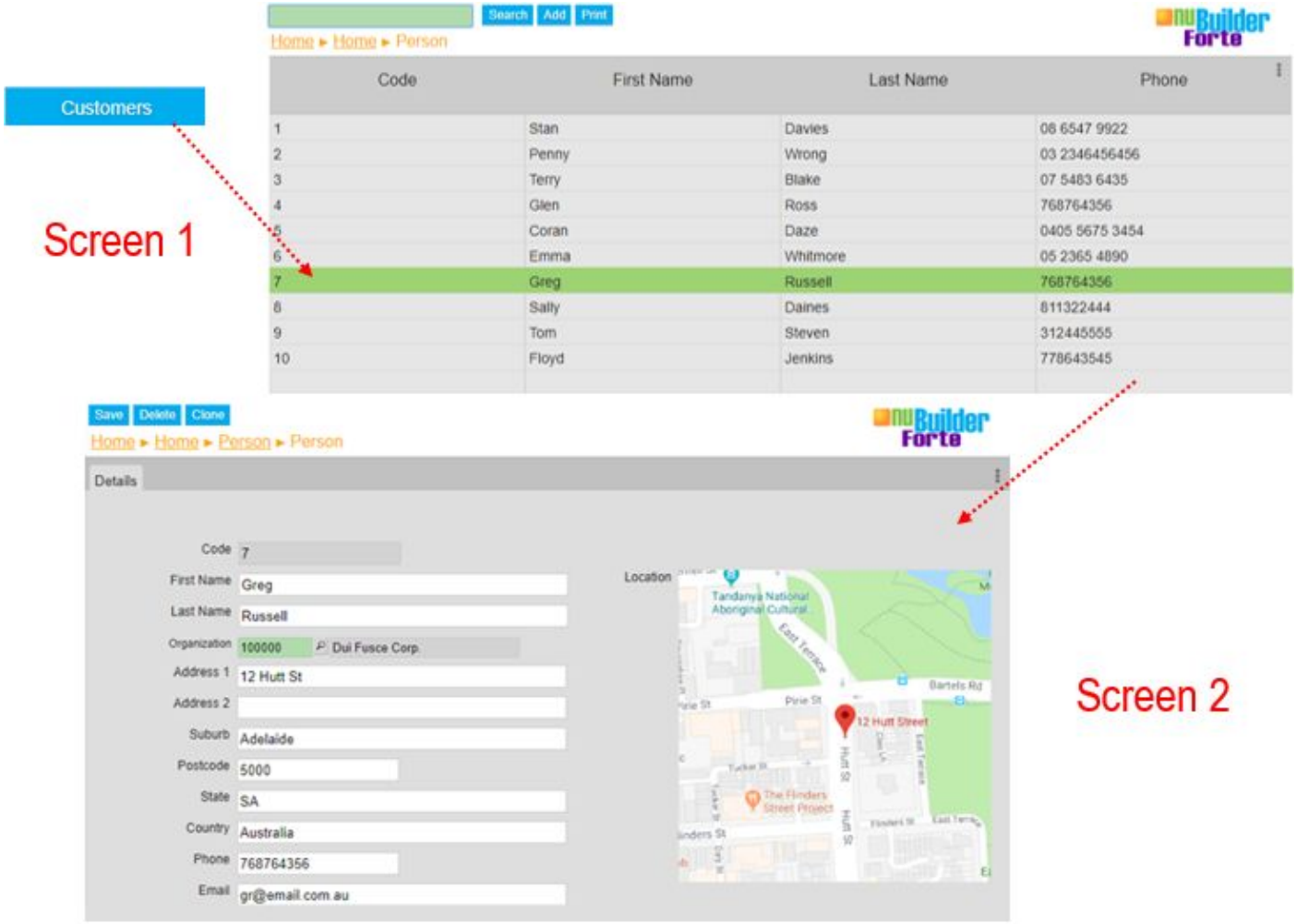
Everything in nuBuilder can be broken down into the following two basic actions:

1. Searching for information
2. Adding or editing information

Example 1. Let's say that you need to find the address for a customer in your organisation; you would normally follow these two steps: ⟶ *Figure 3.*

1. On your **Screen** you would have a **Button** called 'Customers'. Clicking on that **Button** will load **Screen 1 ⟶ Search Screen**
2. Once you have found the customer, you would then click on that row in the **Search Screen** which then loads up **Screen 2 ⟶ Edit Screen**

Figure 3.

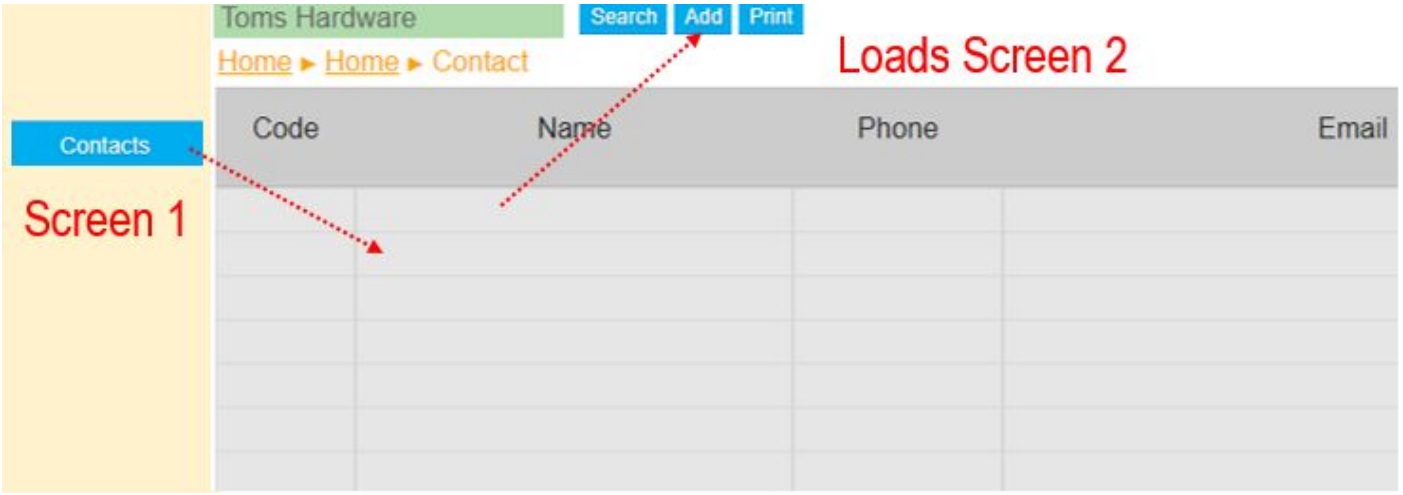


Example 2. Let's say that the customer which you are looking for does not exist in your database and therefore you want to add that customer to you database. You would normally follow these two steps: → *Figure 4.*

1. On your **Screen** you would have a **Button** called 'Contacts'. Clicking on that **Button** will load **Screen 1 → Search Screen**
2. Once you have determined that the customer which you are looking for does not exist in your database, you would then click on the **Add Button**. This will load **Screen 2 → Edit Screen**

The **Add Button**, takes you to the same **Screen** which is used for editing a record. The only difference is that it is a blank record.

Figure 4.



You could be searching for Invoices, Jobs, Quotes, Customers, Contacts, Bookings or anything else that has been developed in your nuBuilder site; everything in nuBuilder will follow the pattern of a **Button** on the **End User's HomePage** which then loads a **Search Screen**, which then loads an **Edit Screen**.

Advantages of the two screen model

1. Once you have trained an **End User** to search and edit information, you have effectively trained that **End User** to use your whole application

2. Designing an application becomes easier because all you need to do is to break down the design of your application into a list of **Screens** that you will need to build

3. You do not need to spend anytime in designing a menu system or a hierarchy of navigation buttons for your new application

Key Terms

The following is a list of key terms used in the nuBuilder system.

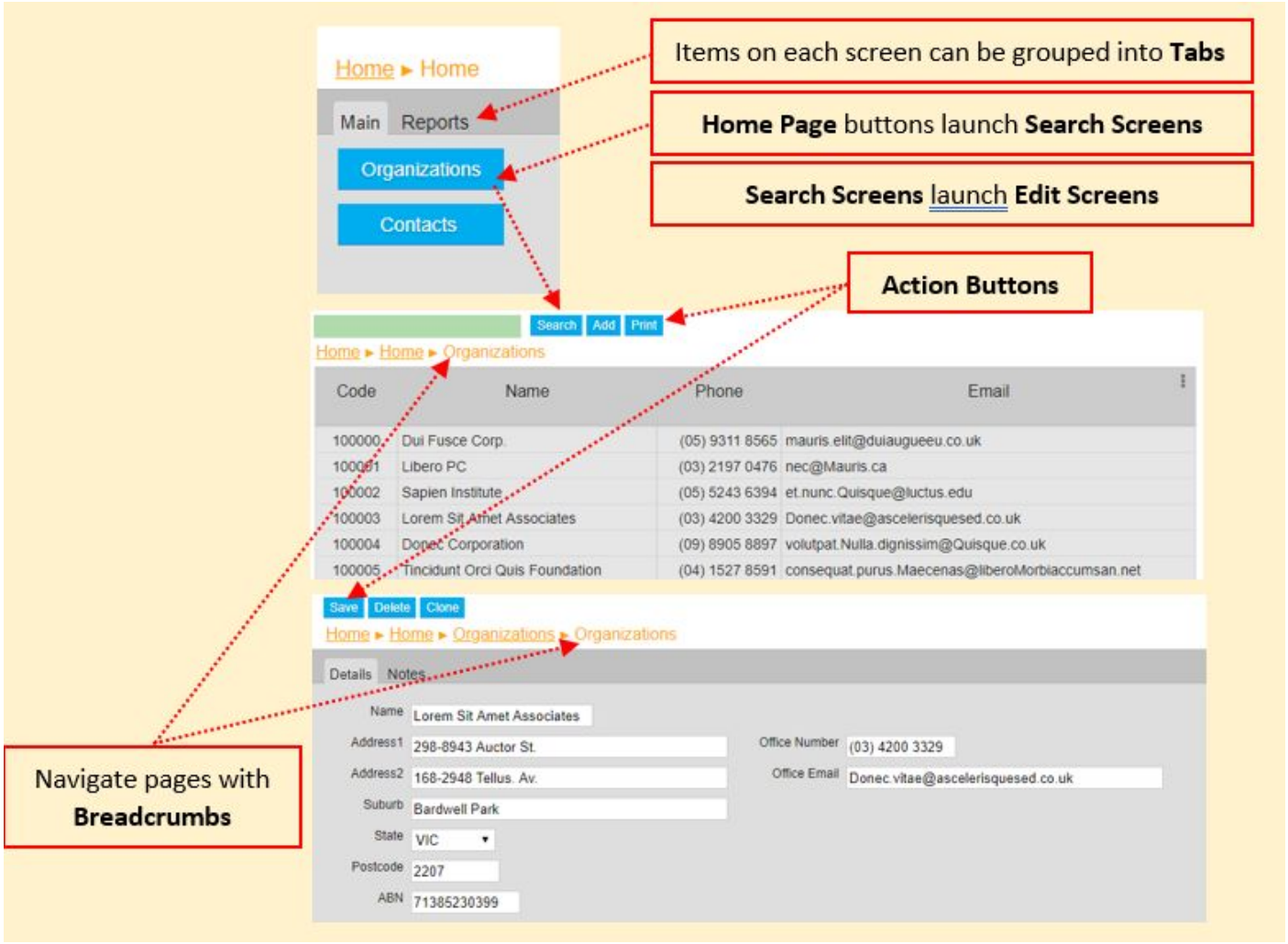
End Users	The people who will login and use your nuBuilder application
Access Levels	Groups permissions granted to **End Users**
Forms	The section in nuBuilder where **Search Screens** and **Edit Screens** are configured.
Objects	11 different **Screen** components that are linked to **Forms**
Tabs	Grouping of **Objects** on **Screens**
Home Page	The first page loaded when logging into nuBuilder
Search Screen	The search section of a **Form**
Edit Screen	The edit section of a **Form**
Criteria Screen	The **Screen** used to run **Reports** or **Procedures**
Breadcrumbs	Navigation links
Actions Buttons	**Save**, **Clone**, **Delete**, **Add** and **Print**
Table	A collection of data stored in a **Database**, comprised of **Rows** and **Columns**
Field	An individual element of a **Table**
Column	Alternative name for a **Field**
Browse Screen	Alternative name for a **Search Screen**
Record	Refers to a specific **Row** in a **Table**

	**Forms** are primarily concerned with **Tables**

	**Objects** are primarily concerned with **Fields**

Please familiarize yourself with some of the nuBuilder **Screen** components shown in ⟶ *Figure 5a & 5b.*

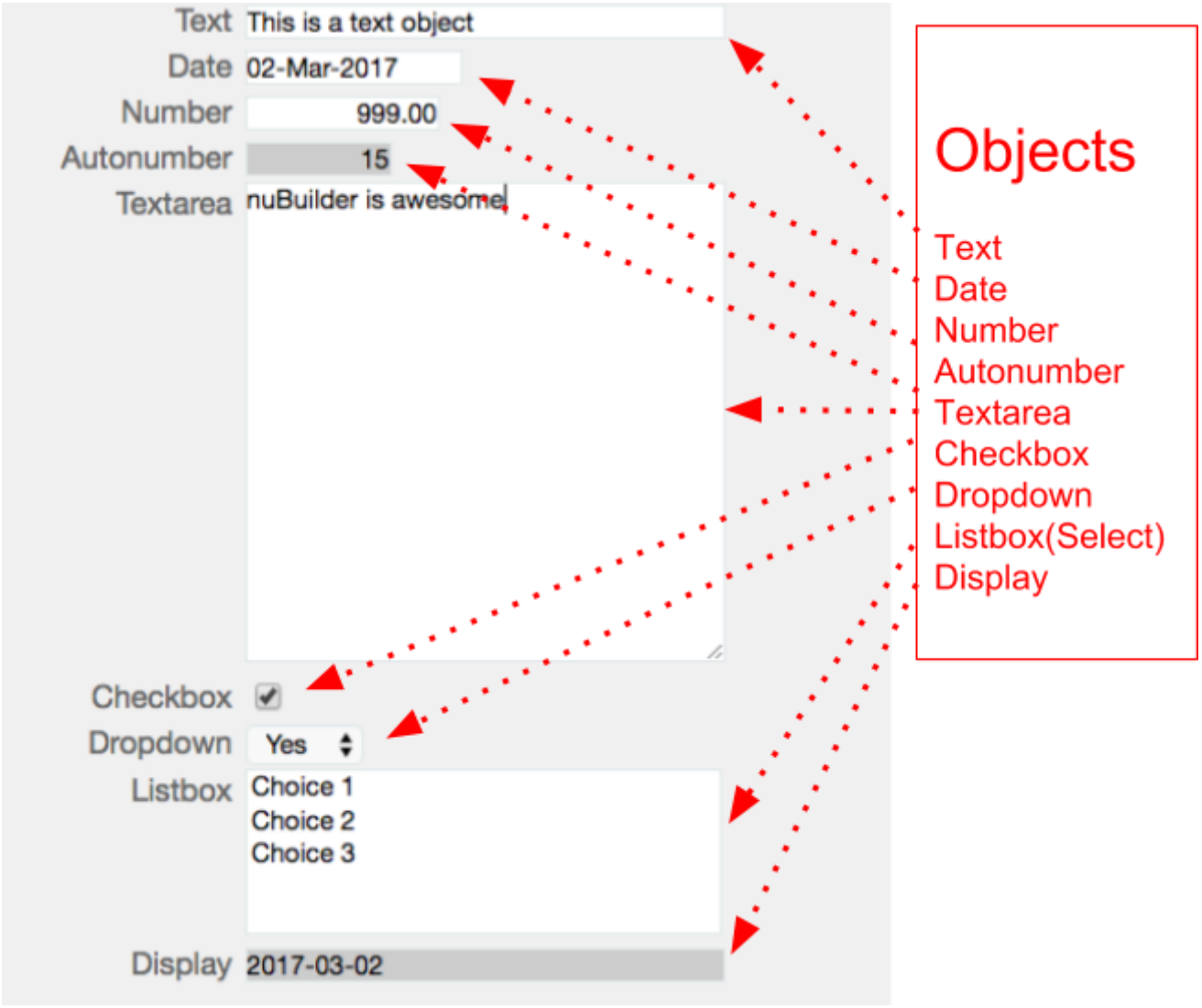
Figure 5a.



	The **Screen** that a user will first see when they initially login to a nuBuilder site is called the **Home Page.** The **Home Page** is simply an **Edit Screen** without any **Actions Buttons**.

	The **Screen** used to run **Reports** or **Procedures** is called a **Criteria Screen** which is simply an **Edit Screen** with different **Actions Buttons**.

Figure 5b.


Text
This is a text object
Date
02-Mar-2017
Number
999.00
Autonumber
15
Textarea
nuBuilder is awesome
Checkbox
Dropdown
Yes
Listbox
Choice 1
Choice 2
Choice 3
Display
2017-03-02
Objects
Text
Date
Number
Autonumber
Textarea
Checkbox
Dropdown
Listbox(Select)
Display

Relational databases

In this section we will learn about the following:

- What is a relational database
- Tables
- Unique values
- Primary keys
- Foreign keys
- One to many relationships

What is a relational database

A relational database is an information storage system which is organised by tables.
Each table within a relational database can have columns set up as either a primary key or a foreign key.
Primary keys and foreign keys enable tables to be linked together in a relationship.

Tables

In order to learn about tables, it is helpful to start with looking at information that is stored in a spreadsheet.

Imagine you have a simple spreadsheet in your business which you use to store information regarding all of the organizations that you work with. Let's start with some basic information like the business name, the phone number, suburb and postcode. ⟶ *Figure 6*.

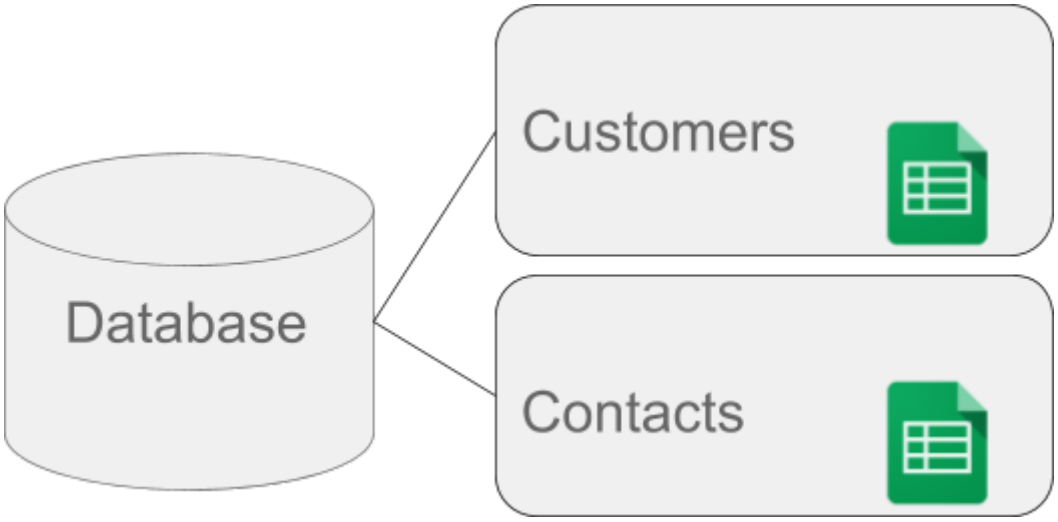
Figure 6.

	A	B	C	D	E
1	**Business Name**	**Phone**	**Suburb**	**Postcode**	
2	Bobs Plumbing	881321231	Woodcroft	5162	
3	A1 Printing	881456719	Adelaide	5000	
4	Mobile Repairs	220121043	Belair Athol	2560	
5					
6					

A database table is similar to a spreadsheet because they both contain columns and rows.

Picture a relational database as a container which holds all of your spreadsheets. Instead of using the word 'spreadsheet', use the term 'table'. ⟶ *Figure 7*.

Figure 7.



Unique values

The next thing we need to consider is what do we do if there are two businesses with the same name like 'Bob's Plumbing'? What would be an easy way to tell them apart from each other?

What we need to do is to add another column called 'customer number' and give each row in the spreadsheet a unique number. As you can see in *Figure 8*, we now can tell them apart.

Figure 8.

	A	B	C	D	E
1	Customer Number	Business Name	Phone	Suburb	Postcode
2	269470	Bobs Plumbing	881321231	Woodcroft	5162
3	269471	A1 Printing	881456719	Adelaide	5000
4	269472	Mobile Repairs	220121043	Belair Athol	2560
5	269473	Bobs Plumbing	21223432231	Sydney	2000
6					

Unique Values

Primary keys

Tables can have a column designated as the ‘primary key’. The database server will use the primary key to create an index which enables the database engine to quickly find information.

1. It is recommend that each table have a column set as the primary key
2. Primary keys must be unique
3. Primary keys cannot contain a blank value

Once a table has been configured in a nuBuilder application, the nuBuilder system will take care inserting a unique value in each primary key whenever a new record is created.

The type of primary keys that nuBuilder creates are 25 characters long and contain both letters and numbers.

A typical primary key might look like the following: `14e49d3d0ef3a7` This is not the type of value that a human would type or use in a search because it is too complex to easily recognise.

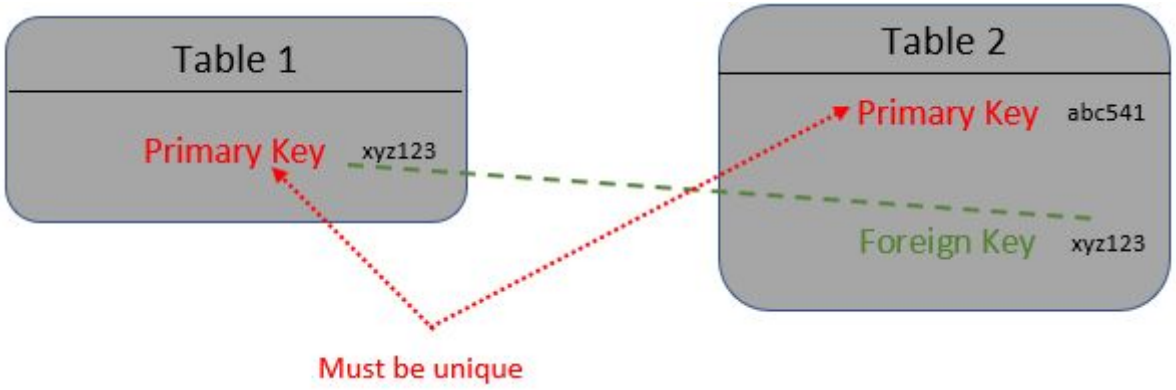
Primary keys are not normally displayed on a **End User's Screen**, they are columns which only a **Developer** would use when writing code.

	A primary key is similar to our example of *Unique Customer Numbers*. We could configure the table to use the 'Customer Number' field as the Primary Key. Although this will work, there are times when some business want to change their numbering system, therefore it is best to create a special field in a table that will never get edited or changed.

Foreign keys

A foreign key is a field in a table which contains a copy of another tables' primary key, this provides a way for tables to be linked together in a relationship.

Figure 9.



Just like primary keys, foreign keys are not normally displayed on an **End user's Screen** and are only used by **Developers** when writing code.

The type of foreign keys used in nuBuilder are the same as primary keys and also typically look like the following combination of numbers and letters: `14e49d3d0ef3a7`

Looking at *Figure 9*; there could be many rows in **Table 2** that relate back to **Table 1**. For this reason the value of a foreign key does not need to be unique within **Table 2** but must be unique within **Table 1**.

One to many relationships

Let's consider another spreadsheet to keep a list of people who work in each of the business in our earlier spreadsheet. Let's call this spreadsheet 'Contacts'. We would probably want to keep their First Name, and Last Name, a Mobile number and an Email address. And just in case we have two or more contacts called John Smith, will should also give them a unique number. → *Figure 10.*

Figure 10.

	A	B	C	D	E	F
1	Contact Number	First Name	Last Name	Phone	Mobile	Email
2	269470	John	Smith	881321231	401911123	john.smith@gmail.com
3	269471	Shane	Gelven	881456719	407779816	shane@nubuilder.com
4	269472	Steven	Copley	220121043	407779600	steven@nubuilder.com
5	269473	John	Smith	21223432231	401199119	john.smith@hotmail.com
6						
7						

So now we have two spreadsheets, one to store information concerning each business and another to store information concerning individual contacts within each of these business.

It is typical to expect that a business would have several people working in their organisation. So each business in our first spreadsheet would end up having one or more contacts in our second spreadsheet.

The term used to describe the concept of many contacts relating to a single business is a '**One to Many Relationship**'.

In ⟶ *Figure 11a*, we can see a single row in **Table 1** which shows the information of a single customer record. It has a primary key value of '`58a11df94545f2b`'.

We can also see many rows in **Table 2** which all relate to the single record in the customer table. This is achieved by having the same primary key value from **Table 1** of '`58a11df94545f2b`' stored in the foreign key column on **Table 2**.

Figure 11a.

Table 1

organization_id	org_code	org_name	org_address1	org_address2	org_suburb	org_state
58a11df94545f2b	100099	A Odio Industries	789-5519 Fusce Ave	Ap #817-6378 Mauris Ave	Goomalling	QLD

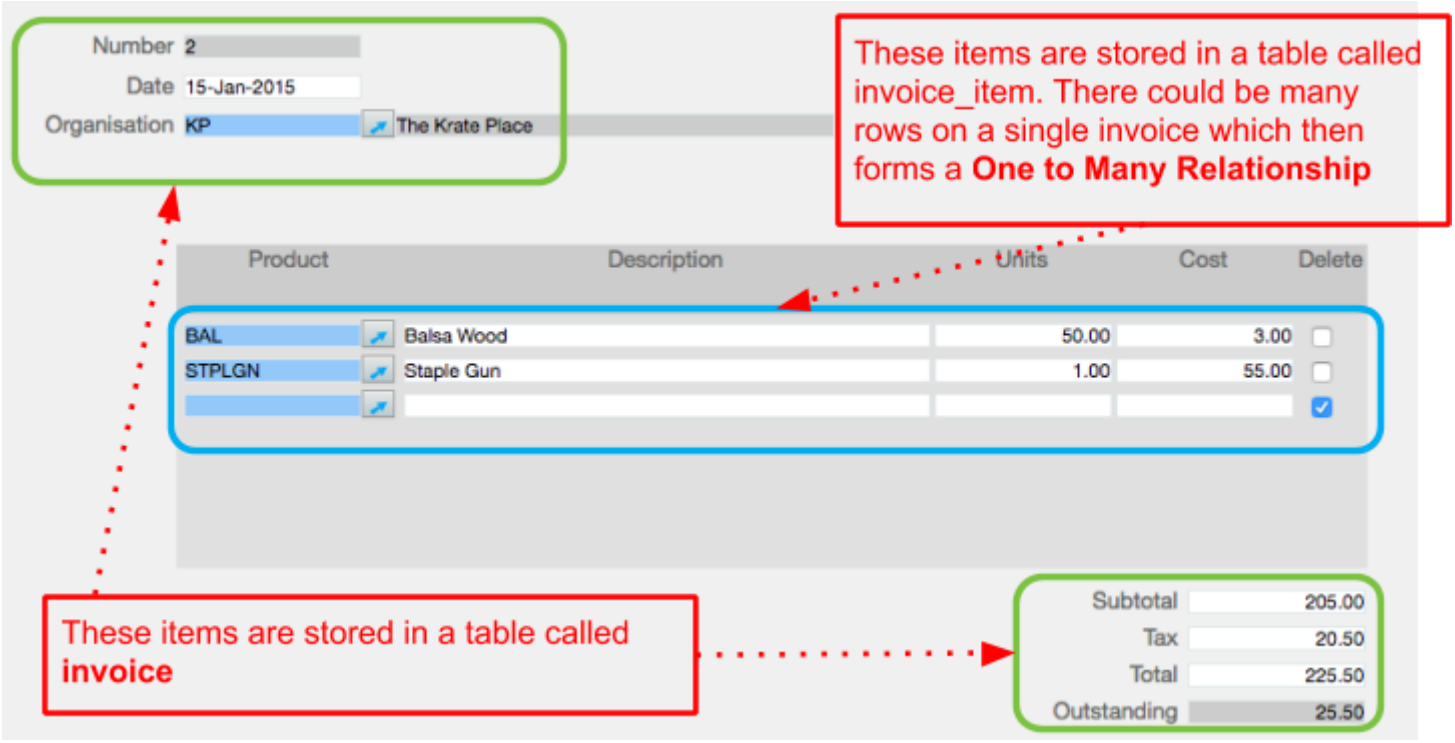
Table 2

contact_id	con_organization_id	con_firstname	con_lastname	con_email
58a11ee36f2e08f	58a11df94545f2b	Harlan	Thornton	Vivamus.nisi@ornare.com
58a11ee4e7da5a3	58a11df94545f2b	Chandler	Parsons	libero.lacus@ac.co.uk
58a11ee4e7f65c8	58a11df94545f2b	Carl	Booker	vehicula@euismodestarcu.ca
58a11ee4e80f84e	58a11df94545f2b	Rosalyn	Patton	id.erat.Etiam@nislsemconsequat.net
58a11ee4e8288eb	58a11df94545f2b	Nelle	Berger	diam.dictum.sapien@ornareInfaucibus.co.uk
58a11ee4e841398	58a11df94545f2b	Alden	Rose	sociis.natoque.penatibus@nunc.net
58a11ee4e85abee	58a11df94545f2b	Jesse	Campbell	varius@luctussitamet.com
58a11ee4e873173	58a11df94545f2b	Mikayla	Wallace	ligula@tortor.edu

In ⟶ *Figure 11b*, we can see another typical example of a **One to Many Relationship.** This screen shows an Invoice. The fields on the screen which have a **green border** are all stored in a table called '`invoice`'. The fields on the screen which have a **blue border** are all stored in a table called '`invoice_item`'.

We often refer to the row in the database which stores the invoice as the '**Parent Record**' and the rows that store each item on the invoice as the '**Child Record(s)**'.

Figure 11b.



Introduction to SQL

In this section we will learn about the following:

- What is SQL
- Running a query
- phpMyAdmin
- Steps to run an SQL query
- The SQL SELECT Statement

What is SQL

SQL stands for Structured Query Language.

SQL is a programming language which allows programmers to perform various actions within a relational database.

The following list shows some of the actions possible:

Types of Action	SQL Common Commands
Create new rows in a table	`INSERT, CREATE`
Read information that is stored in the database	`SELECT, JOIN`
Update information that is stored in the database	`UPDATE`
Delete information that is stored in the database	`DELETE, DROP, TRUNCATE`

As you can see, these actions can be abbreviated as C.R.U.D which is an easy way to remember the core functions of any database application. **C**reate / **R**ead / **U**pdate / **D**elete

The SQL language has over 800 reserved words which are used as commands in the SQL language. Most of the time you will only need to concern yourself with the short list of common SQL commands shown above. Furthermore, the nuBuilder system takes care of a large amount of SQL for you, therefore at this stage of this user guide you only need to learn about the following three SQL words:

```
SELECT
FROM
WHERE
```

	If you would like to read ahead of this user guide and teach yourself more about SQL you can follow this link: https://www.w3schools.com/sql/

Running a query

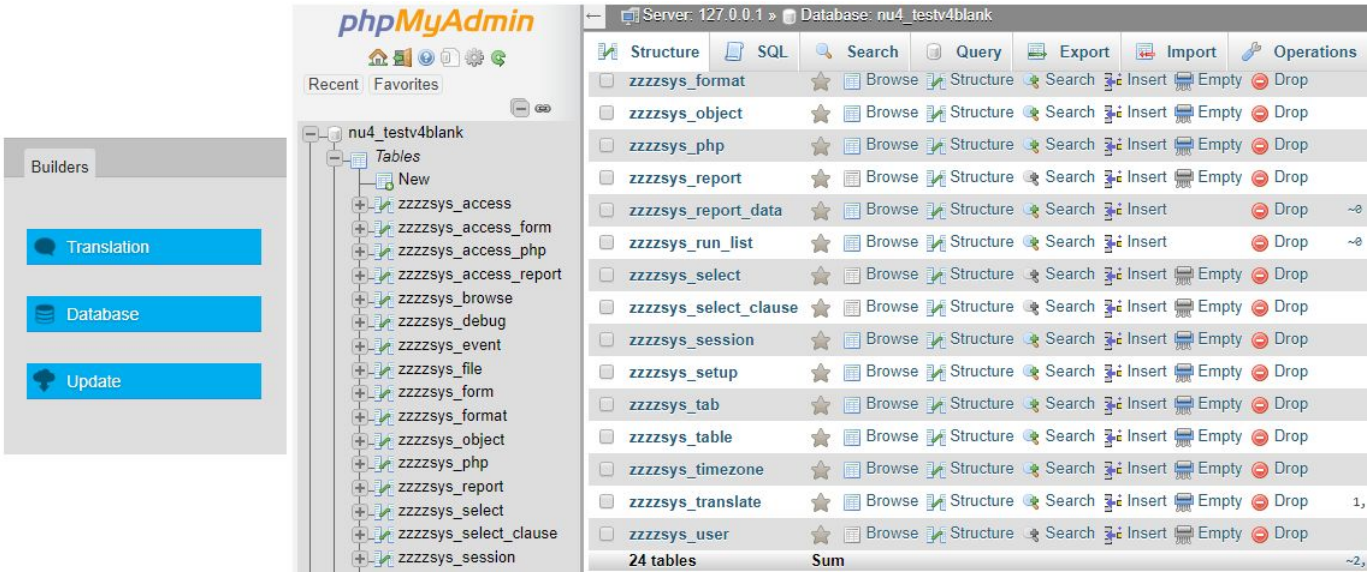
To use SQL you need to construct an SQL statement which is then interpreted and executed by the SQL server. This is known as '*running a query*'. To do this we will use an application called 'phpMyAdmin'.

phpMyAdmin

phpMyAdmin is a separate application to nuBuilder, however it is bundled with each copy of nuBuilder.

You can directly access phpMyAdmin within nuBuilder whilst logged in as 'globeadmin'. → *Figure 12.*

Figure 12.



As a nuBuilder developer you will be frequently using phpMyAdmin for the following tasks:

1. Creating tables
2. Testing queries

	phpMyAdmin is an open-source application.

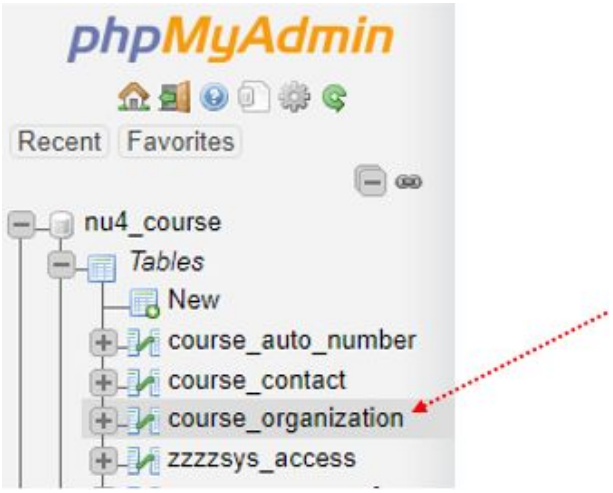
Steps to run an SQL query

1. Log into nuBuilder with your **globeadmin** account

2. Click on the 'Database' **Button** in the Builders tab to load up phpMyAdmin

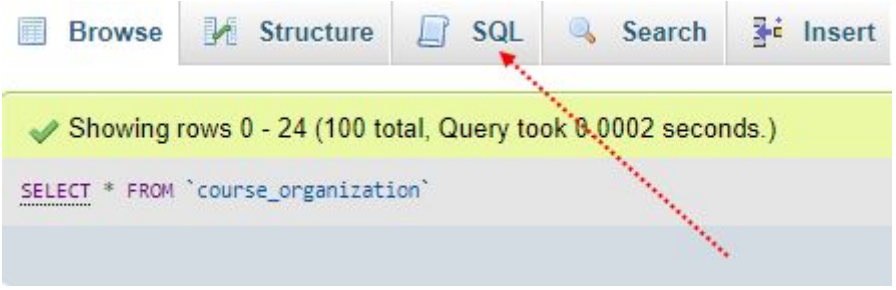
3. Your nuBuilder site has already been setup with a few tables with some dummy data so that you can quickly get started with learning SQL. Select the course_organization table by clicking on the link in the left panel which is labeled 'course_organization' ⟶ *Figure 13.*

Figure 13.



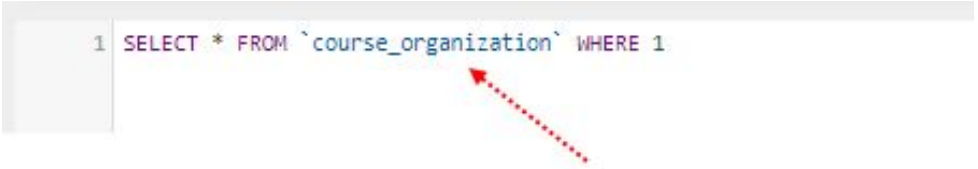
Click on the tab labeled SQL ⟶ *Figure 14.*

Figure 14.



You will see a textarea where you can type in SQL commands. There will already be an initial SQL command on the screen which you can use ⟶ *Figure 15.*

Figure 15.



You can now click on the **Go** button to run the above SQL statement.

Congratulations, you have just run the following SQL command ⟶ *Figure 16.*

Figure 16.

```
SELECT * FROM course_organization WHERE 1
```

We will now break down and explain the above SQL statement.

The SQL SELECT Statement

The following two examples show the basic format of a SELECT statement. ⟶ *Figure 17 and 18.*

Figure 17.

```
SELECT column_name, column_name FROM table_name
```

Figure 18.

```
SELECT * FROM table_name
```

SELECT

The `SELECT` command is used to retrieve information out of one or more tables within a database.

The asterisk symbol `*` tells the SQL server to retrieve all columns from all of tables in the SQL statement. Alternatively you could remove the asterisk symbol `*` and replace it with a comma separated list of actual column names as shown in ⟶ *Figure 19*

Figure 19.

```
SELECT org_code, org_name FROM course_organization
```

FROM

The `FROM` command informs the SQL server what table(s) to retrieve information from. In the example in *Figure 16,* there is only a single table included ⟶ `course_organization` Alternatively you can provide a comma separated list of table names as shown in ⟶ *Figure 20*

Figure 20.

```
SELECT * FROM course_organization, course_contact
```

WHERE

The `WHERE` command is used to filter the results of an SQL query.

In the example shown in *Figure 16,* the `WHERE 1` statement is a default filter that the phpMyAdmin application uses. The use of the number `1` might seem a bit confusing as it really has no effect on the end result. Most of the time we would execute this same type of SQL command without using a `WHERE` command ⟶ *Figure 21.*

Figure 21.

```
SELECT * FROM course_organization
```

When you know the specific value of a primary key in a table, you can use that value to retrieve a single row. We can be certain that we will always get a single row from our SQL statement because we know that primary keys are always unique. ⟶ *Figure 22.*

Figure 22.

```
SELECT * FROM course_organization
WHERE organization_id = '58a11df94545f2b'
```

Our final example of a SELECT statement demonstrates how we can retrieve rows in table that match a specific value. ⟶ *Figure 23.*

Figure 23.

```
SELECT * FROM course_organization
WHERE org_state = 'SA'
```

	The number `1` in programming languages is often used to mean *true* and the number `0` is often used to mean *false*.

	You may have noticed several tables in phpMyAdmin which have a prefix of `'zzzzsys_'` These are tables used by nuBuilder, it is best to leave these tables alone.

Designing your first application

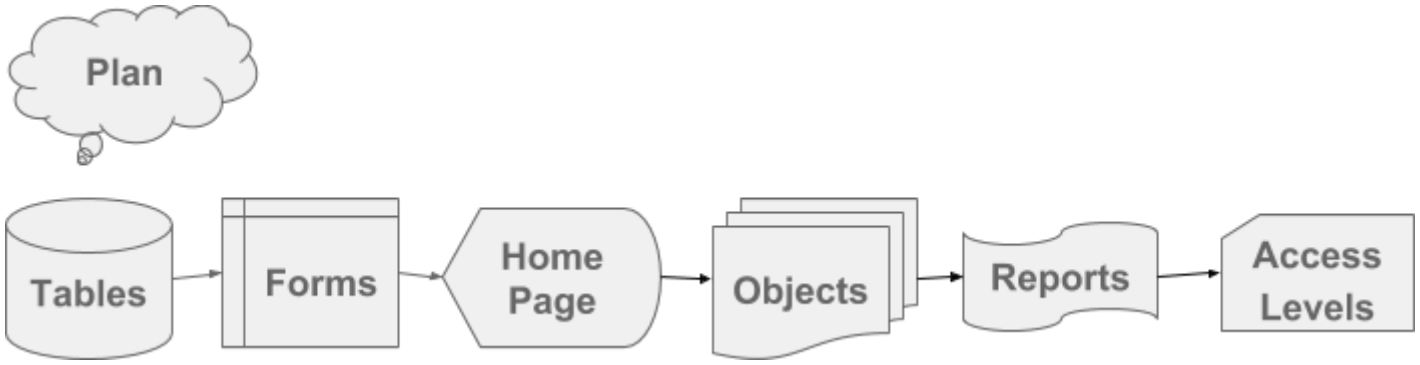
In this section we will learn about the following:

- The nuBuilder development process
- Planning your application
- Database Schemas
- Naming Conventions
- Creating Tables

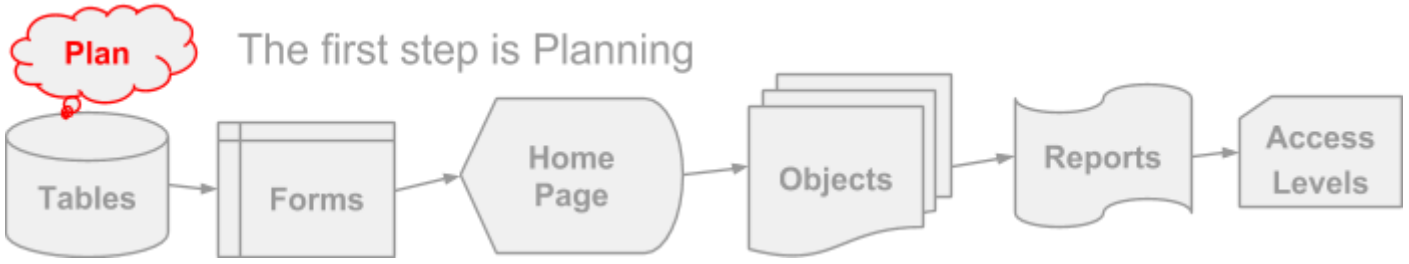
The nuBuilder development process

Developing applications within nuBuilder involves the following steps. ⟶ *Figure 24.*

Figure 24.



Planning your application



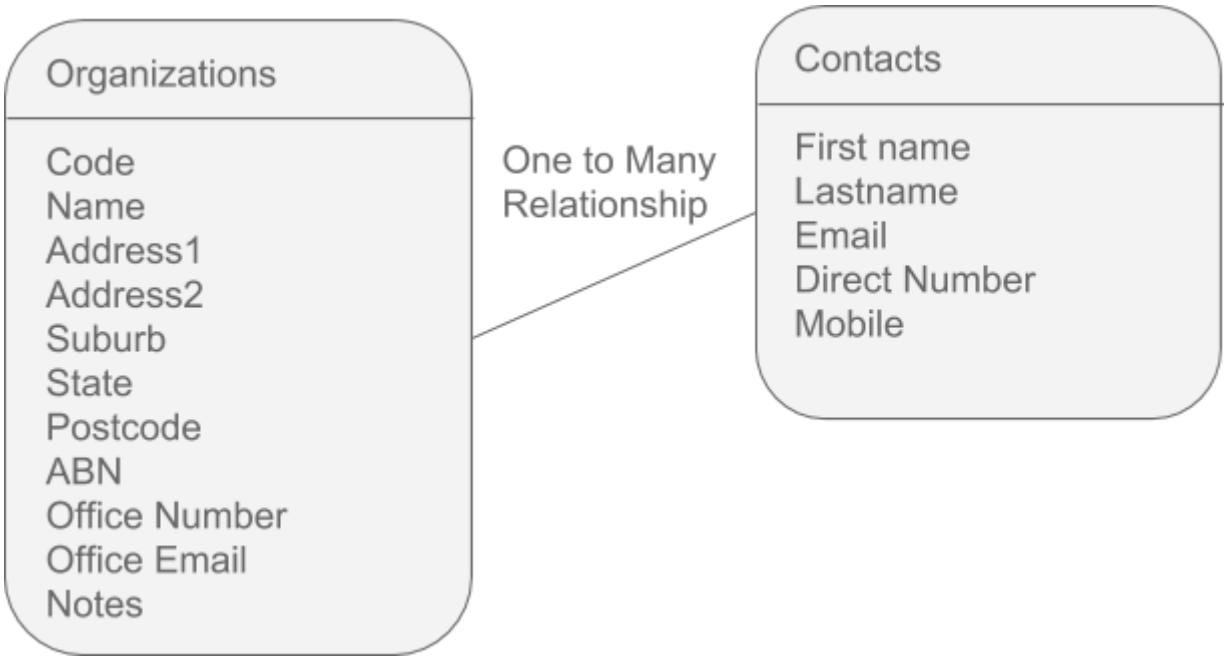
Our plan is to build a basic CRM system. CRM stands for Customer Relationship Manager. This CRM is based on the two spreadsheets we have been using. We will build a **Screen** to store information concerning each 'Customer' and another **Screen** to store information concerning each 'Contact'.

Database Schemas

A good indication that we have designed our application with enough planning is when we have completed a database diagram.

The name for a database design is called a 'Database Schema'. ⟶ *Figure 25a.*

Figure 25a.



Naming Conventions

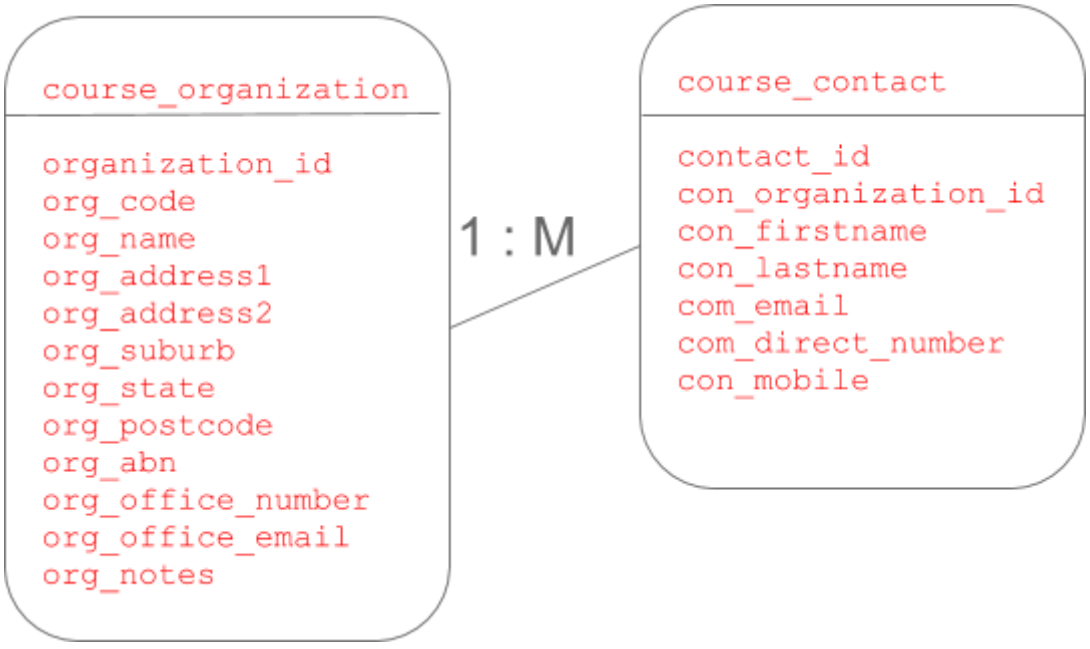
To complete this database Database Schema we could add the primary key and foreign key names and then choose complete field names for each field.

To do this we will introduce the naming conventions that we typically use when designing tables.

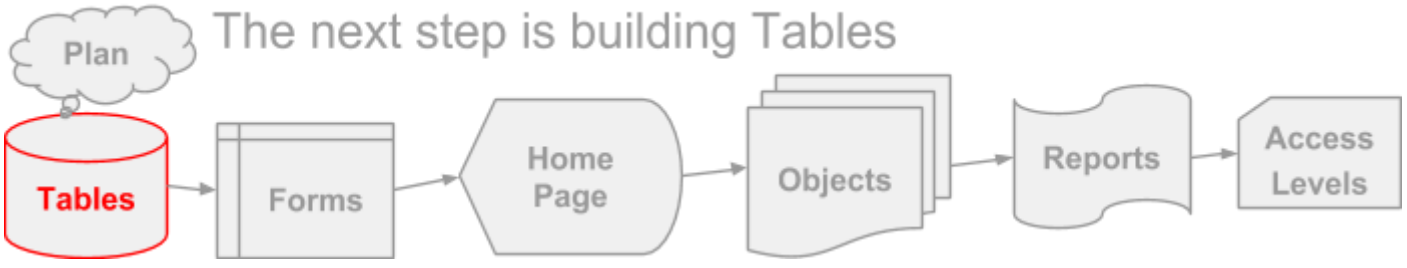
1. Lowercase characters for all table and field names
2. All table names are singular
3. The combination of the table name and '_id' as the suffix is used for the primary key column
4. Delimit each word with the underscore character '_'
5. Add a three letter prefix to each field which is derived from the table name

See ⟶ *Figure 25b* for a complete schema.

Figure 25b.



Creating Tables



We have already created the tables you need and populated them with test data so that you can get started straight away. You can log into phpMyAdmin to take a closer look at these tables.

For your reference we have provided the SQL statements used to create the 'course_organization' and 'course_contact' tables. → *Figure 26a and 26b.*

Figure 26a.

```
CREATE TABLE course_organization (
organization_id varchar(25) NOT NULL,
org_code varchar(10) NOT NULL,
org_name varchar(500) NOT NULL,
org_address1 varchar(500) NOT NULL,
org_address2 varchar(500) NOT NULL,
org_suburb varchar(500) NOT NULL,
org_state varchar(300) NOT NULL,
org_postcode varchar(300) NOT NULL,
org_abn varchar(30) NOT NULL,
org_office_number varchar(30) NOT NULL,
org_office_email varchar(150) NOT NULL,
org_notes text NOT NULL,
PRIMARY KEY (organization_id)
)
```

Figure 26b.

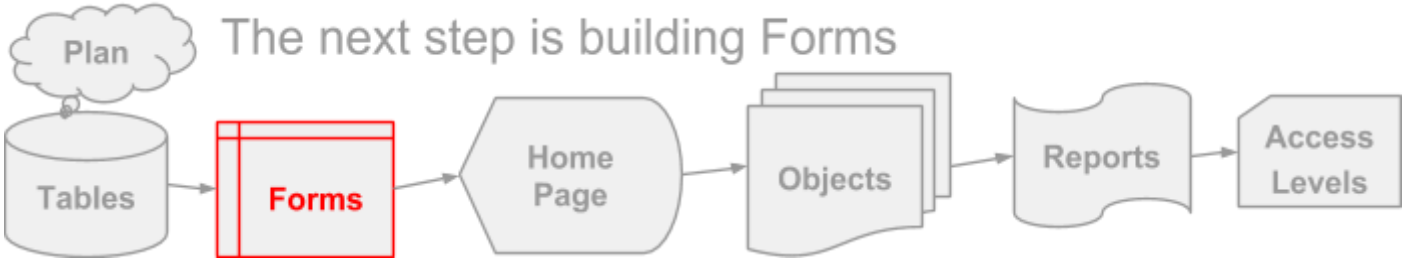
```
CREATE TABLE course_contact (
contact_id varchar(25) NOT NULL,
con_organization_id varchar(25) NOT NULL,
con_firstname varchar(150) NOT NULL,
con_lastname varchar(150) NOT NULL,
con_email varchar(150) NOT NULL,
con_direct_line varchar(150) NOT NULL,
con_mobile varchar(150) NOT NULL,
PRIMARY KEY (contact_id)
)
```

Building your first application

In this section we will learn about the following:

- Steps to build a **Search** & **Edit Screen**
- Steps to put a **Button** onto the **Home Page**
- Steps to add **Objects** to an **Edit Screen**
- Creating **Tabs**
- Controlling **Object** Placement
- Building Text Input **Objects**
- Choosing **IDs**
- Building **Dropdown Objects**
- Alternative Method to Create an **Object**
- Building **Textarea Objects**
- Building **Autonumber Objects**
- Edit **Object** Shortcuts
- Edit **Form** Shortcuts
- **Fast Form**
- **Object** Search Shortcut
- **Lookup Objects**
- Subform

Steps to build a Search & Edit Screen



All **Screens** are developed by creating a **Form**.

1. Log into nuBuilder with your **globeadmin** account
2. Click on the **Forms** button in the **Setup** tab ⟶ *Figure 27.*

Figure 27.

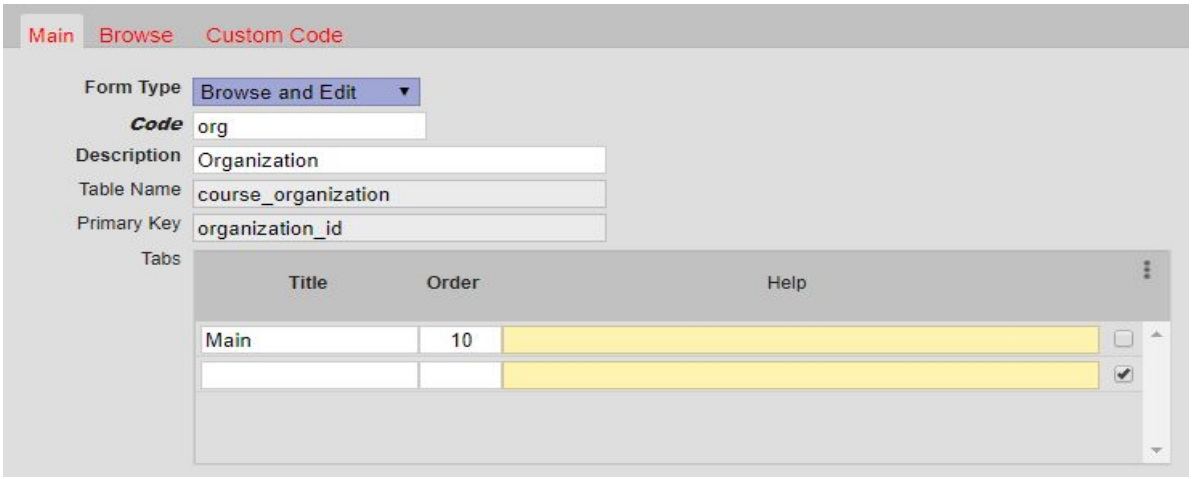
3. Click **Add** button.

As you can see on *Figure 28,* there are three **Tabs** on on the **Form** screen; 'Main', 'Browse' and 'Custom Code'. At this stage we only need to look at the first two **Tabs**; 'Main' and 'Browse'.

Figure 28.

4. On the **Main** tab, fill in the following items ⟶ *Figure 29.*

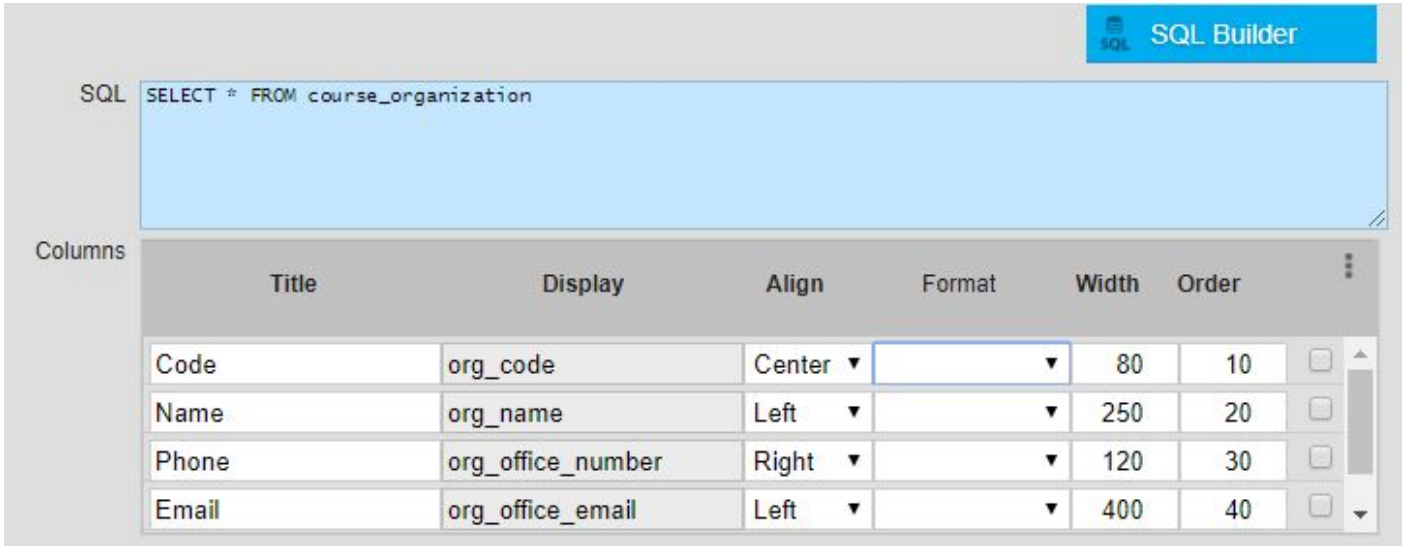
Figure 29.



Form Type	Browse and Edit	The form type is to give access to what the end user can do to the form
Code	org	Each **Form** needs a unique code so it can be referenced in other sections of nuBuilder
Description	Organization	The description is used on **Breadcrumbs** and **Lookups**
Table	course_organization	This is where we tell nuBuilder what table to use for this **Screen**
Primary Key	organization_id	This is where we tell nuBuilder the name of the primary key for the table we are using

5. On the **Browse Tab**, fill in the following items on each row ⟶ *Figure 30.*

Figure 30.



SQL	SELECT * FROM course_organization	The following SQL would have automatically appeared after you tabbed out of the above table field

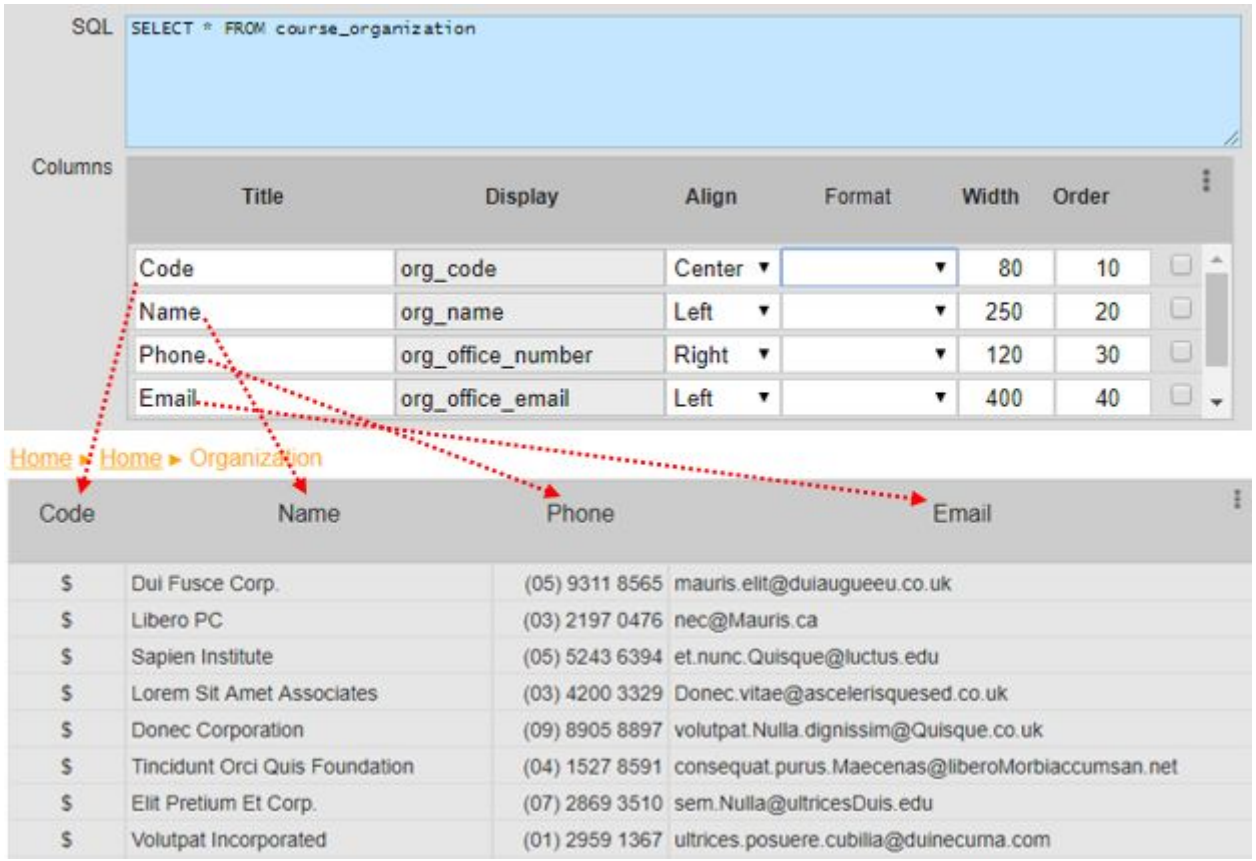
Title	Display	Align	Format	Width	Order
Code	org_code	Center		80	10
Name	org_name	Left		250	20
Phone	org_office_number	Right		120	30
Email	org_office_email	Left		400	40

6. Click **Save**

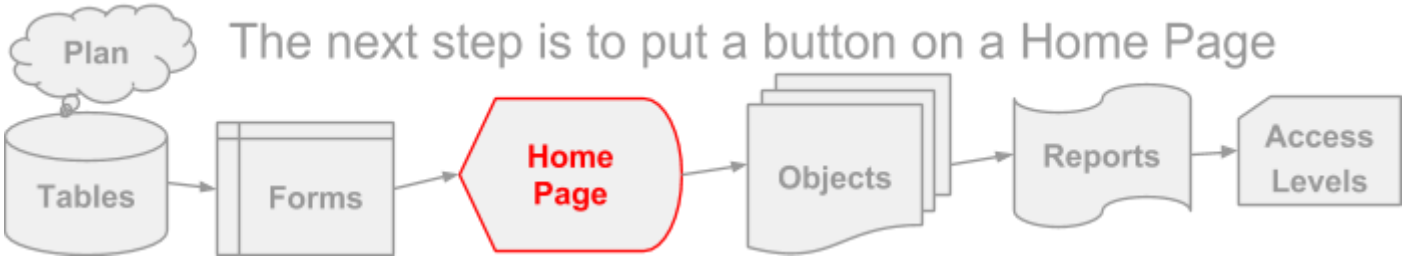
You have now completed all of the initial steps to create a **Form** in nuBuilder. A **Form** provides the information for nuBuilder to link a table in your database to both a **Search Screen** and an **Edit Screen**.

	The information that you entered under **Browse Tab** on the previous step; will determine the columns names and widths in the **Search Screen** ⟶ *Figure 31.*

Figure 31.


SQL
SELECT * FROM course_organization
Columns
Title
Display
Align
Format
Width
Order
Code
org_code
Center
80
10
Name
org_name
Left
250
20
Phone
org_office_number
Right
120
30
Email
org_office_email
Left
400
40
Home ▸ Home ▸ Organization
Code
Name
Phone
Email
$ Dui Fusce Corp. (05) 9311 8565 mauris.elit@duiaugueeu.co.uk
$ Libero PC (03) 2197 0476 nec@Mauris.ca
$ Sapien Institute (05) 5243 6394 et.nunc.Quisque@luctus.edu
$ Lorem Sit Amet Associates (03) 4200 3329 Donec.vitae@ascelerisquesed.co.uk
$ Donec Corporation (09) 8905 8897 volutpat.Nulla.dignissim@Quisque.co.uk
$ Tincidunt Orci Quis Foundation (04) 1527 8591 consequat.purus.Maecenas@liberoMorbiaccumsan.net
$ Elit Pretium Et Corp. (07) 2869 3510 sem.Nulla@ultricesDuis.edu
$ Volutpat Incorporated (01) 2959 1367 ultrices.posuere.cubilia@duinecuma.com

Steps to put a Button onto the Home Page



Now that we have a **Form**, we need a way to get to that **Form**. We will now place a **Button** on the default **Home Page**. As per the **Two Screen** pattern mentioned earlier, this **Button** will load a **Search Screen** which can then be used to load an **Edit Screen**.

The default **Home Page** that is used by **End Users** is built under the green 'User Home' button.

1. Log into nuBuilder with your **globeadmin** account
2. Click on the **Objects** button ⟶ *Figure 32.*

Figure 32.

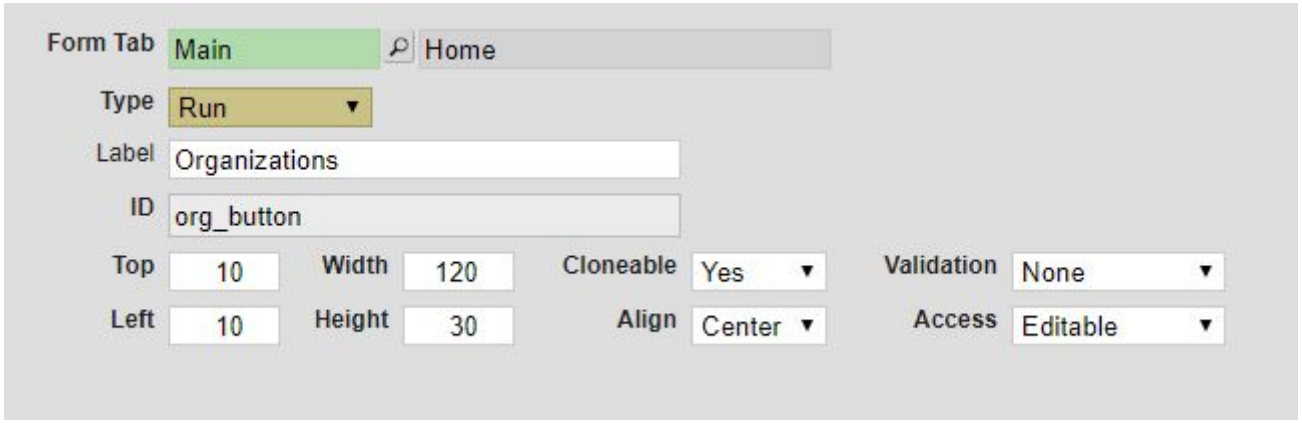
3. Click **Add** button.

As you can see on *Figure 33,* there are eleven **Tabs** on on the **Objects Screen**.
At this stage we only need to look at the following two **Tabs**; 'All' and 'Run'.

Figure 33.

4. On the **All Tab**, fill in the following items ⟶ *Figure 34.*

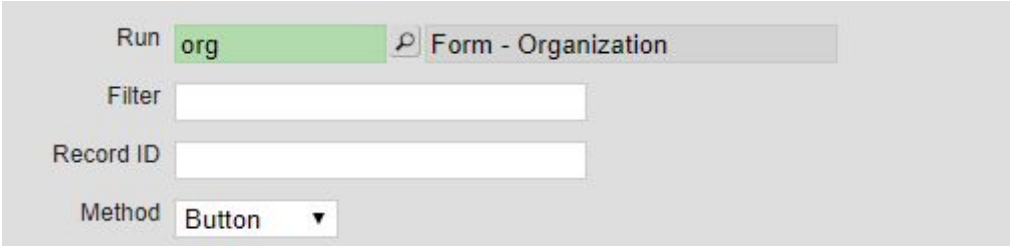
Figure 34.



Form Tab	Main	All **Objects** need to be linked to a **Form**. This tells nuBuilder where to display this **Object**. We are going to display this **Object** on the default **Home Page** screen. **Main** is the title of the **Tab** that this **Object** will be grouped under
Type	Run	Select the type of **Object** to create. In this case it is **Run** to run the form we want to display
Label	Organization	In the case of a button **Object**, the title will be displayed on the **Button**
ID	org_btn	All **Objects** need an ID, it must not contain spaces
Top	10	Spacing of the **Object** from the top of the screen
Left	10	Spacing of the **Object** from the left side of the screen
Width	120	120 is a typical starting value for width, you can adjust this according to your needs for each **Object**
Height	30	30 is a typical starting value for height, you can adjust this according to your needs for each **Object**
Align	Center	**Objects** which are displayed as buttons look best if they have their alignment set in the center

5. On the **Run Tab**, fill in the following item ⟶ *Figure 35.*

Figure 35.



Run	`org`	Select the **Form** that we created previously
Method	`Button`	Type of **Object** to be created to run the **Form**

6. Click **Save**

You have now completed all of the steps to create a **Button** to launch a **Form**.

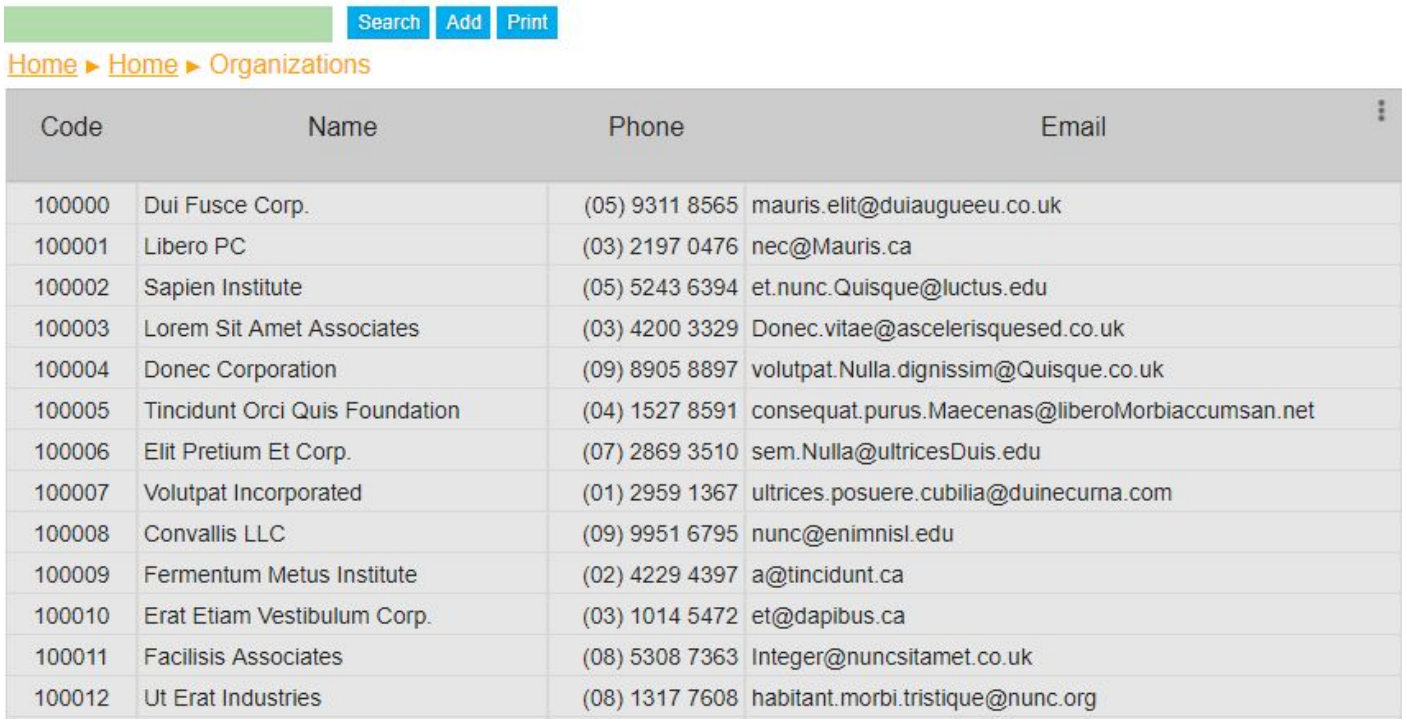
To see this in action, click on the green 'User Home' button.

You should see a screen as below ⟶ *Figure 36.*

Figure 36.

When you click on the button labeled 'Organizations' you will see the **Search Page** that we configured earlier in the **Forms** section. ⟶ *Figure 37.*

Figure 37.



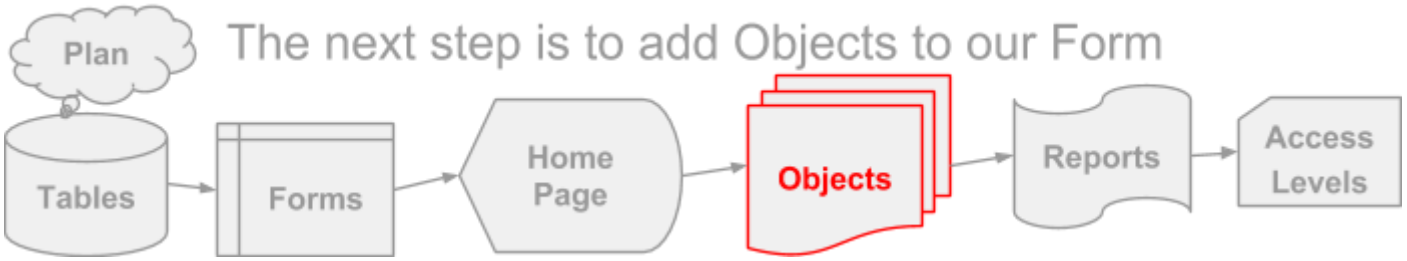
Code	Name	Phone	Email
100000	Dui Fusce Corp.	(05) 9311 8565	mauris.elit@duiaugueeu.co.uk
100001	Libero PC	(03) 2197 0476	nec@Mauris.ca
100002	Sapien Institute	(05) 5243 6394	et.nunc.Quisque@luctus.edu
100003	Lorem Sit Amet Associates	(03) 4200 3329	Donec.vitae@ascelerisquesed.co.uk
100004	Donec Corporation	(09) 8905 8897	volutpat.Nulla.dignissim@Quisque.co.uk
100005	Tincidunt Orci Quis Foundation	(04) 1527 8591	consequat.purus.Maecenas@liberoMorbiaccumsan.net
100006	Elit Pretium Et Corp.	(07) 2869 3510	sem.Nulla@ultricesDuis.edu
100007	Volutpat Incorporated	(01) 2959 1367	ultrices.posuere.cubilia@duinecurna.com
100008	Convallis LLC	(09) 9951 6795	nunc@enimnisl.edu
100009	Fermentum Metus Institute	(02) 4229 4397	a@tincidunt.ca
100010	Erat Etiam Vestibulum Corp.	(03) 1014 5472	et@dapibus.ca
100011	Facilisis Associates	(08) 5308 7363	Integer@nuncsitamet.co.uk
100012	Ut Erat Industries	(08) 1317 7608	habitant.morbi.tristique@nunc.org

Now click on any of one of the records listed in the **Search Screen**, you should see the following **Edit Screen**. ⟶ *Figure 38.*

Figure 38.

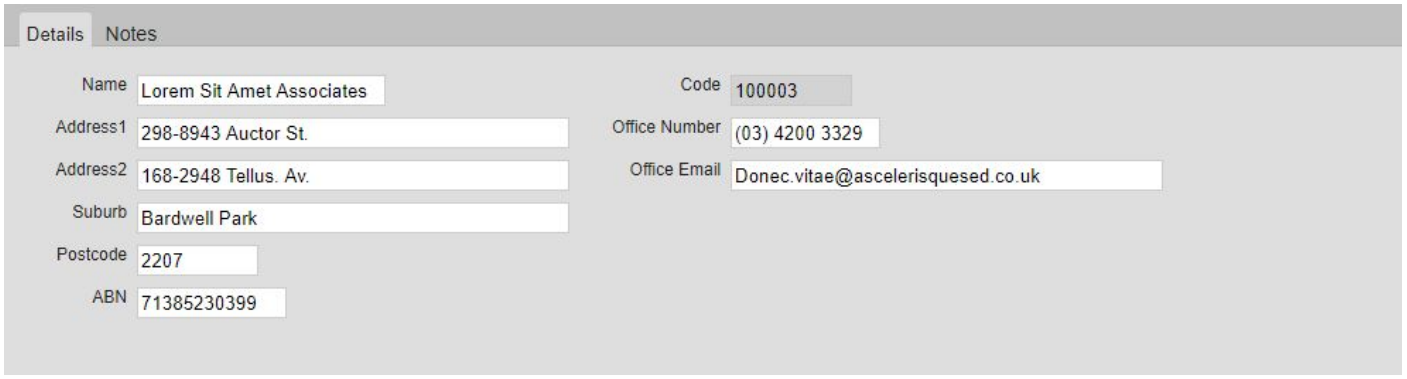
The **Edit Screen** is blank because we have not created any **Objects** to go on it yet, we will do that in the next step.

Steps to add Objects to an Edit Screen



We are now going the take the **Form** which is shown in *Figure 38* and make it complete by adding **Objects** to it. When it is complete, it will then look like the following **Screen** ⟶ *Figure 39.*

Figure 39.



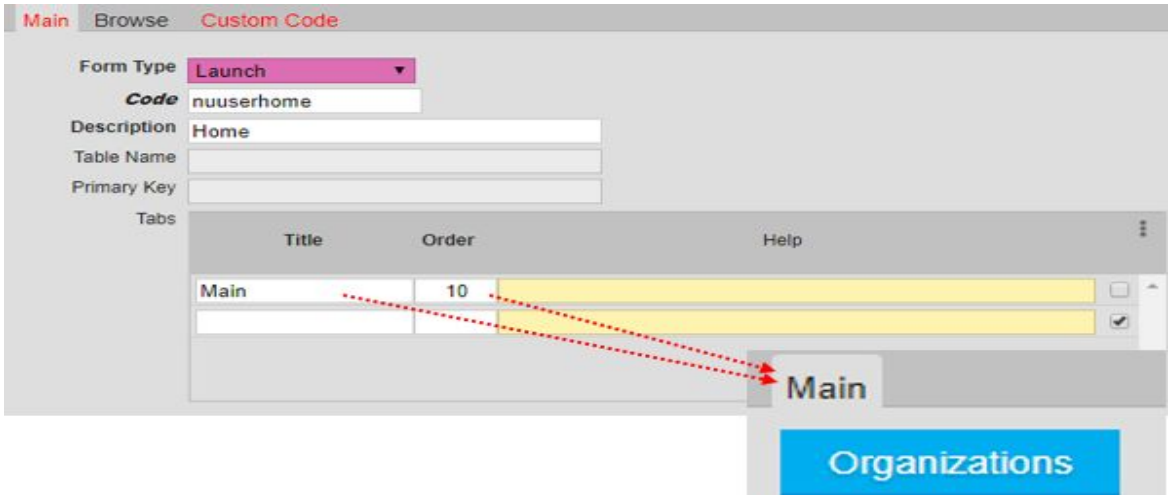
Before we delve into building the above **Screen** we need to first take a look at how **Tabs** are created and how to control the placement of **Objects** on a **Screen**.

Creating Tabs

Tabs are automatically created by nuBuilder when you build **Objects**.

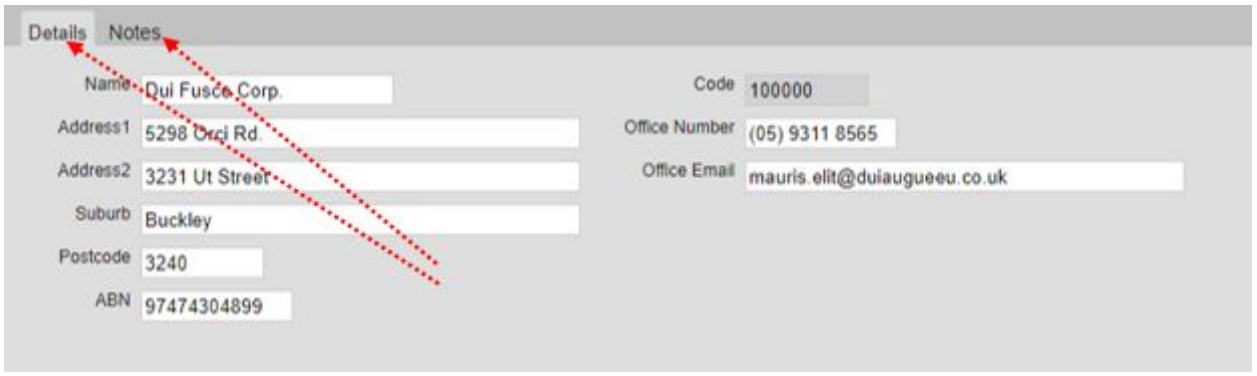
If you recall on the **Objects Screen** earlier, when we created our first **Object**, which was a **Button** to launch a **Form**, nuBuilder is provided a default **Form Tab** of '`Main`' and a **Tab Number** of '`10`'. This was all of the information needed for nuBuilder to automatically create the **Tab** as seen in ⟶ *Figure 40.*

Figure 40.



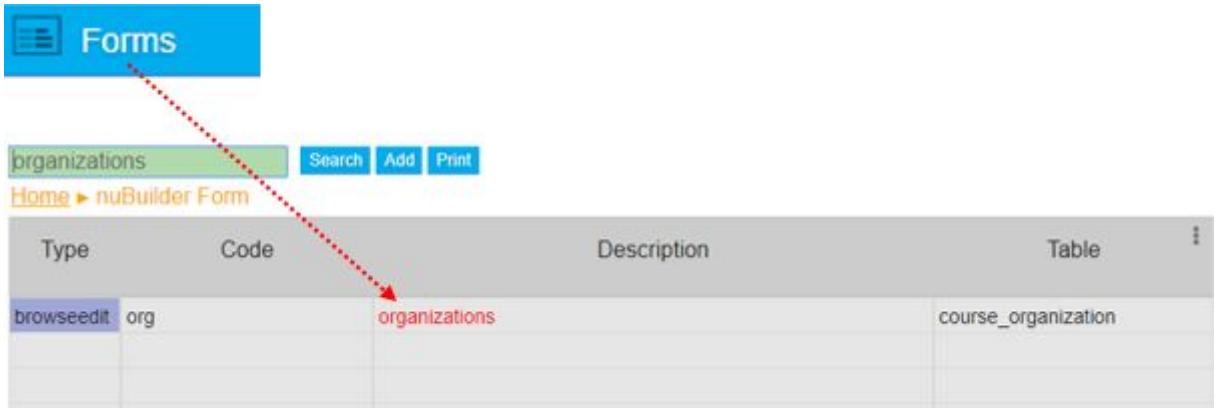
On *Figure 41* we can see that we are going to be building two **Tabs**. `Details` and `Notes`.

Figure 41.



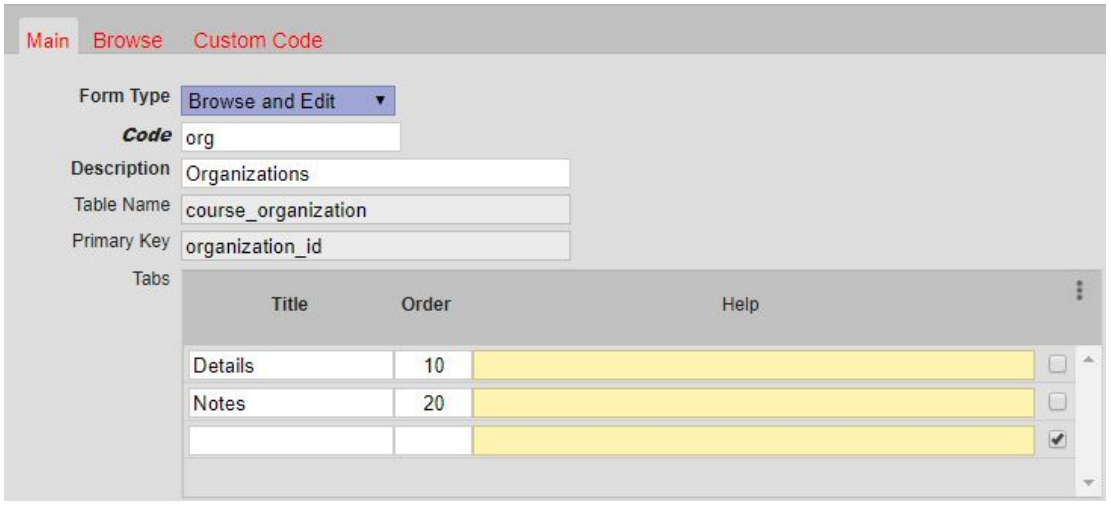
1. Log into nuBuilder with your **globeadmin** account
2. Click on the **Forms** button in the Setup tab and click on the organizations form → Figure 42

Figure 42.



To create the two **Tabs**, we will enter the label '`Details`' for the **Tab Title** and the number '`10`' for the **Tab Order Number.** We will then enter the label '`Notes`' for the **Tab Title** and the number '`20`' for the **Tab Order Number** → *Figure 43.*

Figure 43.



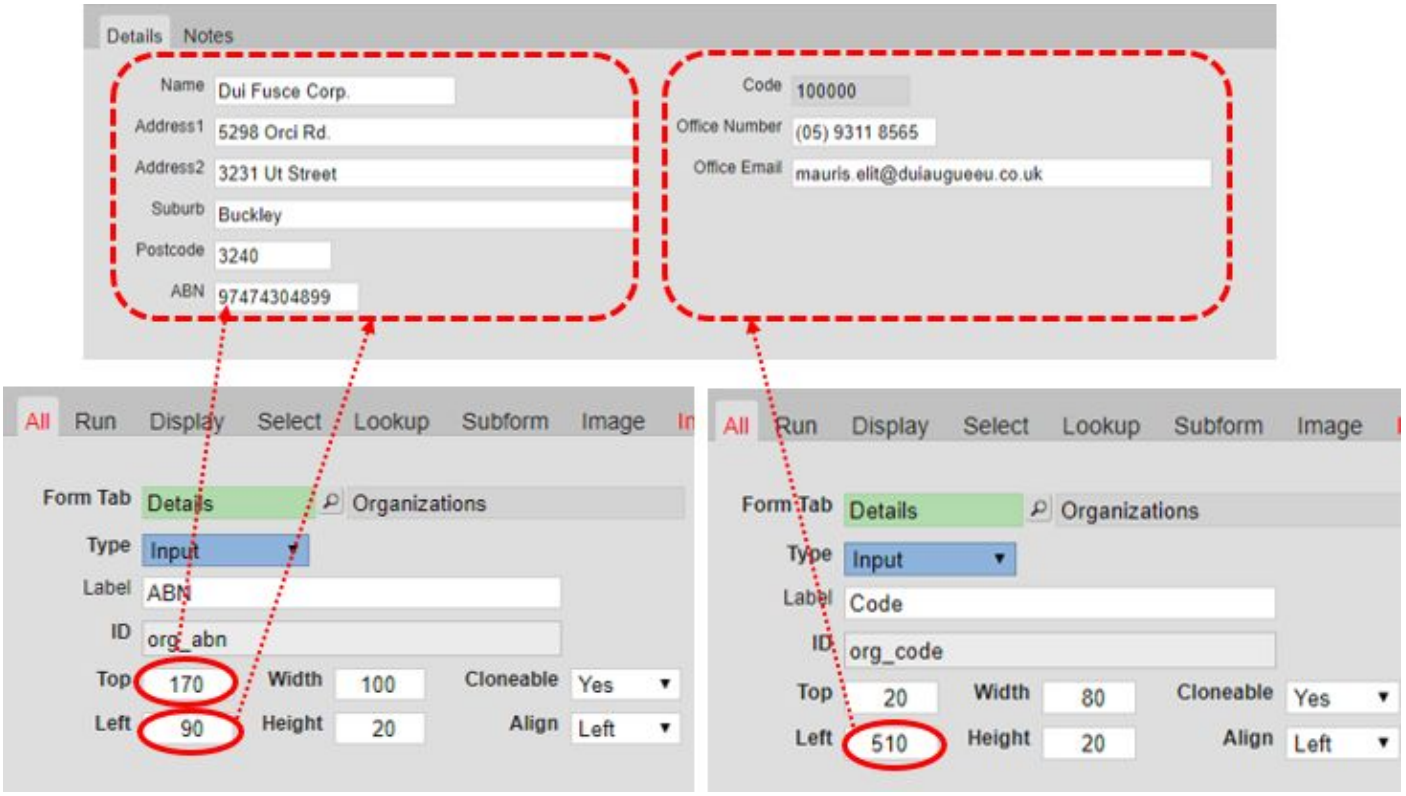
This will become clear when we go through the process of creating each of these **Objects**.

Controlling Object Placement

Notice on → *Figure 44* that there are two columns of **Objects**. All of the **Objects** on the left side have a **Left** spacing of '90'. All of the **Objects** on the right side have a **Left** spacing of '510'.

We can also control the order that each **Object** displayed in each column. The 'ABN' field is displayed last on the left column because it has the highest **Top** alignment of '170'.

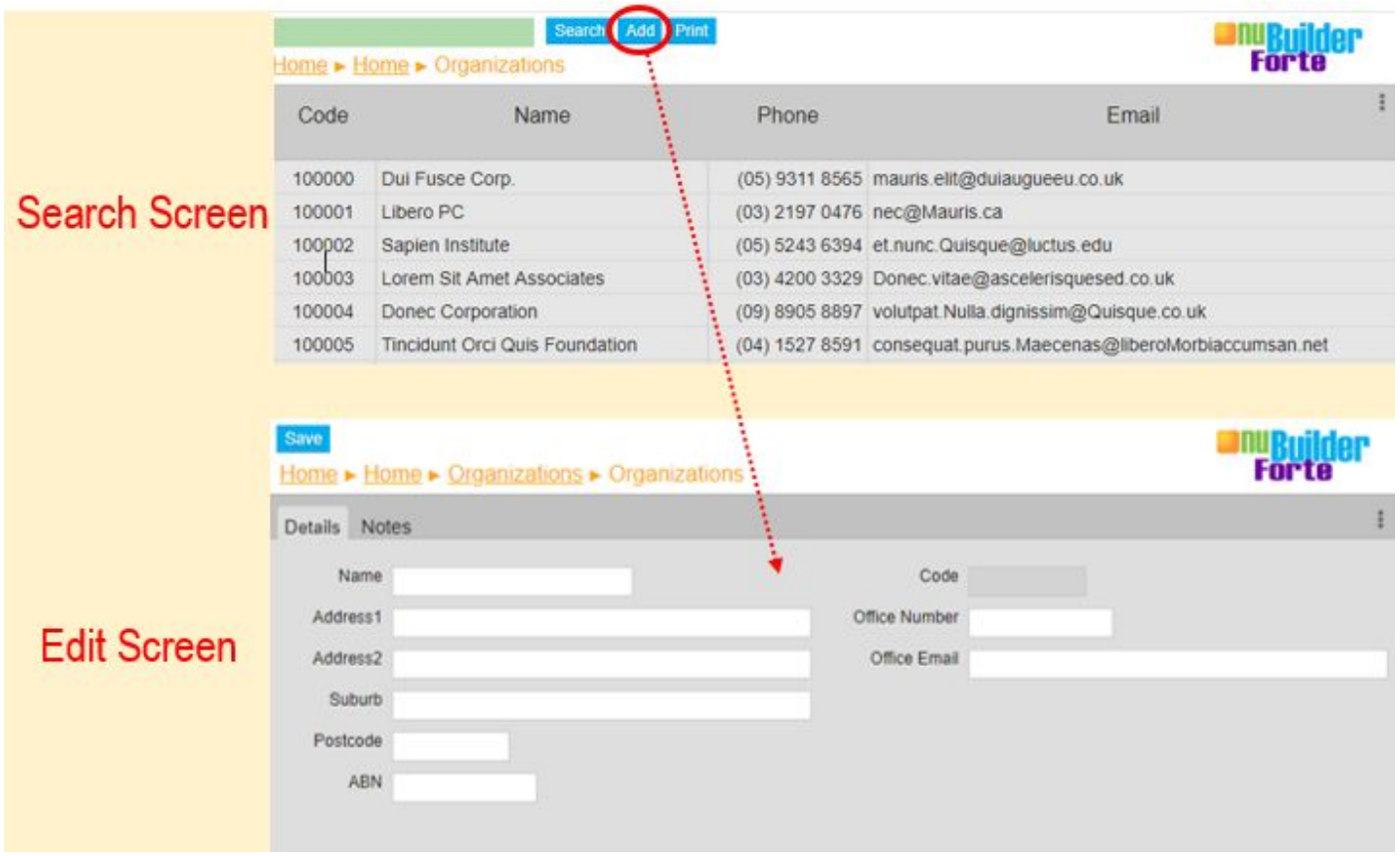
Figure 44.



Another way we can control the object placement is to manually drag the **Objects** onto the preferred location on the **Form**.

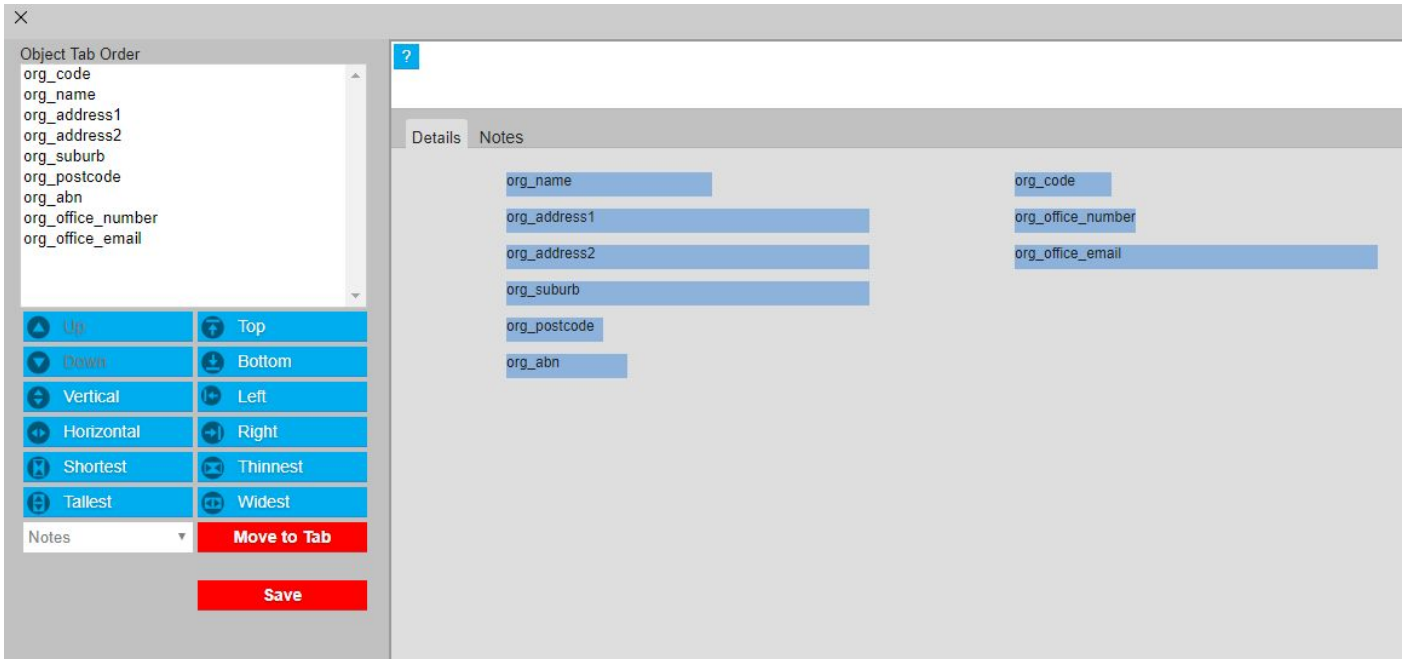
1. Log into nuBuilder with your **globeadmin** account
2. Click on the **User Home** button in the **Setup Tab**
3. Click on the Organizations button we have created earlier
4. On the **Search Screen,** click on the Add button. This will display **Edit Screen** → Figure 45

Figure 45.



5. Click on the **Options**  button (on the top right of the **Edit Screen**) → Arrange Objects
6. Click and drag the **Objects** to arrange them on the **Edit Screen** → Figure 46

Figure 46.



7. When you've finished arranging the **Objects**, click Save

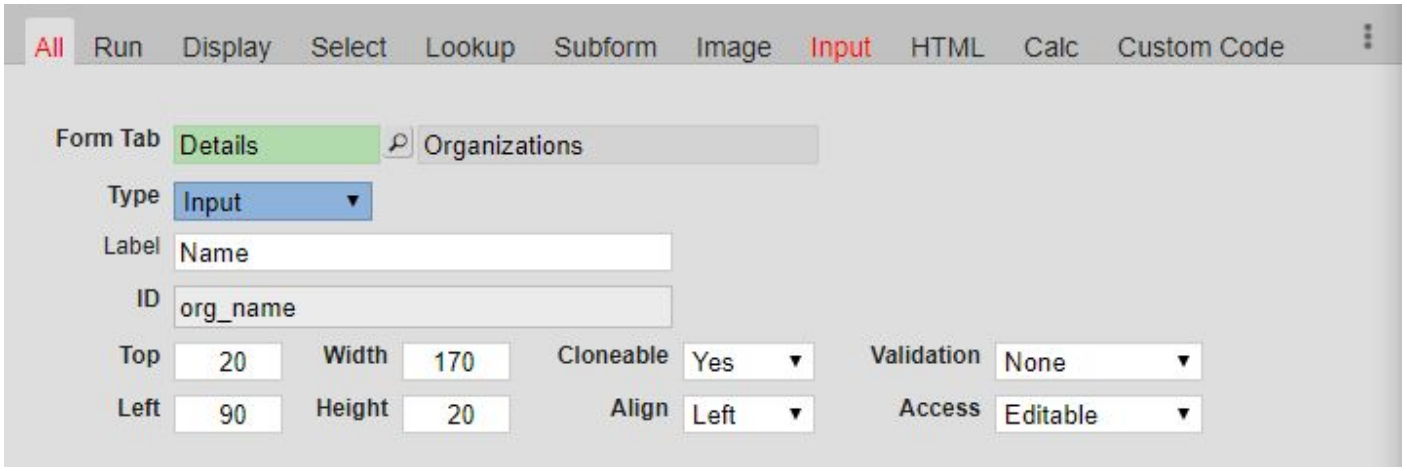
8. Click on the **Options** button again → Refresh or click on **Save** to see the new arrangement of the **Objects** on the form

Building Text Input Objects

The next **Object** that we will create will be a **Text Input** to store the name of an organization.

1. Log into nuBuilder with your **globeadmin** account
2. Click on the **Objects** button
3. Click **Add** button
4. On the **All Tab**, fill in the following items → *Figure 47.*

Figure 47.



Form Tab	Details - Organizations	All **Objects** need to be linked to a **Form Tab** and **Form**. This tells nuBuilder where to display this **Object**.
Type	Input	Select the type of **Object** to create
Label	Name	In the case of a text **Object**, the label (title) will be displayed next to the Input box
ID	org_name	All **Objects** need an ID, it must not contain spaces
Top	20	Spacing of the **Tab** from the top of the **Edit Screen.** You can adjust this according to your needs for each **Object**
Left	90	Spacing of the **Tab** from the left side of the **Edit Screen.** You can adjust this according to your needs for each **Object**
Width	170	170 is a typical starting value for width, you can adjust this according to your needs for each **Object**

Height	`20`	20 is a typical starting value for height, you can adjust this according to your needs for each **Object**
Align	`Left`	The alignment of the text **Input** is left by default.
Access	`Editable`	Allows text **Input** to be edited on the **Edit Screen**

5. Click **Save**

You have now completed all of the steps to create a **Text Input** to store information.

	Whenever you enter **IDs, Table Names** and **Primary Keys**; make sure your haven't accidently placed a hidden space as the last character of the input.

Choosing IDs

IDs must be unique for each **Form** they are connected to. An **ID** must not contain spaces.

In order to have information on the screen saved to the database, you must use an **ID** that matches an actual column in the table which is connected to the **Form**.

When we created a **Button Object** previously, we used the following **ID**: '`org_btn`'. There isn't a field in the corresponding table that matches that value, and there does not need to be a matching field because **Buttons** are not used for storing information. Whenever you need to create **Objects** that do not write to the table connected to the **Form**, you can choose any unique **ID** you want to use.

However when we created the **Text Input Object,** we used the following **ID**: '`org_name`'. If you refer back to the SQL used to create the organization table, you will see that this matches a column in that table.

The following **Objects** need to have **IDs** which match actual columns in the table connected to the **Form**

Basic Input	Listing
Input	Select
Textarea	Lookup

The following **Objects** do not need to have **IDs** which match actual columns in the table connected to the **Form**

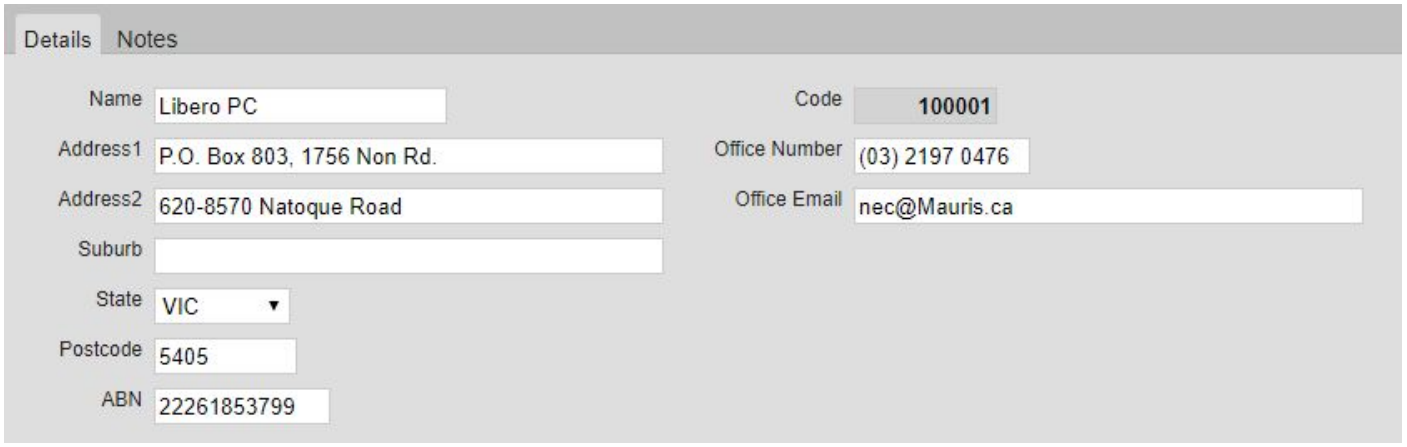
Actions	One to Many Relation	Cosmetic/ Calculated	Mashing
Run	SubForn	Word	HTML
		Display	

Repeat the previous five steps to continue creating the following **Text Input Objects**, using the information provided below.

Label	Type	Form Type	ID	Top	Left	Width	Height	Align	Access
Address1	Input	Details	org_address1	50	90	300	20	Left	Editable
Address2	Input	Details	org_address2	80	90	300	20	Left	Editable
Suburb	Input	Details	org_suburb	110	90	300	20	Left	Editable
Postcode	Input	Details	org_postcode	140	90	80	20	Left	Editable
ABN	Input	Details	org_abn	170	90	100	20	Left	Editable
Office Number	Input	Details	org_office_number	50	510	100	20	Left	Editable
Office Email	Input	Details	org_office_email	80	510	300	20	Left	Editable

After completing these steps, your **Edit Screen** should look like the following. ⟶ *Figure 48.*
We still need to add a **Dropdown Object,** an **Autonumber Object** and a **Textarea Object**.

Figure 48.



Building Dropdown Objects

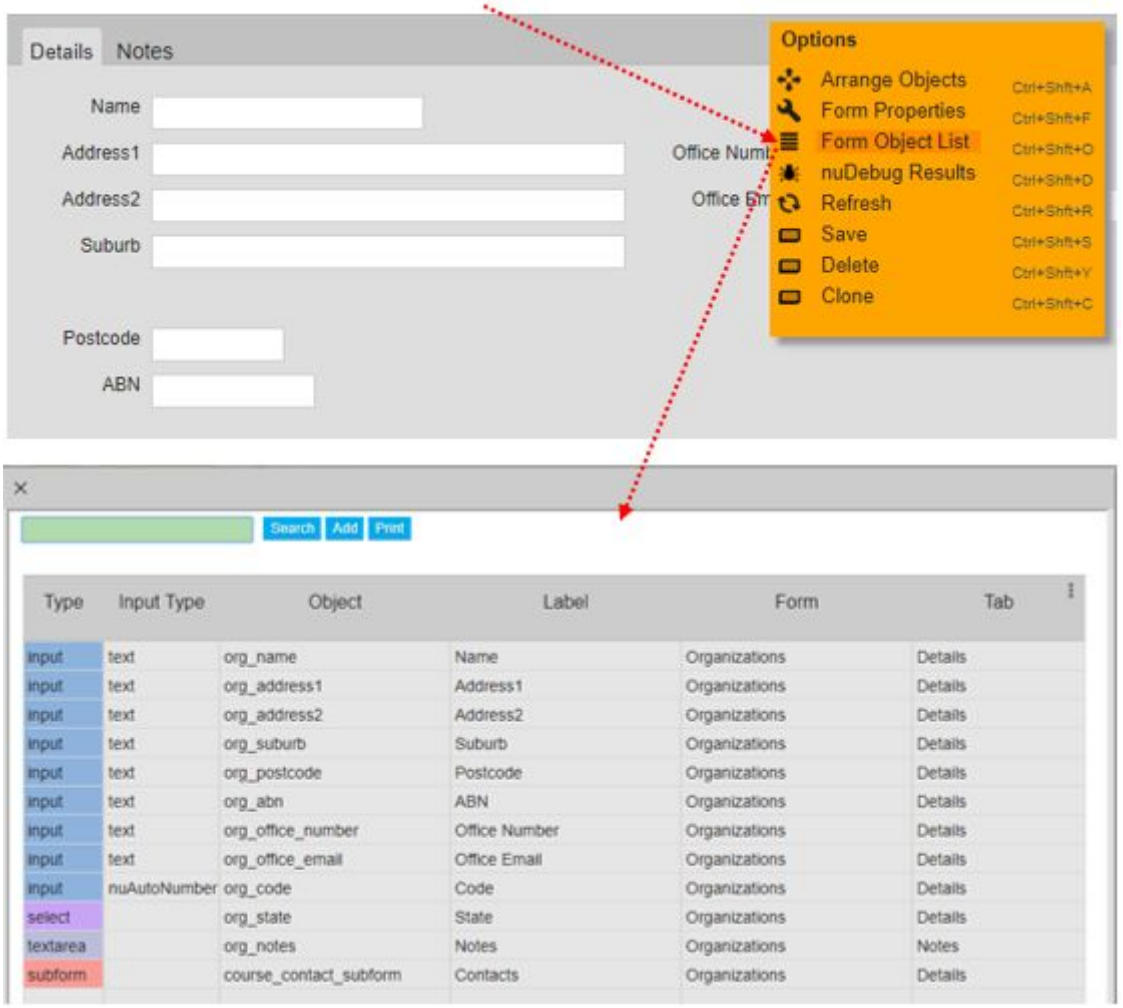
Alternative Method to Create an Object

The next **Object** that we will create will be a **Dropdown** to store the different states of Australia. This time, we will use an alternative way to create this **Object** on the **Form** we have created

1. Log into nuBuilder with your **globeadmin** account
2. Click on the green 'User Home' button, then click on 'Organizations' and then 'Add Record'.

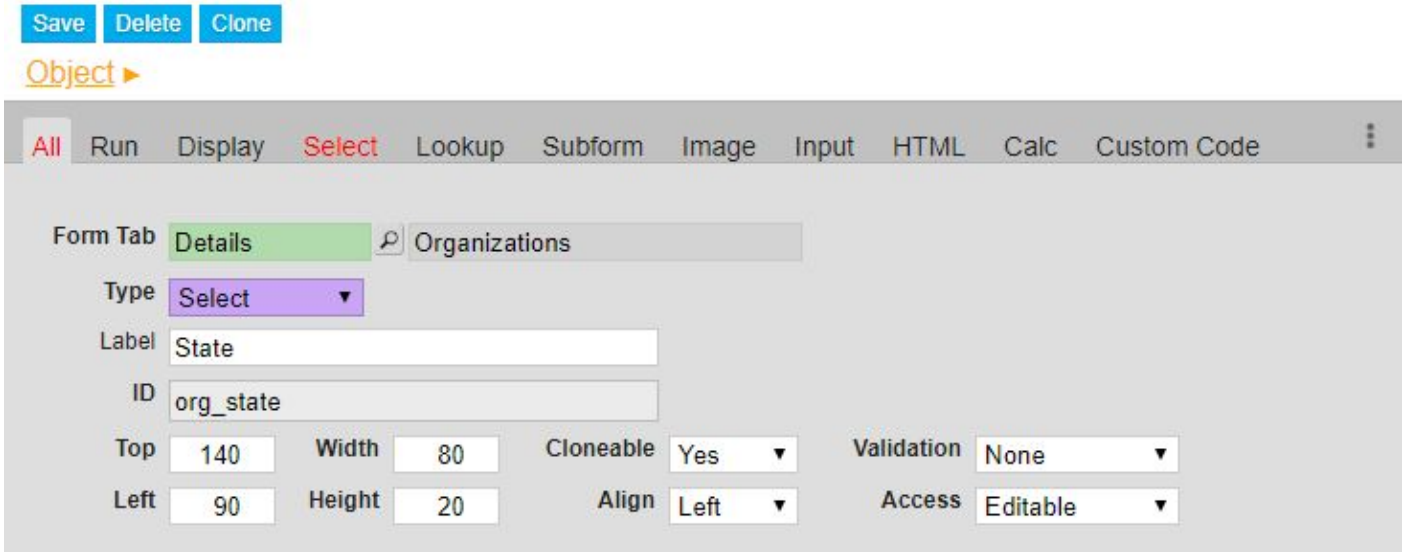
3. On the **Edit Screen** that appears, you can click on the **Options**  button on the top right of the **Form** and click 'Form Object List' this will shows all of the **Objects** connected the **Form** shows all of the **Objects** connected the **Form** we have created earlier . ⟶ *Figure 49.*

Figure 49



4. Click **Add** button.
5. On the **All Tab**, fill in the following items ⟶ *Figure 50.*

Figure 50.



Form Tab	Details - Organizations
Type	Select
Label	State
ID	org_state
Top	140
Left	90
Width	80
Height	20
Access	Editable

5. On the **Select Tab**, fill in the following ⟶ *Figure 51.*

Figure 51.

SQL	NSW|NSW|VIC|VIC|QLD|QLD|WA|WA|SA|SA|TAS|TAS|ACT|ACT|NT|NT

6. Click **Save**

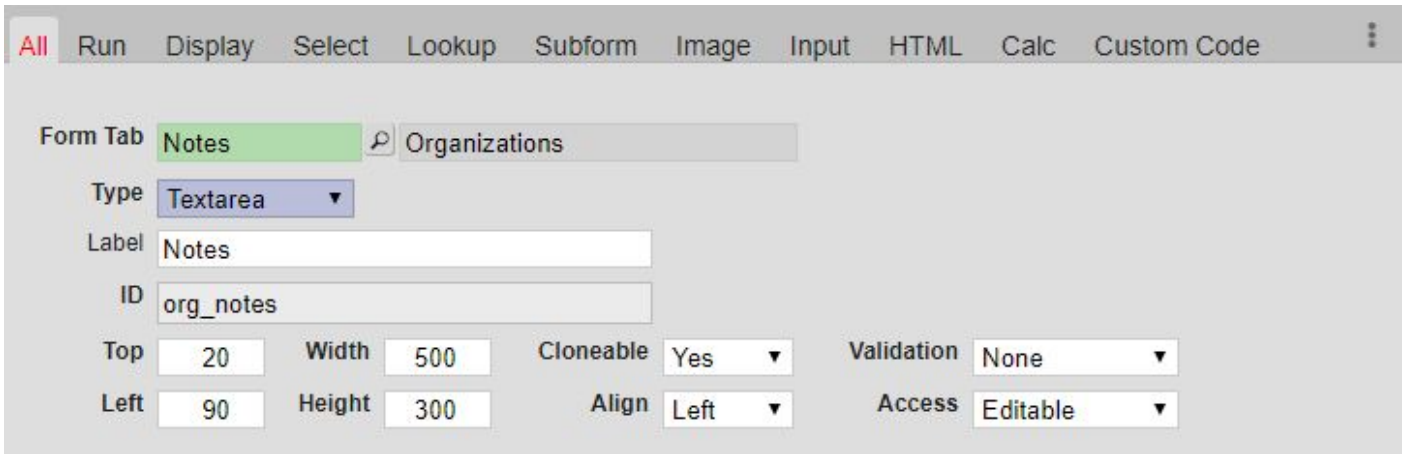
You have now completed all of the steps to create a **Dropdown Object**.

Building Textarea Objects

The next **Object** that we will create will be a **Textarea** to store notes.

1. Log into nuBuilder with your **globeadmin** account
2. Click on the **Objects** button
3. Click **Add** button
4. On the **All Tab**, fill in the following items ⟶ *Figure 52.*

Figure 52.



Form Tab	Notes - Organizations
Type	Textarea
Label	Notes
ID	org_notes
Top	20
Left	90
Width	500
Height	300
Align	Left
Access	Editable

5. Click **Save**

You have now completed all of the steps to create a **Textarea Object**.

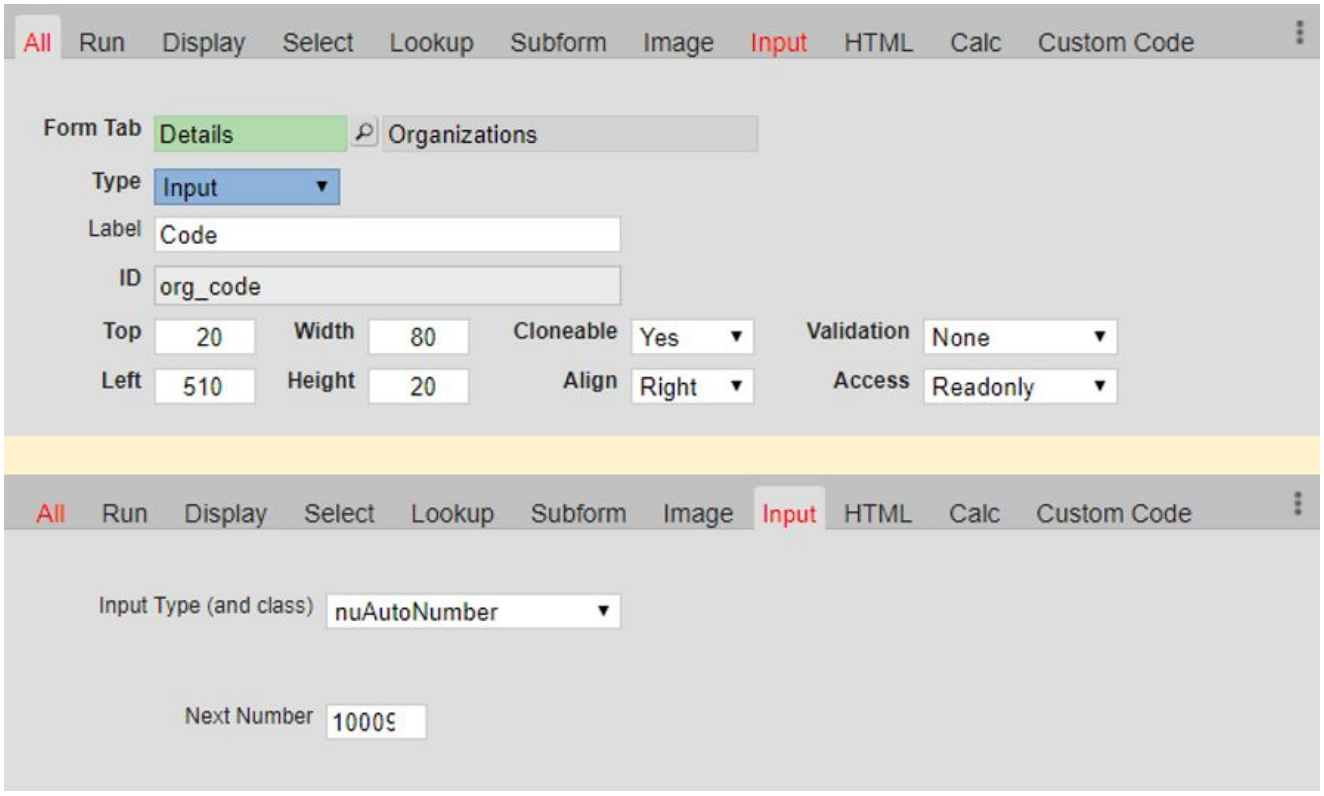
Building Autonumber Objects

The final **Object** that we will create to complete our **Edit Screen** will be an **Autonumber Object**.

Autonumber Objects are actually just **Text Input Objects** with a few extra settings added.

1. Log into nuBuilder with your **globeadmin** account
2. Click on the **Objects** button
3. Click **Add** button
4. On the **All Tab** and the **Input Tab**, fill in the following items ⟶ *Figure 53.*

Figure 53.



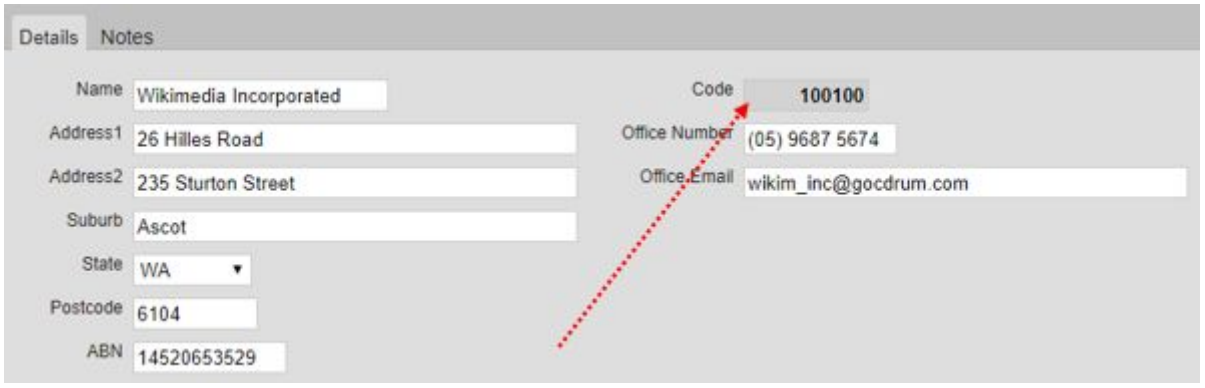
Align	`Right`	**Objects** which are used for displaying numbers look best if they have their alignment set to the right
Access	`Readonly`	**Objects** which are automatically populated should be set to read only
Input Type (and class)	`nuAutoNumber`	Nubuilder has an inbuilt autonumber input type to automatically generate a new number when a new record is inserted
Next Number	`100099`	Type in the last **org_code** record in the form. When a new record is inserted, the **org_code** will automatically increase by 1 from this number.

5. Click **Save**

You have now completed all of the steps to create an **Autonumber Object**.
To see this in action, click on the green 'User Home' button, then click on 'Organizations' and then 'Add Record'.

As you can see in *Figure 54,* once you have entered a new record and saved it, a new number will be generated.

Figure 54.



Edit Object Shortcuts

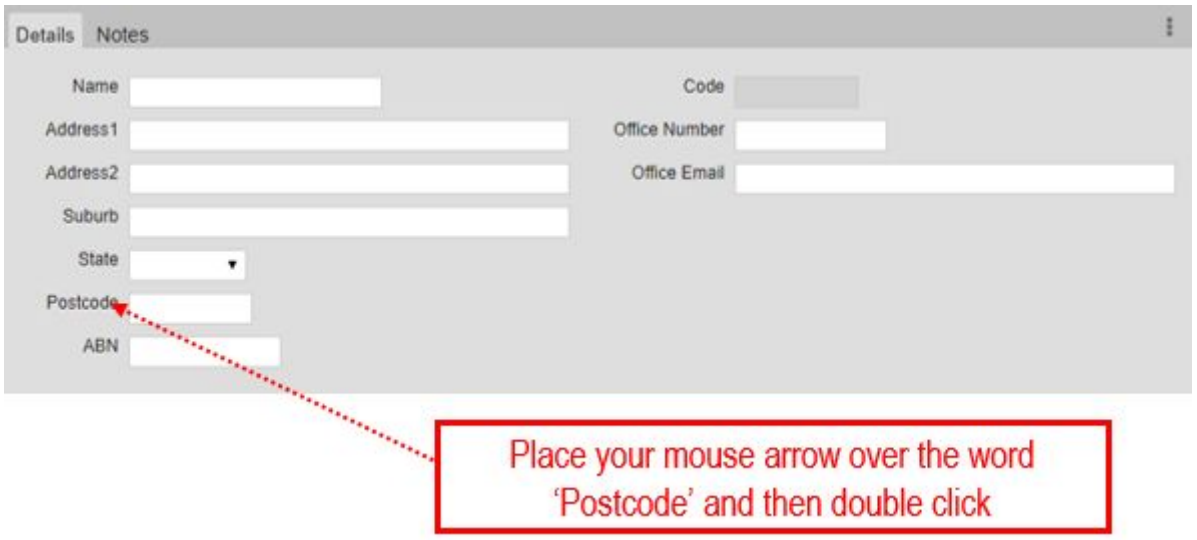
Up until now we have always followed the steps of clicking on the **Objects Button** and then clicking the **Add Button** whenever we needed to create **Objects**. However we can skip these steps by using some of the built-in shortcuts.

As an example, we will now edit the 'Postcode' field that we created earlier. The Postcode field is designed to contain numbers, therefore it is best to have the alignment set to the right and set the input formatter to mask all other characters other than numbers. To do this we can follow the following steps:

Click on the green 'User Home' **Button**, then click on 'Organizations' and then 'Add'.

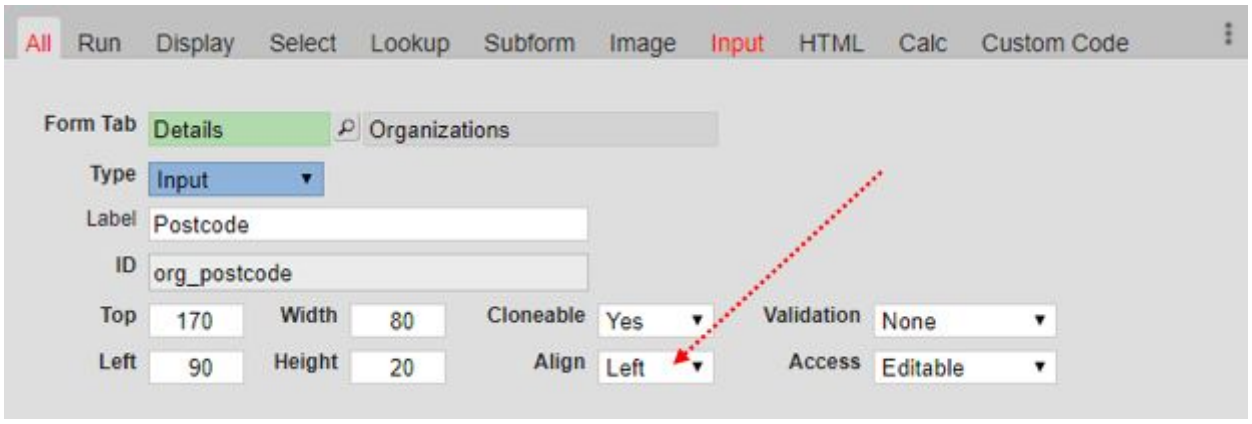
After you double click the title of an **Object,** a new web browser tab will open, taking you directly to the **Objects Screen**. ⟶ *Figure 55.*

Figure 55.



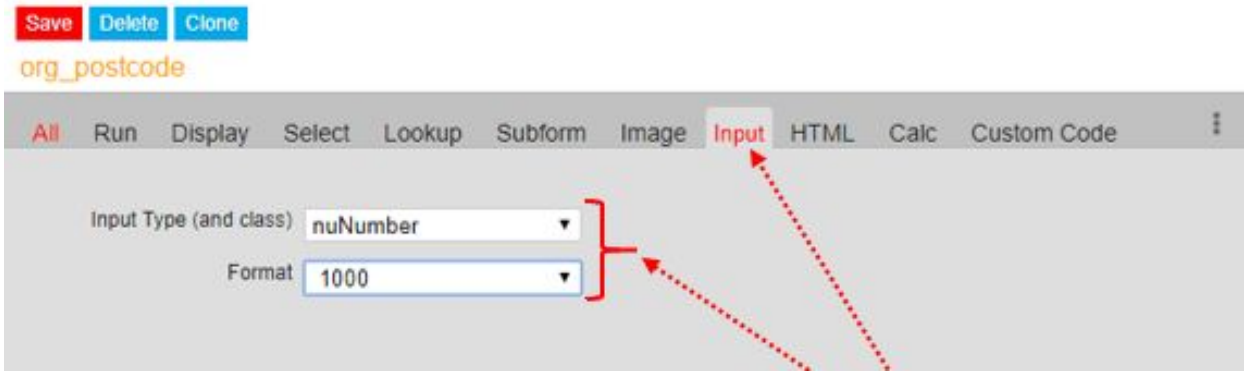
Now you can change the **Align** value. ⟶ *Figure 56.*

Figure 56.



Next, switch the to **Input Tab** and set the **Input Type** to '`nuNumber`' **Format** to '`1000`' and then click **Save**. ⟶ *Figure 57.*

Figure 57.



Edit Form Shortcuts

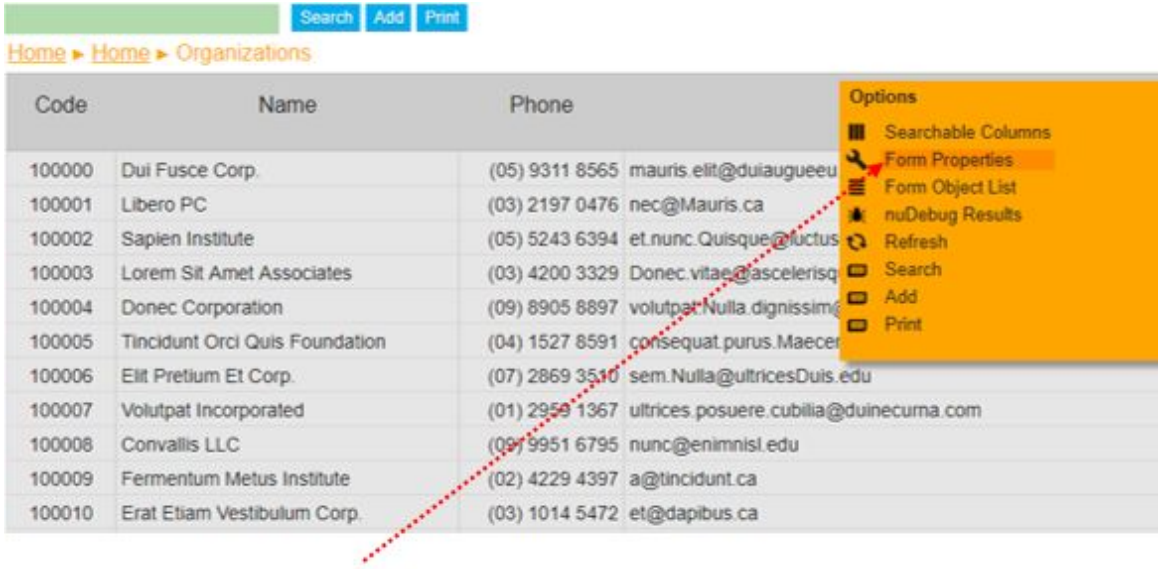
The is a shortcut link which provides quick access to the **Form Screen.**
Click on the green 'User Home' **Button**, then click on 'Organizations'.

On the **Search Screen** you can click on the **Options**  button on the top right of the **Form**. → *Figure 58.*

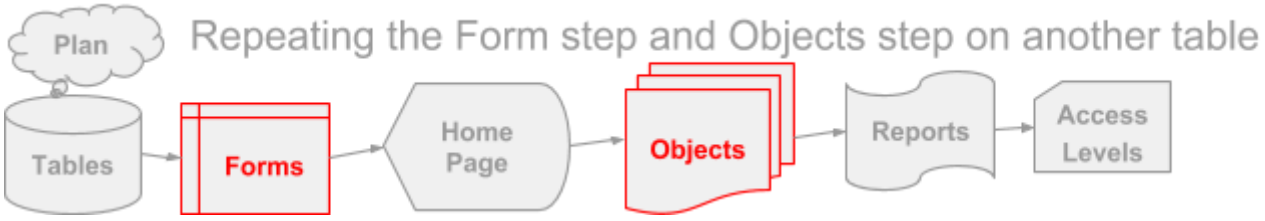
Figure 58.

Then click on **Form Properties** → *Figure 59.*

Figure 59.



Fast Form



To continue building our CRM application we now need to build a **Screen** to store information concerning each 'Contact'.

When it is complete, it will then look like the following **Screen** ⟶ *Figure 60.*

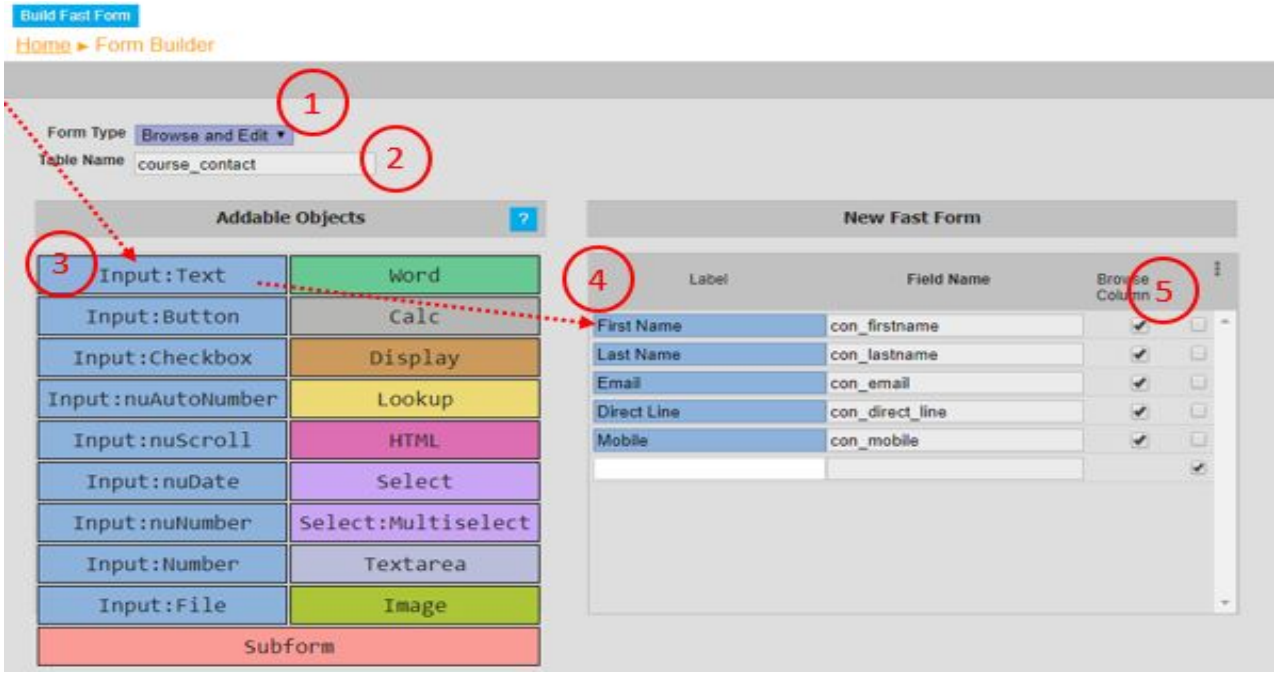
Figure 60.

We could go through all of the previous steps to build a new **Form** using the contacts table, however there is a faster way.

We are now going to build a **Form** using the **Fast Form**.

1. Log into nuBuilder with your **globeadmin** account
2. Click on the **Fast Form** button in the **Builders Tab** ⟶ *Figure 61.*

Figure 61.



1. Select the form type '`Browse and Edit`'
2. Type in the form type '`course_contact`'
3. Click the '`Input:Text`' **Button** (this is to add an object to the form)
4. Replace the **Label** text with '`First Name`' and **Field Name** '`con_firstname`'
5. Check the checkbox under **Browse Column**. This will display the object on the **Search Screen** of the **Subform**.

Repeat steps 3 to 5 to create the following **Text Input Objects**, using the information provided below.

Label	Field Name
`Last Name`	`con_lastname`
`Email`	`con_email`
`Direct Line`	`con_direct_line`
`Mobile`	`con_mobile`

6. Click the **Build Fast Form** button.

These are all of the steps needed to create a **Form**.

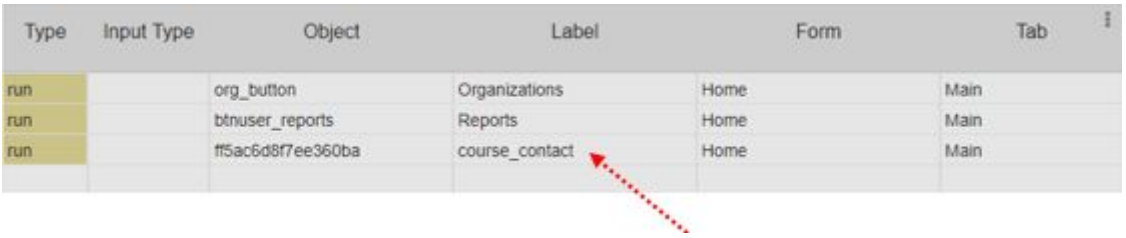
Note: we left off the field '`con_organization_id`' so that I can show in you in detail, all of the steps needed to make a **Lookup Object**.

To see this new **Form** in action click on the green 'User Home' **Button**.

All **Buttons** created by the **Fast Form** are placed on the default **Home Page** under a **Tab** called '`Fast Form`'. We will need to manually change the name of the **Tab** to '`Main`' We will also need to manually change the properties of this **Button Object** such as the **Button Label** from '`course_contact`' to '`Contacts`' and the **ID** to '`contact_btn`'

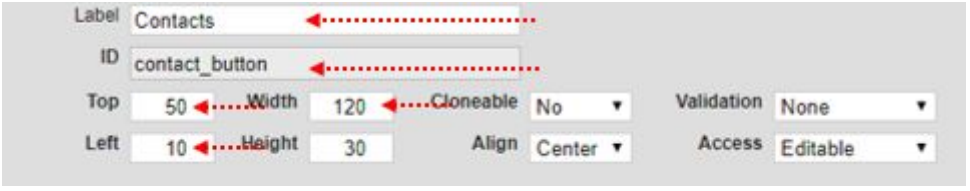
To do this, on the **User Home** screen, click on Options  → Form Object List to load a **Search Screen** which shows all of the **Objects** connected the **Form** ⟶ *Figure 62.*

Figure 62.



Click on the row which shows our new **Button Object** that we want to edit. ⟶ *Figure 63.*

Figure 63.



Label	`course_contact`	⟶	`Contacts`
ID	`ff5ac6f602d4d7680`	⟶	`contact_btn`
Top	`150`	⟶	`50`
Left	`150`	⟶	`10`
Width	`150`	⟶	`120`

Save the changes made to the **Button Object.**

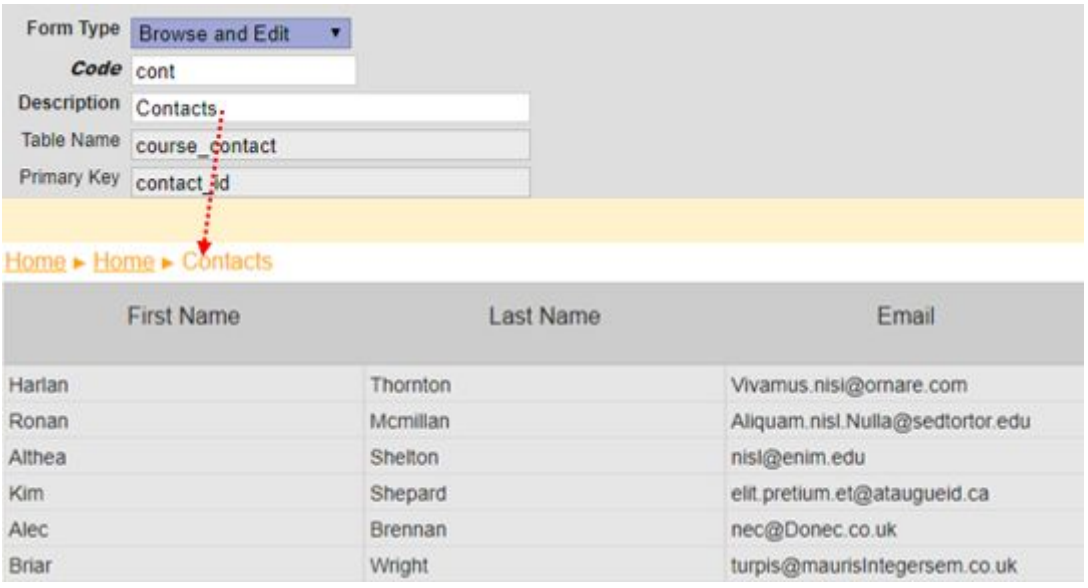
Note that instead of typing in the position of the **Object Button** in **Top** and **Left,** we can also manually move the button by dragging it to the position we want it to be on the form as mentioned previously.

Notice in Figure 64 that when you click on the **Contacts Button** we just created, the **Breadcrumb** is automatically named '`Fast Form()`'. → *Figure 64*

Figure 64.

We will go to the form properties to change the name of the **Breadcrumb** from '`Fast Form()`' to '`Contacts`'. → *Figure 65*

Figure 65.



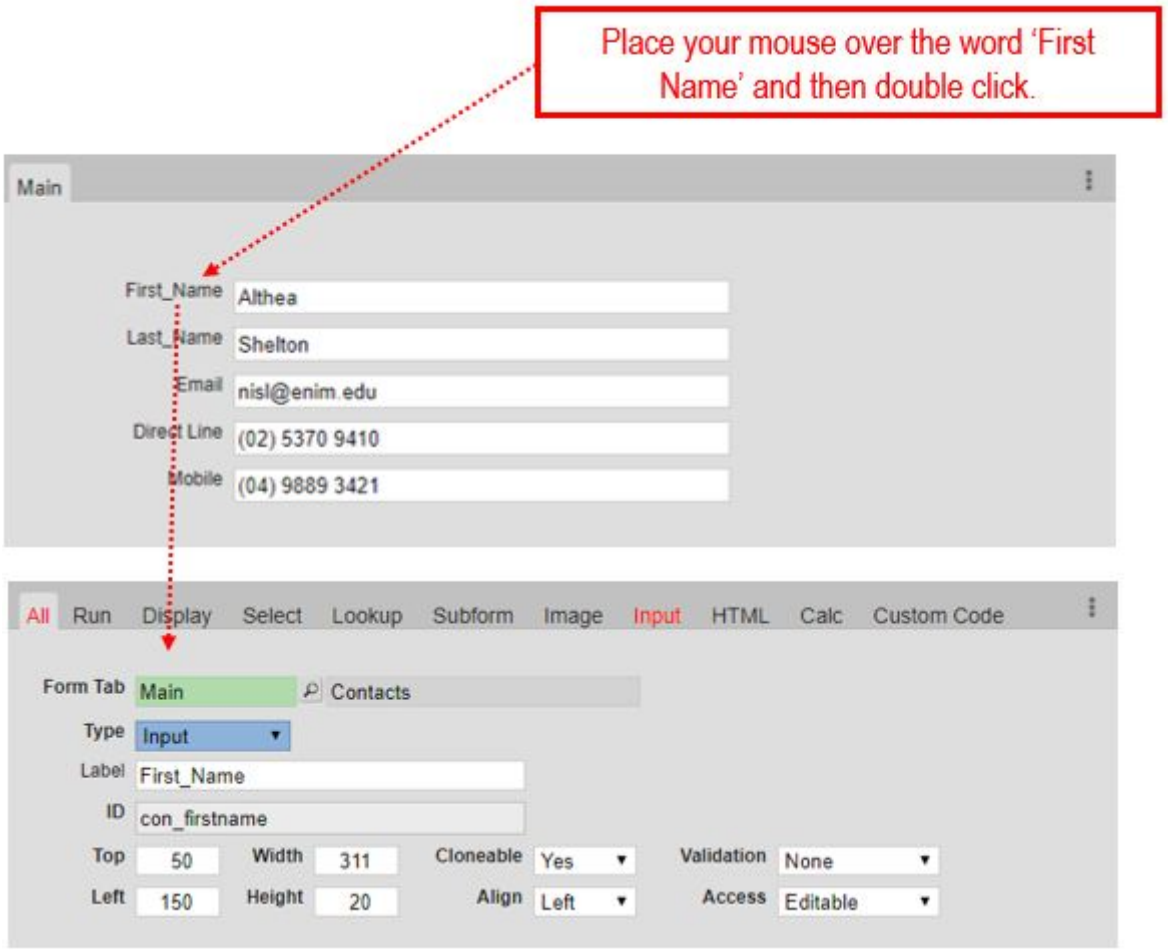
Code	`FF()`	→	`cont`
Description	`Fast Form()`	→	`Contacts`

Save the changes made in the **Form Properties.**

Object Search Shortcut

On an **Edit Screen** you can double click on the **Tab Tile** to load a **Search Screen** which shows all of the **Objects** connected the **Form** ⟶ *Figure 66.*

Figure 66.



Make your changes if needed and click **Save**.

You have now completed all of the steps to create a **Search Screen** and an **Edit Screen** for the contact table.

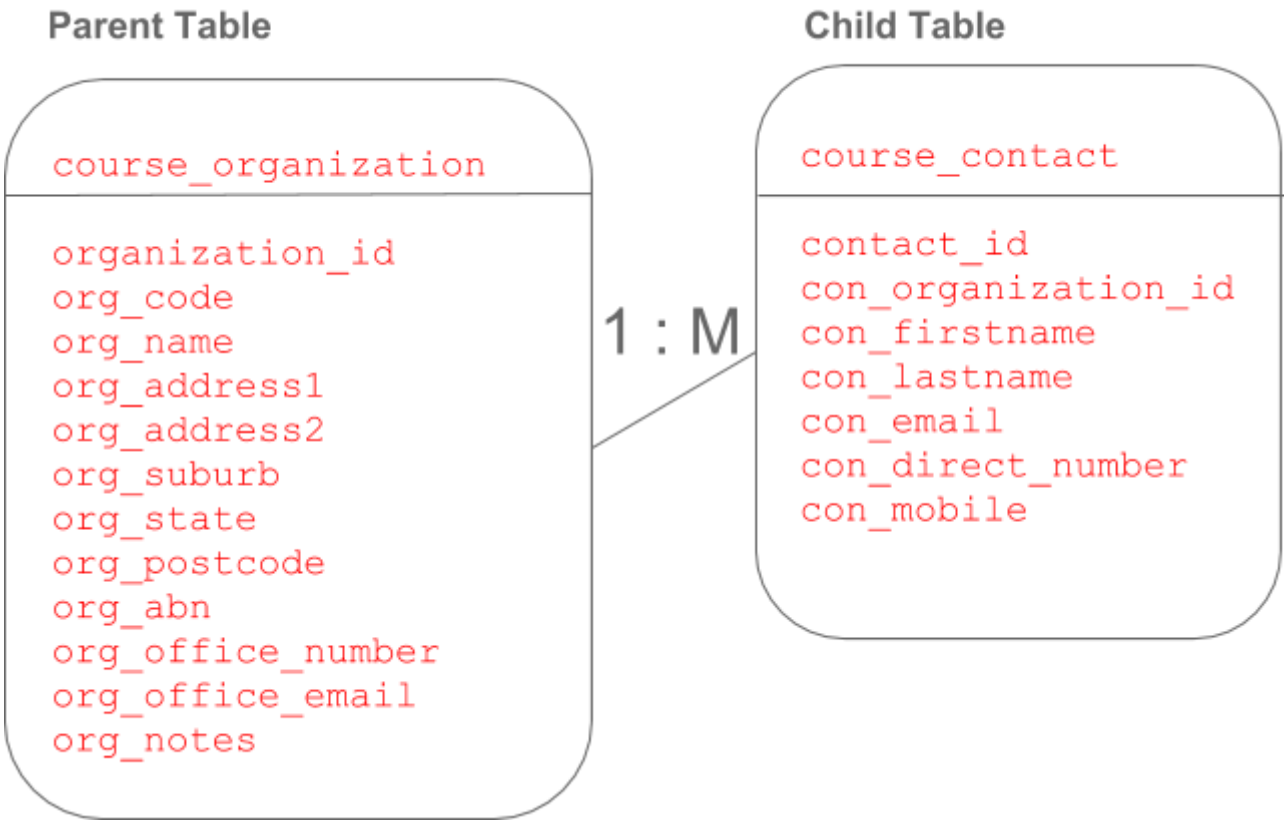
To see this in action, click on the green 'User Home' **Button**, then click on 'Contacts'.

Reminder: The **Description** field on the **Form Screen** is used in the display of the navigation **Breadcrumbs** and **Lookups**

Lookup Objects

In the 'course_contact' table, we have a foreign key called 'con_organization_id'. In this example the 'course_organization' table can be referred to as the **Parent Table** and the 'course_contact' table is the **Child Table**.

Figure 67.



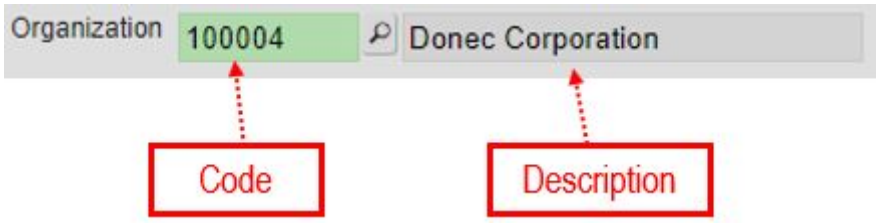
Lookup Objects provide a way for **End Users** to select an 'Organization', and thereby linking an 'Organization' to a 'Contact'.

Lookups are made up of three components:

1. The **ID** which is the foreign key on the child table
2. The **Code** Field on the parent table
3. The **Description** Field on the parent table

Note: The value of foreign keys are not displayed to **End Users**, **End Users** only need to see the **Code** and **Description** components of a **Lookup**. ⟶ *Figure 68.*

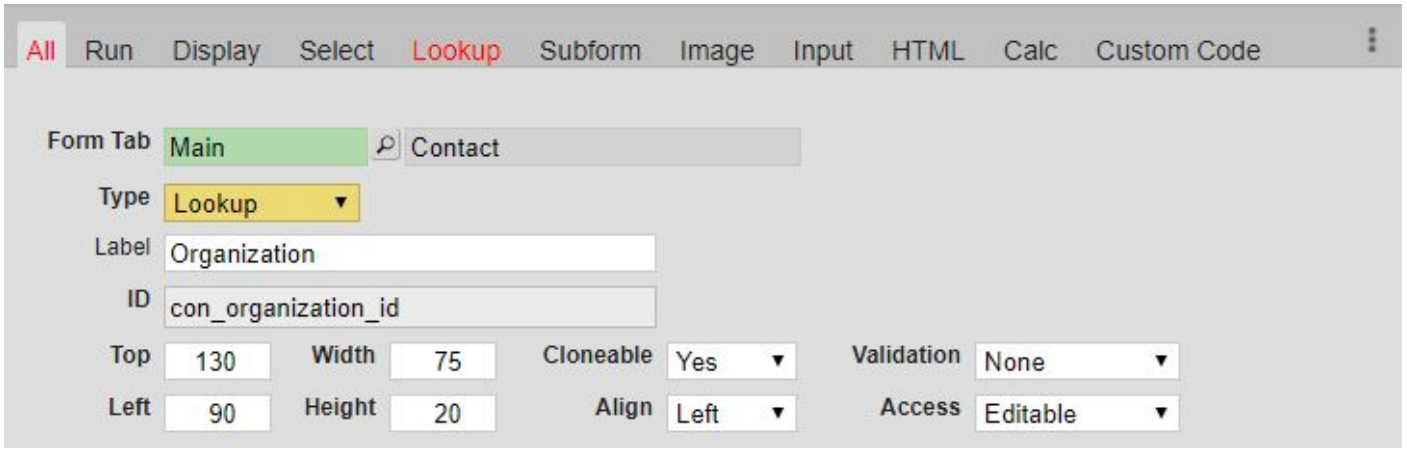
Figure 68.



Follow these steps to create a **Lookup**.

1. Log into nuBuilder with your **globeadmin** account
2. Click on the **Objects Button**
3. Click the **Add Button**
4. On the **All Tab**, fill in the following items ⟶ *Figure 69.*

Figure 69.

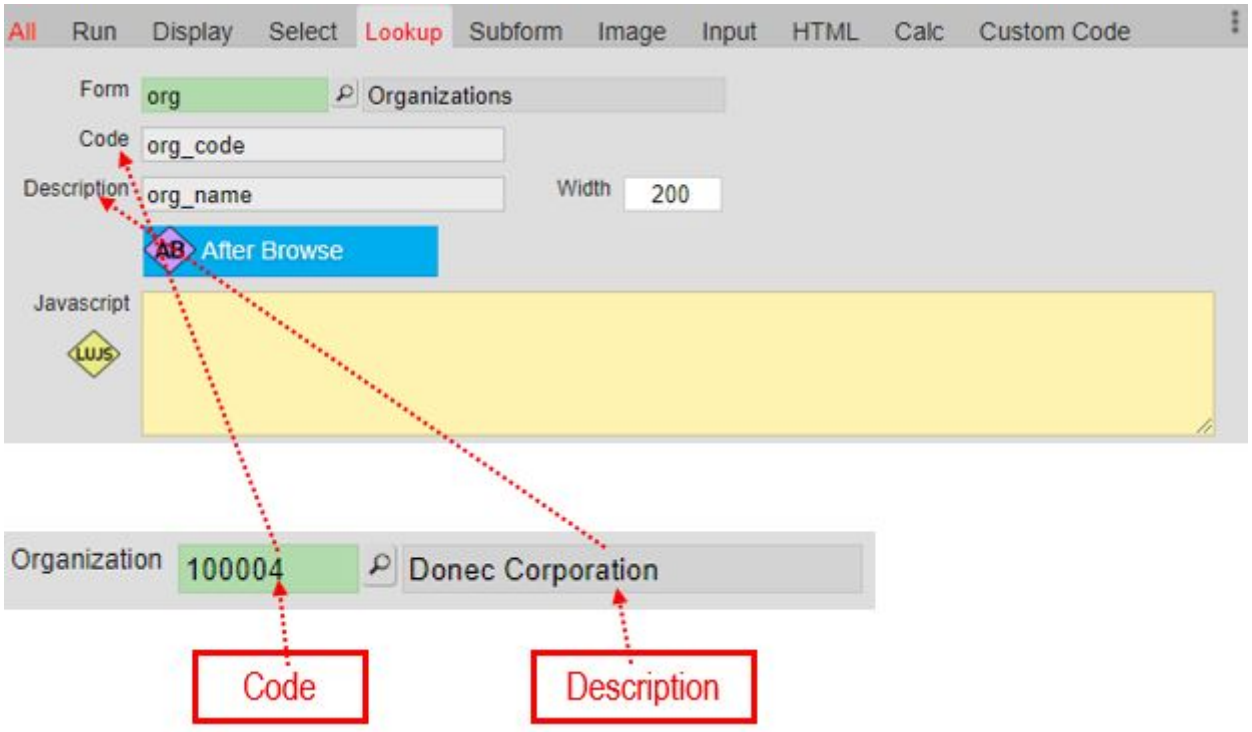


Form Tab	`Main - Contact`	All **Objects** need to be linked to a **Form Tab** and **Form**. This tells nuBuilder where to display this **Object**.
Type	`Lookup`	Select the type of **Object** to create
Label	`Organization`	In the case of a **Lookup Object**, the label (title) will be displayed next to the **Lookup Selector**
ID	`con_organization_id`	**Lookup Objects** need a foreign key for the field value
Top	`130`	Spacing of the **Tab** from the top of the **Edit Screen.** You can adjust this according to your needs for each **Object**

Left	90	Spacing of the **Tab** from the left side of the **Edit Screen.** You can adjust this according to your needs for each **Object**
Width	75	75 is a typical starting value for width, you can adjust this according to your needs for the **Object**
Height	20	20 is a typical starting value for height, you can adjust this according to your needs for each **Object**
Align	Left	The alignment of the text **Input** is left by default.
Access	Editable	Allows text **Input** to be edited on the **Edit Screen**

5. On the **Lookup Tab**, fill in the following items ⟶ *Figure 70.*

Figure 70.



Form	org	Before building **Lookup Objects** you must first have the **Form** built that the **Lookup** will use.
Code	org_code	This lets nuBuilder know what field to use when displaying the **Code** section of the **Lookup**
Description	org_name	This lets nuBuilder know what field to use when displaying the **Description** section of the **Lookup**
Width	200	Width of **Lookup Description** field
After Browse		PHP code that is executed after the **Lookup** selection is made. Useful for populating form fields with other data on running a **Lookup**. The nuLookupRecord() will return the database row for the row that was selected on the browse screen and assign to a field on the form using nuSetFormValue(). E.g. $val = nuLookupRecord()->other_db_field; nuSetFormValue('form_field', $val);
JavaScript		JavaScript code put here will run after the **Lookup** has run.

6. Click **Save**

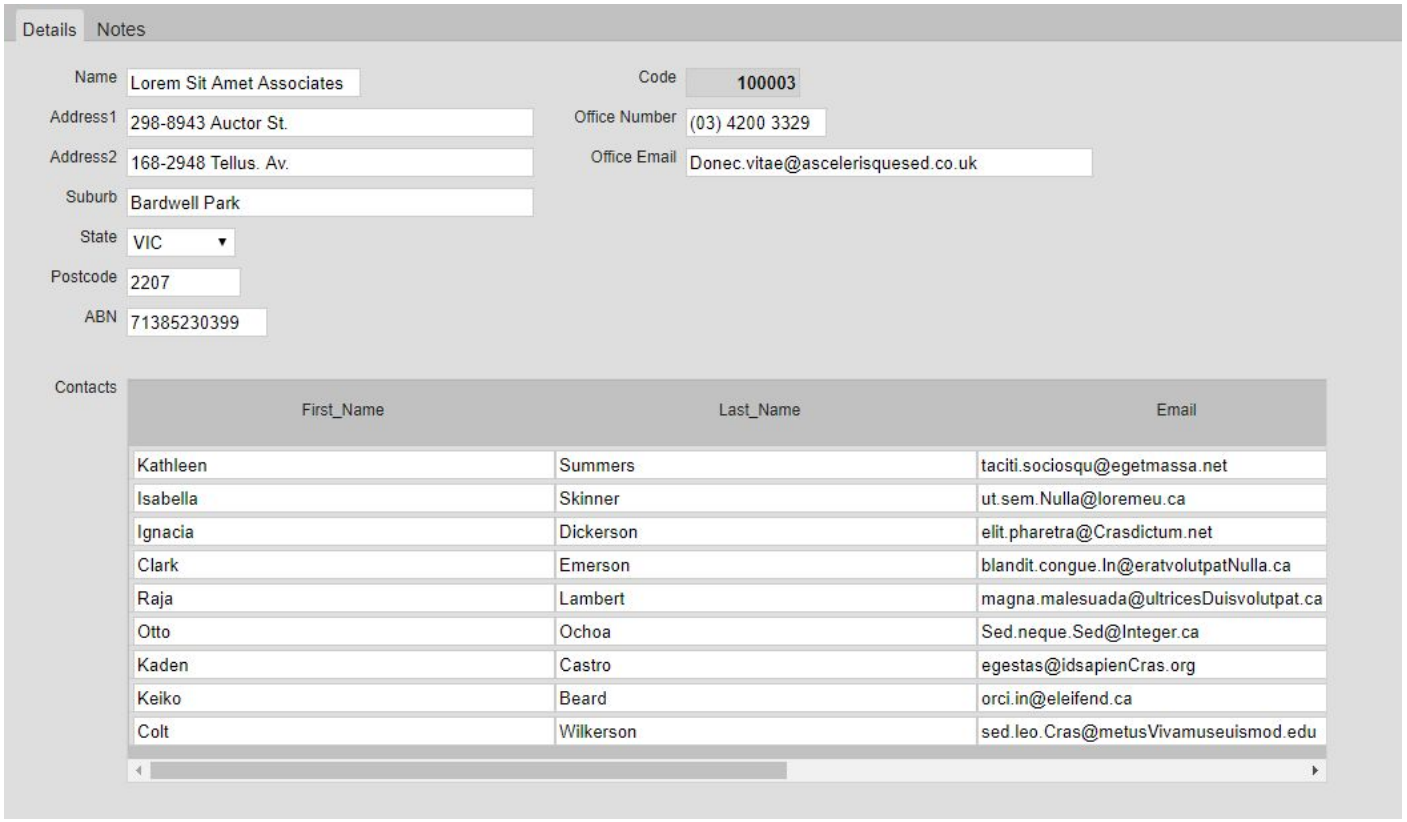
You have now completed all of the steps to create a **Lookup Object**.

Subform

In order to make this application a little more useful, we are now going to put a **Subform** on the 'Organization' **Screen** which shows all of the 'Contacts' that are linked to the 'Organization'.

When it is complete, it will then look like the following screen ⟶ *Figure 71.*

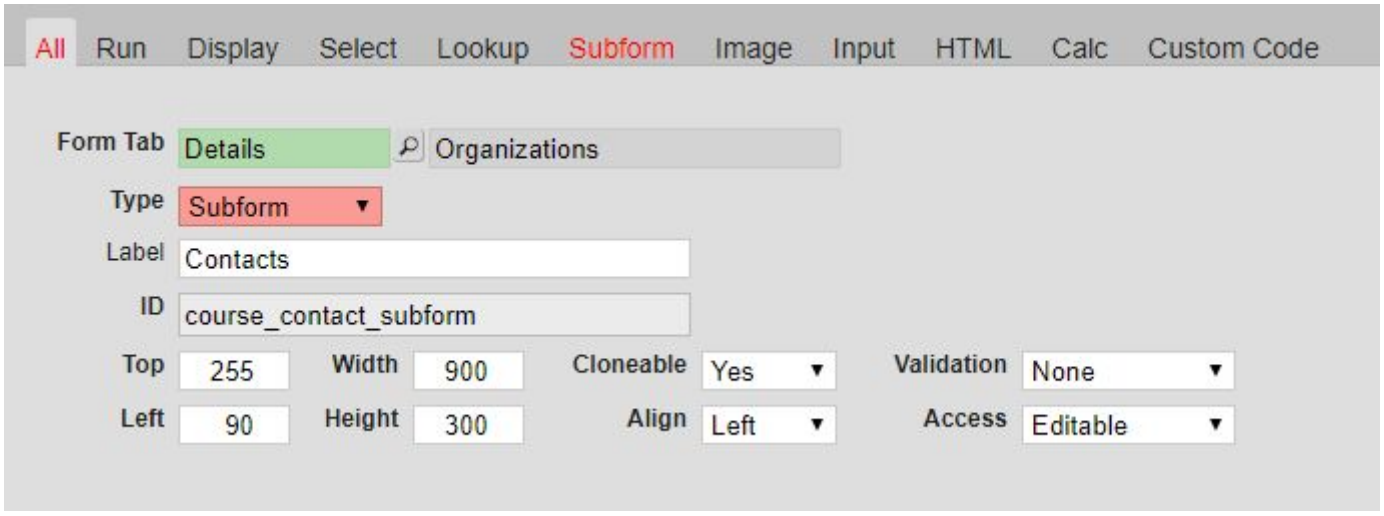
Figure 71.


Details
Notes
Name
Lorem Sit Amet Associates
Code
100003
Address1
298-8943 Auctor St.
Office Number
(03) 4200 3329
Address2
168-2948 Tellus. Av.
Office Email
Donec.vitae@ascelerisquesed.co.uk
Suburb
Bardwell Park
State
VIC
Postcode
2207
ABN
71385230399
Contacts
First_Name
Last_Name
Email
Kathleen
Summers
taciti.sociosqu@egetmassa.net
Isabella
Skinner
ut.sem.Nulla@loremeu.ca
Ignacia
Dickerson
elit.pharetra@Crasdictum.net
Clark
Emerson
blandit.congue.In@eratvolutpatNulla.ca
Raja
Lambert
magna.malesuada@ultricesDuisvolutpat.ca
Otto
Ochoa
Sed.neque.Sed@Integer.ca
Kaden
Castro
egestas@idsapienCras.org
Keiko
Beard
orci.in@eleifend.ca
Colt
Wilkerson
sed.leo.Cras@metusVivamuseuismod.edu

Subforms provide a way for 'one to many' relationship to be implemented.
In this case, we do not need to create another **Form** because the subform we will be creating is using the **Contacts Form**, so we will only be creating a **Subform Object.**

1. Log into nuBuilder with your **globeadmin** account
2. Click on the **Objects Button** in the **Setup Tab**
3. Click the **Add Button**
4. On the **All Tab**, fill in the following items ⟶ *Figure 72*

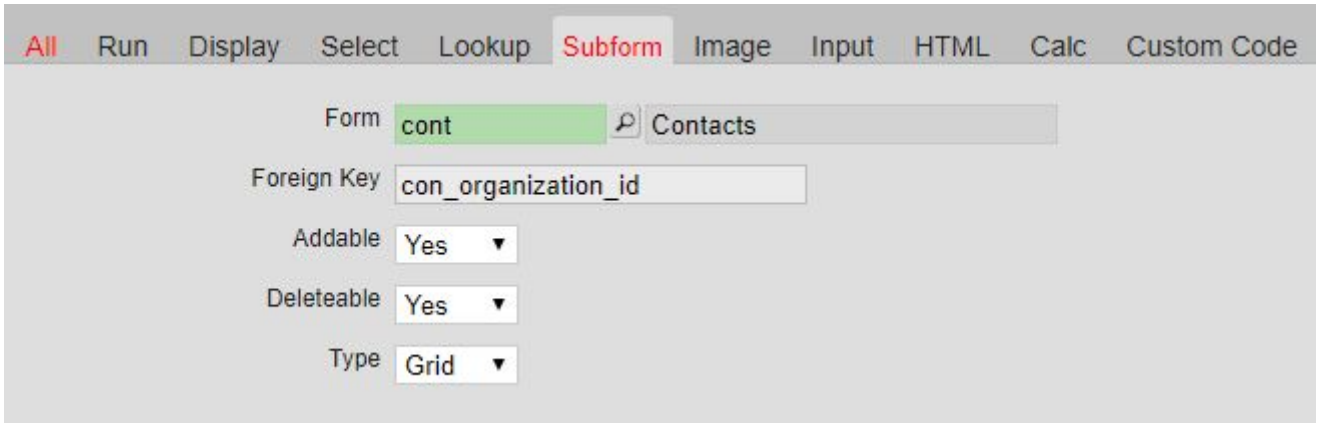
Figure 72.



Form Tab	Details - Organizations
Type	Subform
Label	Contacts
ID	course_contact_subform
Top	255
Left	90
Width	900
Height	300
Align	Left

5. On the **Subform Tab**, fill in the following items ⟶ *Figure 73*

Figure 73.



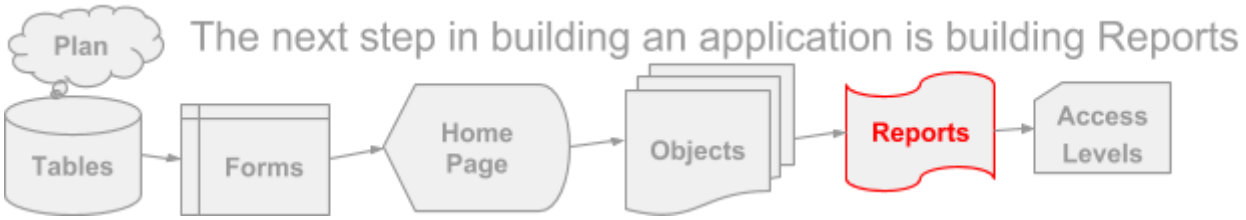
Form	cont - Contacts
Foreign Key	con_organization_id
Addable	Yes
Deleatable	Yes
Type	Grid

6. Click **Save.**
These are all of the steps needed to create a **Subform**.

Click on the green 'User Home' **Button**, then click on 'Organizations' and then click on any record to view the **Screen**.

You have now completed all of the steps to finish your **Subform Object**. Refer back to ⟶ *Figure 71,* to see how your 'Organization' **Screen** should look.

Steps to Program a Report



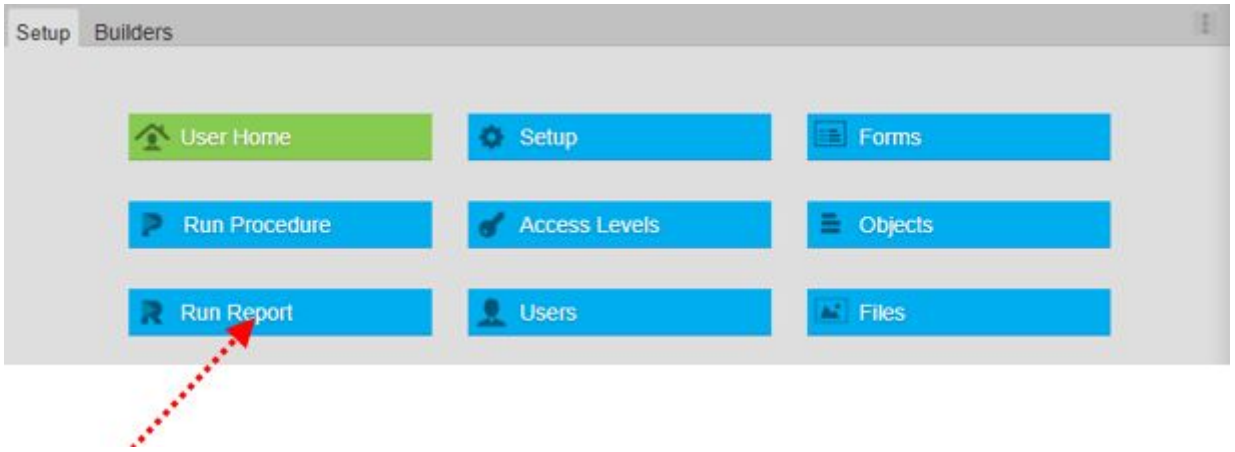
In this section we will learn about the following:

- Preparing your application to run **Reports**
- Create a **Run Report Button**
- The nuBuilder **Report** development process
- How to build a **Report Criteria Screen**
- Build **Criteria Form**
- Add **Objects** to **Criteria Form**
- How to create SQL for a **Report**
- Analysis of **Report** SQL
- Creating the **Report**
- How to design the layout of a **Report**
- **Report Designer**
- **Report Objects**
- **Report Sections**
- Using the **Report Designer**
- Getting Started with **Report Designer**
- **Details Section**
- Saving changes
- **Group Properties**
- **Page Header**
- Underlining **Page Header** Items
- **Report Header**
- **Page Foote**r
- **Report Footer**
- Running a **Report**

Preparing your application to run Reports

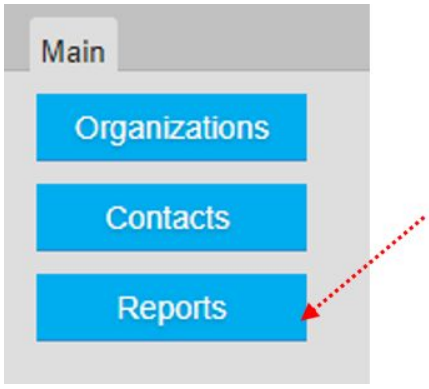
When we create **Reports** for **End Users** we normally duplicate the **Run Report Button** ⟶ *Figure 74.*

Figure 74.



A **Run Report Button** located on the default **Home Page** in the **Setup** tab provides a way for **End Users** to gain access to **Reports**. ⟶ *Figure 75.*

Figure 75.

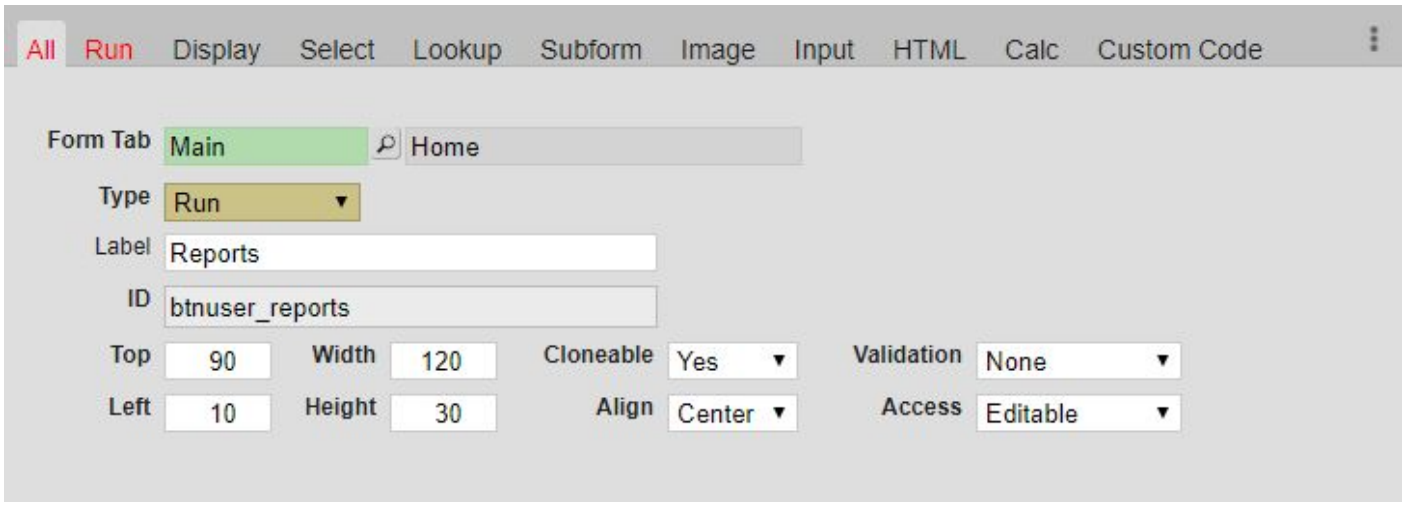


Create a **Run Report Button**

Follow these steps to create a **Run Report Button**.

1. Log into nuBuilder with your **globeadmin** account
2. Click on the **Objects Button** in the **Setup** tab
3. Click the **Add Button**
4. On the **All Tab**, fill in the following items ⟶ *Figure 76.*

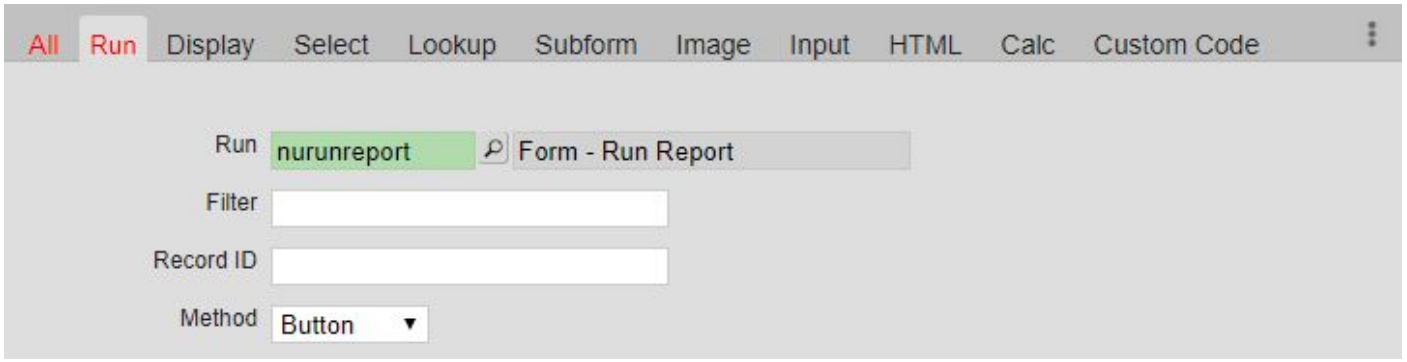
Figure 76.



Form Tab	Main - Home
Type	Run
Label	Reports
ID	btnuser_reports
Top	90
Left	10
Width	120
Height	30
Align	Center

5. On the **Run Tab**, fill in the following items ⟶ *Figure 77.*

Figure 77.



Run	nurunreport	This is a pre-build **Form** in nuBuilder used for searching and running **Reports**
Method	Button	Type of **Object** to be created to run the **Form**

6. Click **Save**

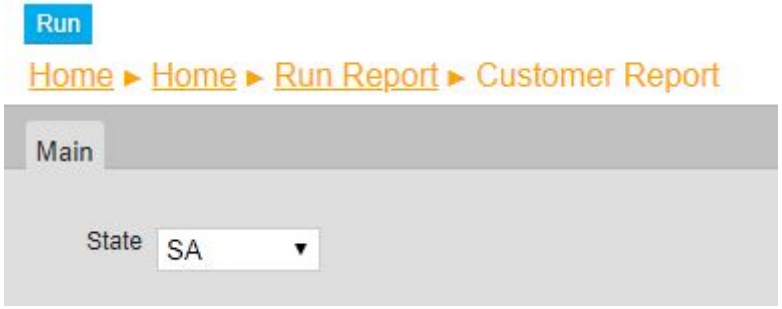
You have now completed all of the steps to create a **Reports Button**.

Reports follow the **Two Screen** pattern. When an **End User** clicks on the **Report Button** they will see a **Search Screen** ⟶ *Figure 78.* showing all **Reports** which have been granted to their **Access Level**.

Figure 78.

Following on in the **Two Screen** pattern, when an **End User** clicks on a **Report** that they have found on the **Search Screen**, a **Criteria Screen** is loaded. ⟶ *Figure 79.*

Figure 79.



	Reminder: **Criteria Screens** are just **Edit Screens** with different **Action Buttons**

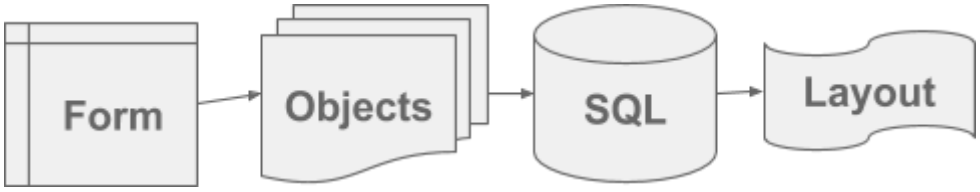
	Using a **Search Screen** for **Reports** is very useful because over time, applications tend to have more and more **Reports** built requiring the ability to search for the **Report** you need.

The nuBuilder Report development process

We are now going to go through the steps needed to build a **Report**. Our **Report** will be a simple listing of all 'Customers' filtered by 'State'.

Developing **Reports** within nuBuilder involves the following steps. ⟶ *Figure 80.*

Figure 80.

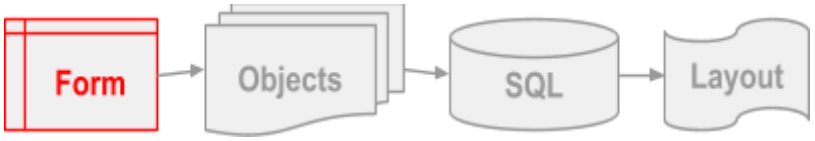


Forms and Objects
You need to create a **Form** and place any **Objects** that you need on that **Form**. This **Form** will be used as the **Criteria Screen**. In our example we need the **End Users** to select a 'State' from a **Dropdown** so that the **Report** can filter the results based on this criteria.

SQL
We then design an SQL statement to build a temporary table which will contain all of the information that we want to be displayed in our **Report**.

Layout
We then design the layout of the **Report** using a special application built into nuBuilder which we call the **Report Designer**.

How to build a Report Criteria Screen



Build **Criteria Form**

1. Follow the steps that you have learnt previously in order to create a new **Form** using the information provided. In the **Main Tab,** fill in the following information below:

Form Type	`Browse and Edit`	The form type is to give access to what the end user can do to the form
Code	`cus`	Each **Form** needs a unique code so it can be referenced in other sections of nuBuilder
Description	`Customer Report`	The **Description** is displayed on **Lookups**
Table	`zzzzsys_setup`	**Report Criteria Screens** do not use a table because reports do not save information. However the nuBuilder system still requires a table on the **Form Screen**. We normally use the '`zzzzsys_setup`' table
Primary Key	`zzzzsys_setup_id`	**Report Criteria Screens** do not use a primary key because reports do not save information, however nuBuilder still requires a primary key on the **Form Screen**. We normally use the '`zzzzsys_setup_id`' key

In the **Browse Tab**, the following information will have already been automatically filled

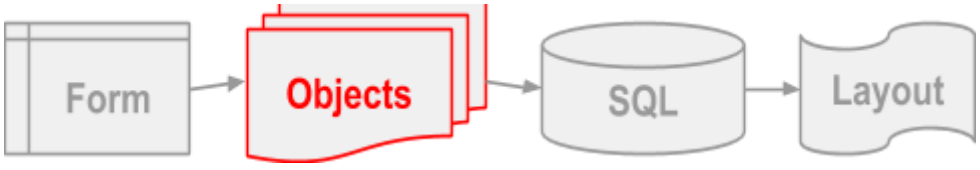
SQL	`SELECT * FROM zzzzsys_setup`	The following SQL would have automatically appeared after you tabbed out of the above table field

You do not need to fill out any other fields on the **Form Screen**.

2. Click **Save**

Add **Objects** to **Criteria Form**

Follow these steps to place **Objects** on our **Report Criteria Screen.**



1. Navigate to the **Objects Search Screen** and search for a **Select Object** that we created earlier for the 'Organization' **Screen**. ⟶ *Figure 81.*

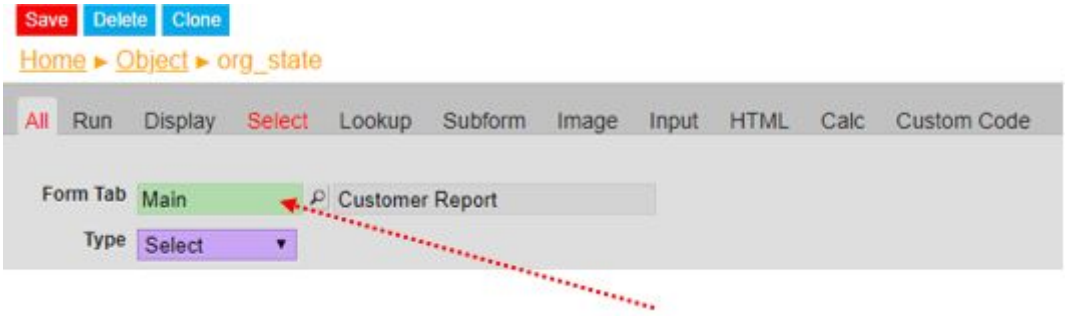
Figure 81.

2. Click on the **Select Object** to load up the **Object Edit Screen**
3. Click on the **Clone Button** ⟶ *Figure 82.*

Figure 82.

6. Change the **Form Tab** field from '`Details`' to '`Main`' ⟶ *Figure 83.*

Figure 83.



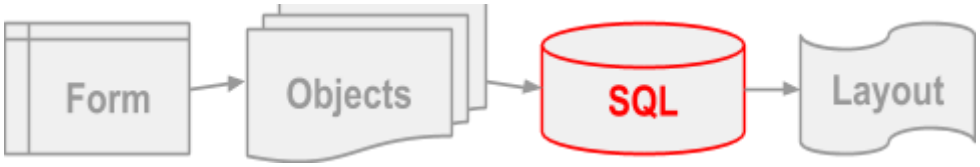
Form Tab `Details(Organizations ⟶ Main(Customer Report))`

7. Click **Save**

	Clicking on the **Clone Button** allows you to copy a record to be used elsewhere.

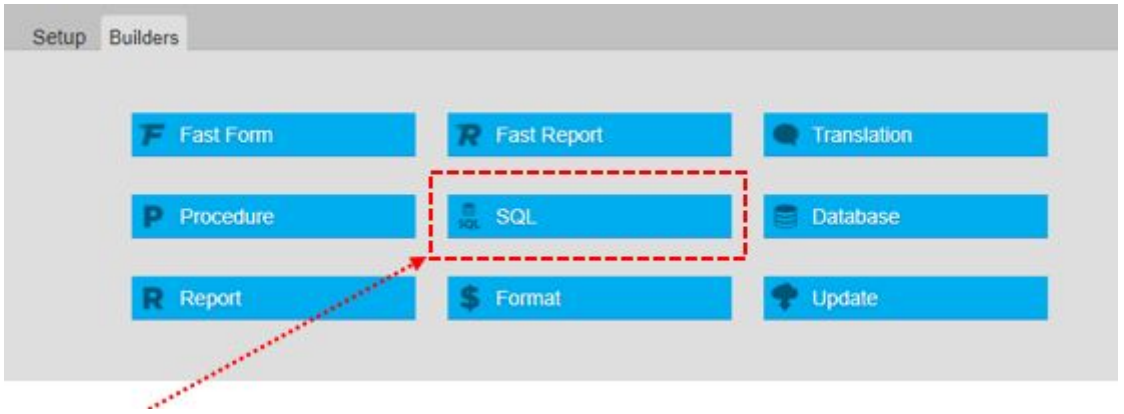
How to create SQL for a Report

Follow these steps to create an SQL statement for our **Report.**



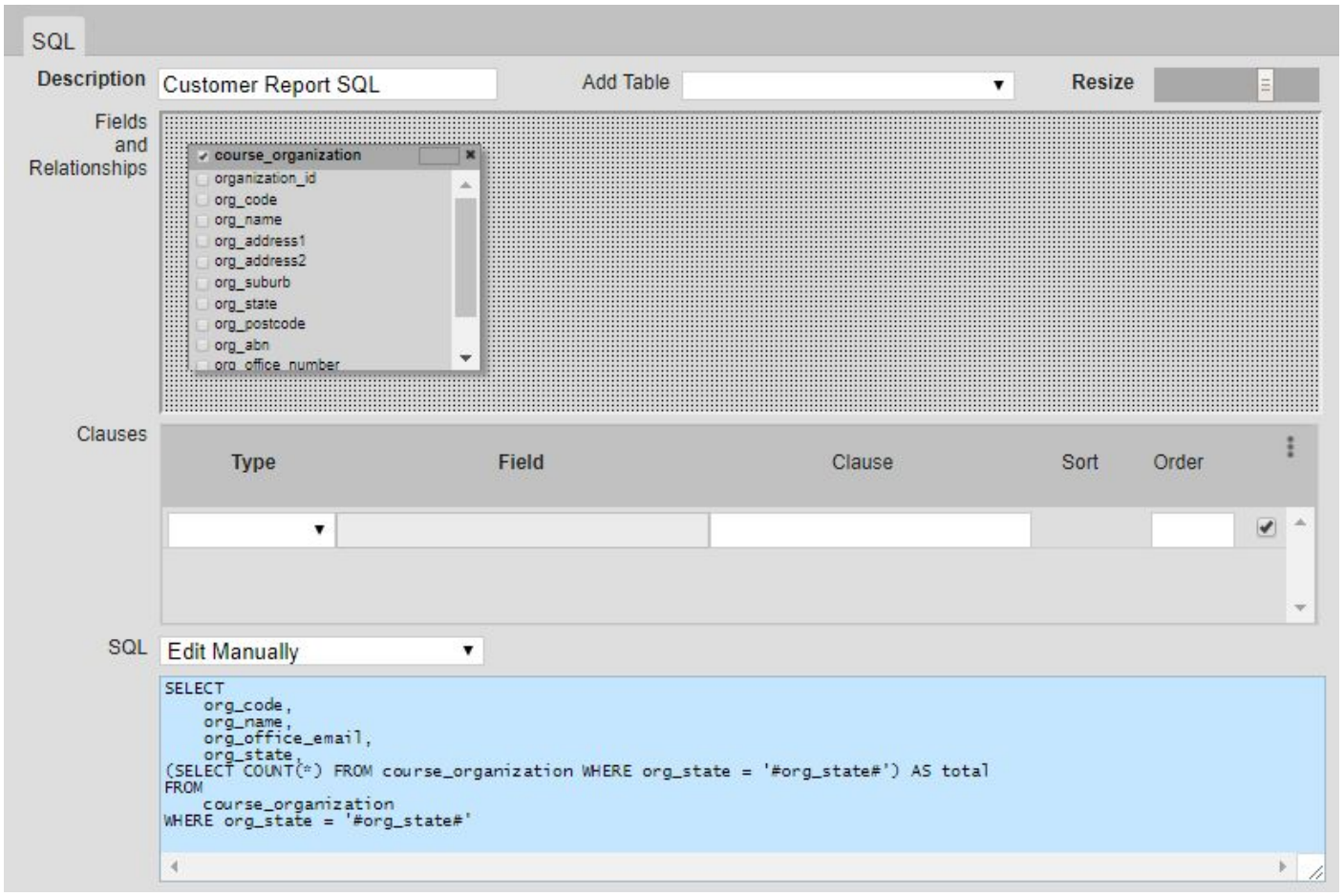
1. Click on the **SQL Button** in the **Builders Tab** ⟶ *Figure 84.*

Figure 84.



2. Click **Add Button**.
3. Click on the **Add Table Dropdown** and choose **course_organization**.
4. Fill in the following items ⟶ *Figure 85a.*

Figure 85a.



Description	Customer Report SQL	The name of this **SQL** which will be used when we build the **Report.**
SQL Dropdown	Edit Manually	Here, we can type in our own sql code in the given textarea at the bottom of this dropdown.
SQL Textarea	SELECT org_code, org_name, org_office_email, org_state, (SELECT COUNT(*) FROM course_organization WHERE org_state = '#org_state#') AS total FROM course_organization WHERE org_state = '#org_state#'	This SQL query will create a temporary table containing all records from the organization table which have a 'State' that matches the 'State' selected by the **End User**

5. Click **Save**

Values enclosed with hash symbol '#' are known as **Hash Variables**. `#org_state#` is a **Hash Variable** that is created by nuBuilder when an **End User** selects an item from the **Dropdown** on the **Criteria Form** that we created previously.

Analysis of Report SQL

The following SQL statement used in this **Report** is an example of a **NESTED SELECT** statement.

```
SELECT org_code, org_name, org_office_email, org_state,
(SELECT COUNT(*) FROM course_organization
WHERE org_state = '#org_state#')
AS total
FROM course_organization
WHERE org_state = '#org_state#'
```

To help understand this SQL statement it might be helpful to remove the items which have been highlighted as bold.

```
SELECT org_code, org_name, org_office_email, org_state,
(SELECT COUNT(*) FROM course_organization
WHERE org_state = '#org_state#')
AS total
FROM course_organization
WHERE org_state = '#org_state#'
```

We then end up with the following SQL statement which is a simple **SELECT** statement that you might be familiar with by now.

```
SELECT org_code, org_name, org_office_email, org_state
FROM course_organization
WHERE org_state = '#org_state#'
```

The remaining items in bold is simply an additional **SELECT** statement **NESTED** in brackets '`( )`' followed by the words '`AS totals`'.

The word '`AS`' followed by a value will place the results in a **Generated Column** which we have decided to call '`totals`'.

```
(SELECT COUNT(*) FROM course_organization
WHERE org_state = '#org_state#')
AS total
```

Testing your SQL

It is a good idea to test your SQL before attempting to complete your **Report.** You can test your SQL using phpMyadmin which is located on the **Button** called **Database** in the **Builders Tab**.

The following demonstrates the **Report** SQL statement without using the **NESTED SELECT** and **Generated Column** ⟶ *Figure 85b*

Figure 85b.

```
SELECT org_code, org_name, org_office_email, org_state
FROM course_organization
WHERE org_state = 'SA'
```

org_code	org_name	org_office_email	org_state
100004	Donec Corporation	volutpat.Nulla.dignissim@Quisque.co.uk	SA
100009	Fermentum Metus Institute	a@tincidunt.ca	SA
100010	Erat Etiam Vestibulum Corp.	et@dapibus.ca	SA
100024	Dui Lectus Inc.	lacinia.mattis@nuncnulla.co.uk	SA
100031	Duis Volutpat Nunc Limited	eu.dui.Cum@netusetmalesuada.co.uk	SA
100043	Posuere At Corporation	orci@rhoncus.net	SA
100049	Ridiculus Industries	non.lacinia.at@DonecegestasAliquam.edu	SA
100072	Vitae Mauris Company	amet.metus.Aliquam@velit.org	SA

The following demonstrates the **Report** SQL statement with the **NESTED SELECT** and **Generated Column** ⟶ *Figure 85c*

Figure 85c.

```
SELECT org_code, org_name, org_office_email, org_state,
(SELECT COUNT(*) FROM course_organization WHERE org_state = 'SA')
AS totals
FROM course_organization
WHERE org_state = 'SA'
```

org_code	org_name	org_office_email	org_state	totals
100004	Donec Corporation	volutpat.Nulla.dignissim@Quisque.co.uk	SA	8
100009	Fermentum Metus Institute	a@tincidunt.ca	SA	8
100010	Erat Etiam Vestibulum Corp.	et@dapibus.ca	SA	8
100024	Dui Lectus Inc.	lacinia.mattis@nuncnulla.co.uk	SA	8
100031	Duis Volutpat Nunc Limited	eu.dui.Cum@netusetmalesuada.co.uk	SA	8

Alternative methods to create SQL for Reports

There are two methods a Developer can use to create the temporary table which is needed for a Report to run.

Method #1, is to provide an SQL statement as seen in → *Figure 85a.* This is a simple approach which will only work if the entire Report can be produced from a single SQL statement.

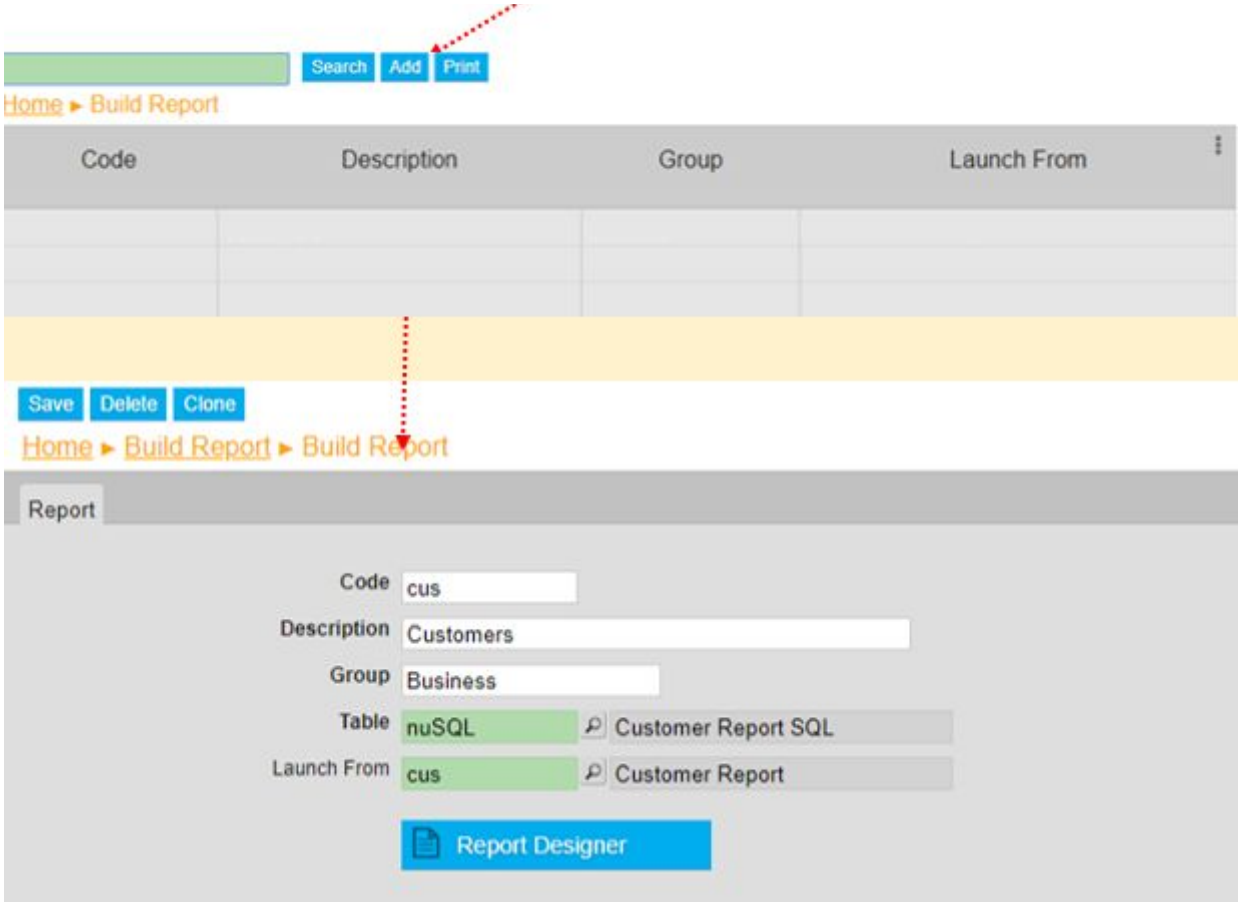
Method #2, is to create a custom section of PHP code and link that code to the Report. This is a more complex approach however it is the method which is used the most as many Reports will need a lot more programing and cannot be produced with a single SQL statement. An explanation on how to use this method is beyond the scope of this user guide.

Creating the Report

Now that we have created the **SQL Script** for the **Report,** we will now create the **Report.**

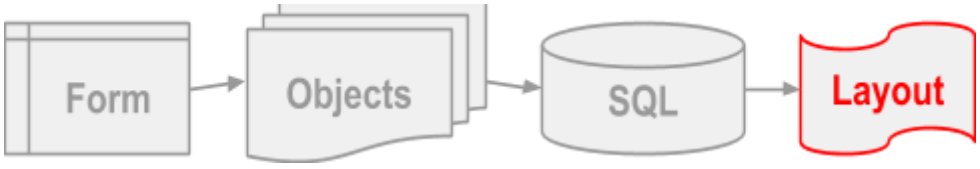
1. Log into nuBuilder with your **globeadmin** account
2. Click on the **Report Button** in the **Setup Tab**
3. Click the **Add Button** ⟶ *Figure 86.*
4. On the **All Tab**, fill in the following items ⟶ *Figure 86*

Figure 86



Code	cus	Each **Report** needs a unique code so it can be referenced in other sections of nuBuilder..
Description	Customers	The **Description** value becomes a **Hash Variable** which can be used in the **Report Designer -** described earlier (refer to page 66)
Table	nuSQL - Customer Report SQL	This connects the **Report** to the **SQL** (Customer Report SQL) which we created earlier.
Launch From	cus - Customer Report	Launches the **Report** from the **Form** we created earlier.

How to design the layout of a Report



Report Designer

The nuBuilder **Report Designer** is a special application within in nuBuilder which allows you to 'Drag and Drop' **Fields** and **Labels** onto different page **Sections**.

To learn how to use the **Report Designer** you need to become familiar with two concepts.

1) Report Objects
2) Report Sections

Report Objects

There are three types of **Report Objects**.

Field	Used to display information which is in the temporary table connected to this **Report**
Label	Used to display static information typed into the **Report Designer**, also used to display **Hash Variables**
Image	Used to link images from the nuBuilder image system

Report Sections

There are five main sections in a **Report Layout** where **Report Objects** can be placed.

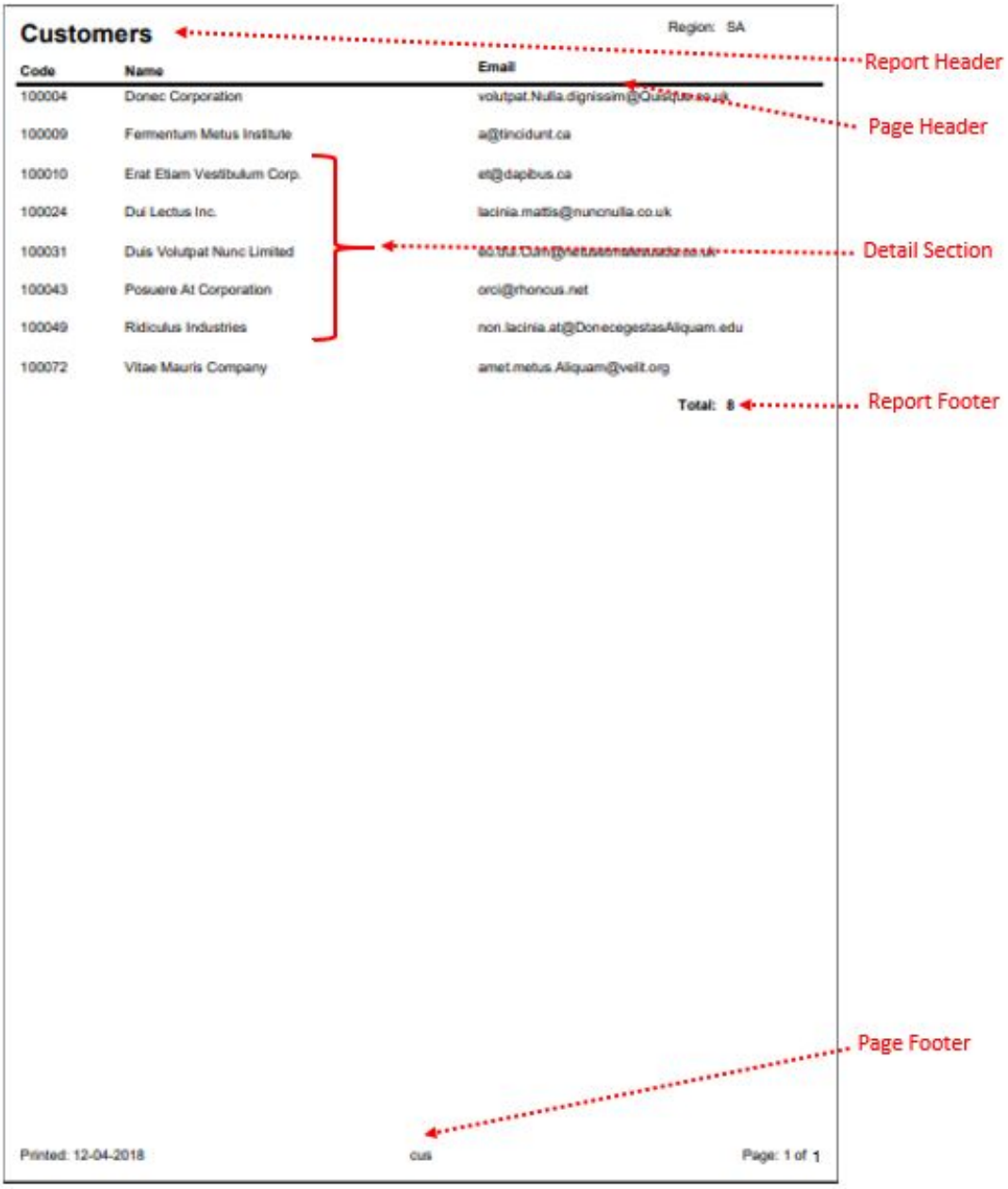
Report Header	**Report Objects** placed in the **Report Header** will only be displayed at the top of the first page of the **Report**
Page Header	**Report Objects** placed in the **Page Header** will be displayed after the **Report Header** and at the top of every other page in the **Report**
Detail	**Report Objects** placed in the **Details** section will be displayed on every page in the **Report**.
Report Footer	**Report Objects** placed in the **Report Footer** will only be displayed on the last page of the **Report** directly after the **Detail** section
Page Footer	**Report Objects** placed in the **Page Footer** will be displayed and at the bottom of every page in the **Report**.

	Don't confuse **Objects** on the **Report Designer** system with **Objects** that we use on **Forms**

When an **End User** clicks on the **Run Button** on a **Report Criteria Screen**; nuBuilder will produce a PDF document. ⟶ *Figure 87* Is an example of the **Report** that we are going to build.

Notice the different sections of the **Report Layout** and compare this to ⟶ *Figure 88;* which is the **Report Designer** interface used to make this **Report Layout**.

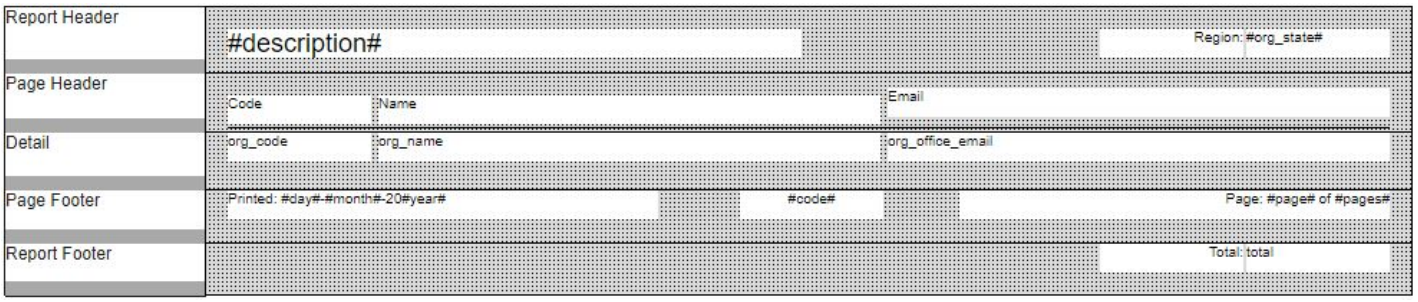
Figure 87.



	Notice that the **Report Footer** is located directly after the **Details Section** and not at the bottom of the page like **Page Footer**

The **Report Designer** application allows you to place different **Report Objects** into each section of a **Report** ⟶ *Figure 88* is an example of the **Report Designer** which has all of the **Report Objects** completed to produce the above PDF.

Figure 88.



Using the Report Designer

Follow these steps to create the **Layout** of our **Report.**

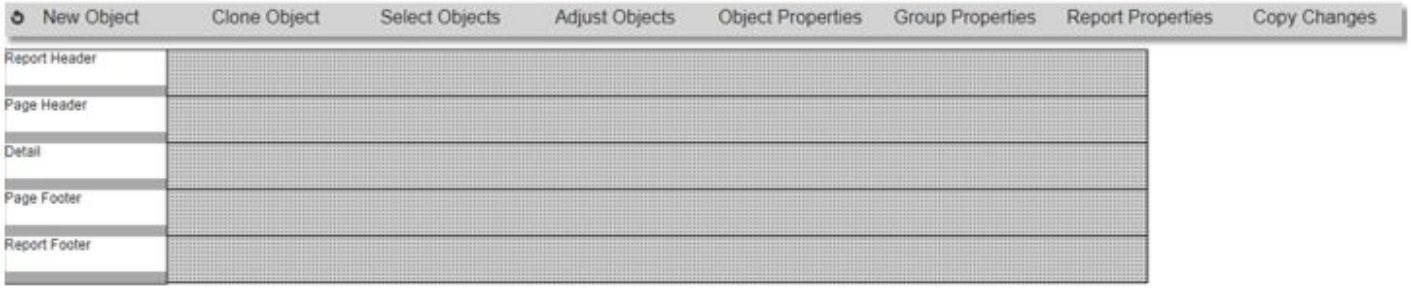
Getting Started with the Report Designer

Remain on the **Report Screen** and click on the **Button** at the bottom of the **Screen** called **Report Designer** ⟶ *Figure 89.*

Figure 89.

You will see the following **Screen** ⟶ *Figure 90.* This is how a blank **Report Builder** will appear, when it is complete it should appear like the previous example ⟶ *Figure 88.*

Figure 90.

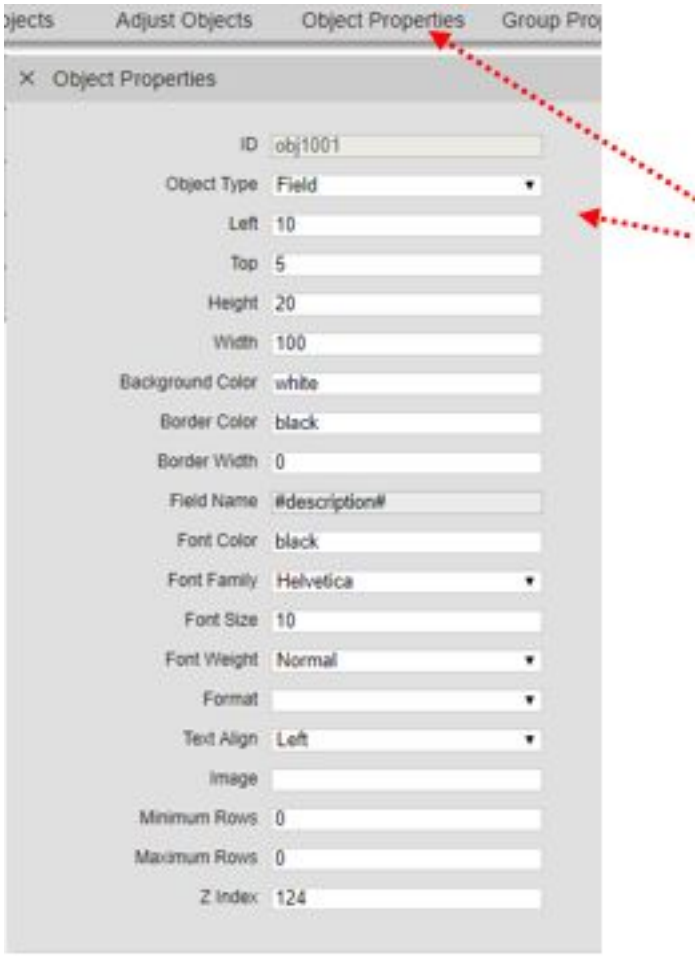


Click on the menu item 'New Object', a new object will appear on the **Screen** ⟶ *Figure 91.*
New **Report Objects** will always appear in the **Details Section**, from there you can change the **Report Objects** properties to place it in a different section.

Figure 91.

When you have a **Report Object** highlighted in red, you can click on the 'Object Properties' menu item to change the properties of a **Report Object**. → *Figure 92.* Shows the **Object Properties** dialogue.

Figure 92.



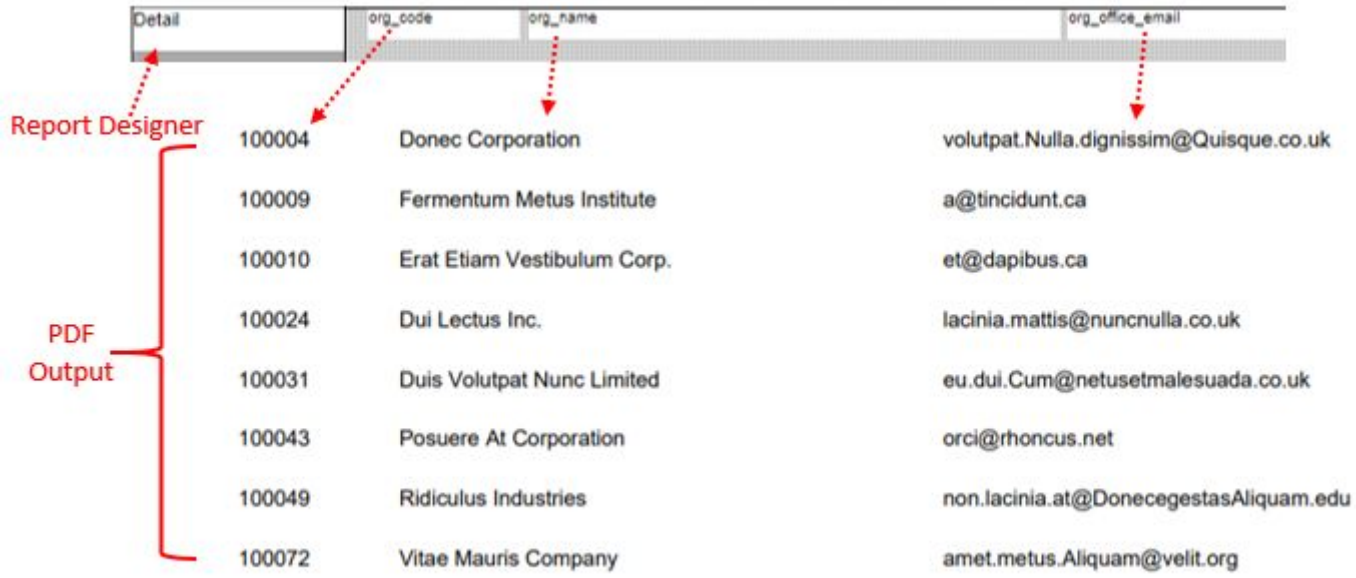
Details Section

Create the following three **Report Objects** within the **Details Section** using the following settings. You can leave all of the other settings on the **Object Properties** dialogue at their default values.

Object Type	Left	Top	Height	Width	Title
Field	15	0	20	100	org_code
Field	120	0	20	350	org_name
Field	475	0	20	350	org_office_email

Figure 93 shows the correlation between the **Report Designer Objects** and the actual PDF output of the **Report.**

Figure 93.



When you are designing **Reports** it is handy to have another screen open on your desktop which shows all of the **Fields** available to use in your **Report**

Saving Changes

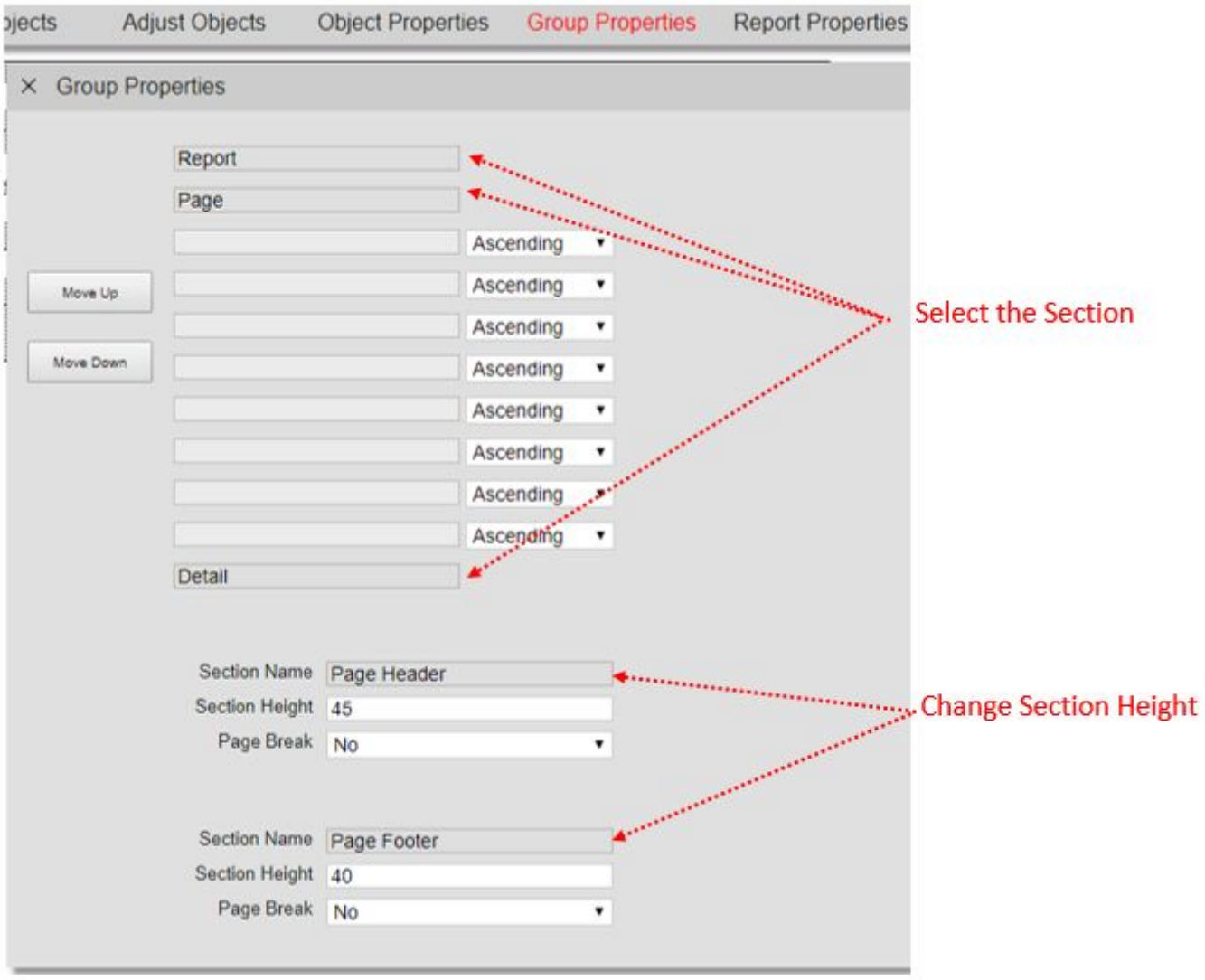
It is a good idea to regularly save your changes when using the **Report Designer.** To do this you first click on the 'Copy Changes' menu item and then click on the 'Save' **Button**. ⟶ *Figure 94.*

Figure 94.

Group Properties

We are now going to set the height for all of the **Sections** in our **Report**. Click on the menu item 'Group Properties', the **Group Properties** dialogue will appear. ⟶ *Figure 95.* As you click on the name of each **Section**, you can then set the height for that **Section**.

Figure 95.



Update the following five **Report Sections** using the following settings. You can leave all of the other settings on the **Group Properties** dialogue at their default values.

Report Header	50
Report Footer	40
Page Header	45
Page Footer	40
Detail Header	40

Then save your changes as described in ⟶ *Figure 94.*

Page Header

Create the following three **Report Objects** within the **Page Header Section** using the following settings. You can leave all of the other settings on the **Object Properties** dialogue at their default values.

Object Type	Left	Top	Height	Width	Title	Font Weight
Label	15	15	20	100	Code	Bold
Label	120	15	20	350	Name	Bold
Label	475	15	20	350	Email	Bold

Then save your changes as described in ⟶ *Figure 94.*

Underlining Page Header Items

Create the following additional **Report Object** within the **Page Header Section** using the following settings. You can leave all of the other settings on the **Object Properties** dialogue at their default values.

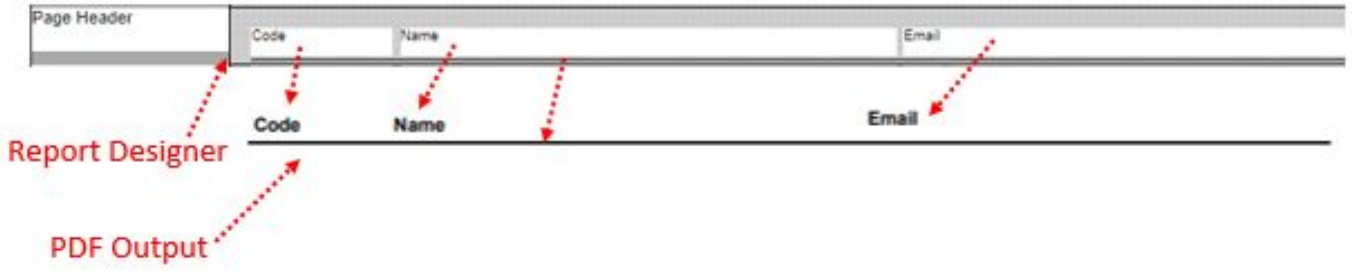
Object Type	Left	Top	Height	Width	Background Color
Label	15	37	1	810	Black

Then save your changes as described in ⟶ *Figure 94.*

Creating a **Report Object** with a **Height** of '1' and a **Background Color** of 'Black' is a good way to make a horizontal line in your **Report.** Leave the **Title** of this **Report Object** blank.

Figure 96 shows the correlation between the **Report Designer Objects** and the actual PDF output of the **Report.**

Figure 96.



Report Header

Create the following three **Report Objects** within the **Report Header Section** using the following settings. You can leave all of the other settings on the **Object Properties** dialogue at their default values.

Object Type	Left	Top	Width	Field Name	Font Size	Font Weight	Text Align
Label	15	15	400	#description#	18	Bold	Left
Label	623	15	100	Region:	10	Normal	Right
Label	725	15	100	#org_state#	10	Normal	Left

Then save your changes as described in ⟶ *Figure 94.*

Figure 97 shows the correlation between the **Report Designer Objects** and the actual PDF output of the **Report.**

Figure 97.

Page Footer

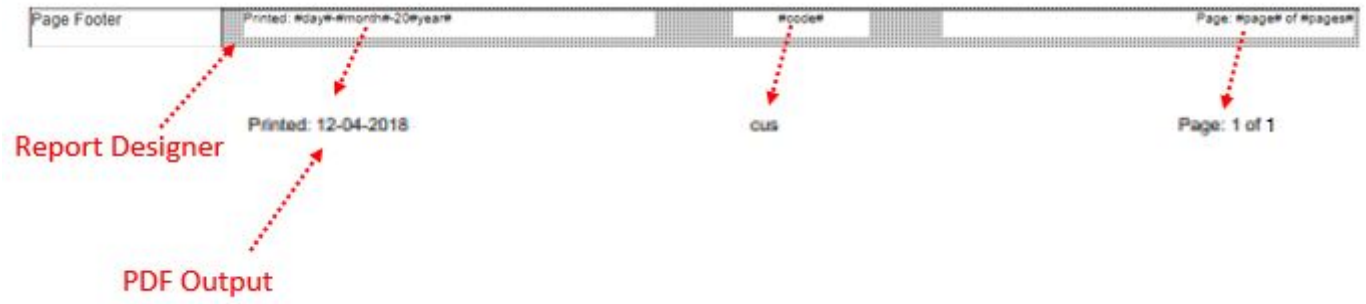
Create the following three **Report Objects** within the **Page Footer Section** using the following settings. You can leave all of the other settings on the **Object Properties** dialogue at their default values.

Object Type	Left	Top	Width	Field Name	Text Align
Label	15	0	300	Printed: #day#-#month#-20#year#	Left
Label	372	0	100	#code#	Centre
Label	525	0	300	Page: #page# of #pages#	Right

Then save your changes as described in ⟶ *Figure 94.*

Figure 98 shows the correlation between the **Report Designer Objects** and the actual PDF output of the **Report.**

Figure 98.



Report Footer

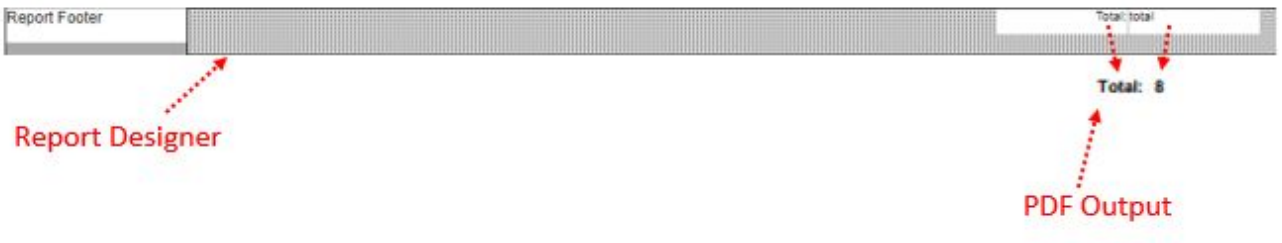
Create the following two **Report Objects** within the **Report Footer Section** using the following settings. You can leave all of the other settings on the **Object Properties** dialogue at their default values.

Object Type	Left	Top	Width	Field Name	Font Weight	Text Align
Label	623	0	100	Total:	Bold	Right
Field	725	0	100	total	Bold	Left

Then save your changes as described in ⟶ *Figure 94.*

Figure 99 shows the correlation between the **Report Designer Objects** and the actual PDF output of the **Report.**

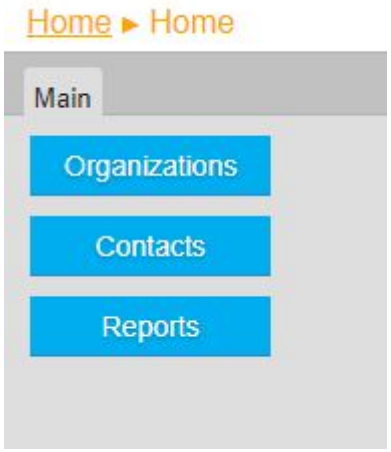
Figure 99.



Running a Report

You have now completed all of the steps to create a **Report**. To see this in action, click on the green 'User Home' **Button**. You should see a **Screen** as below ⟶ *Figure 100.*

Figure 100.

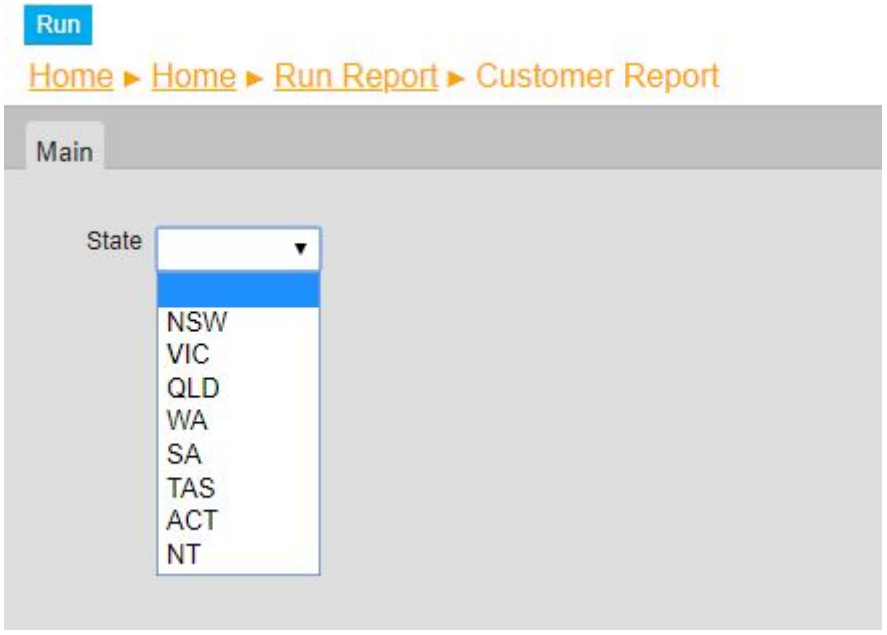


When you click on the **Button** labeled 'Reports' you will see the **Search Page** showing all of the **Reports** within nuBuilder. ⟶ *Figure 101.*

Figure 101.

Now click on the Customers **Report** which we just created, you should see the following **Criteria Screen**. ⟶ *Figure 102.*

Figure 102.



Select 'SA' for the State and then click on the **Run Button,** you should see the following **Report**. ⟶ *Figure 103.*

Figure 103.

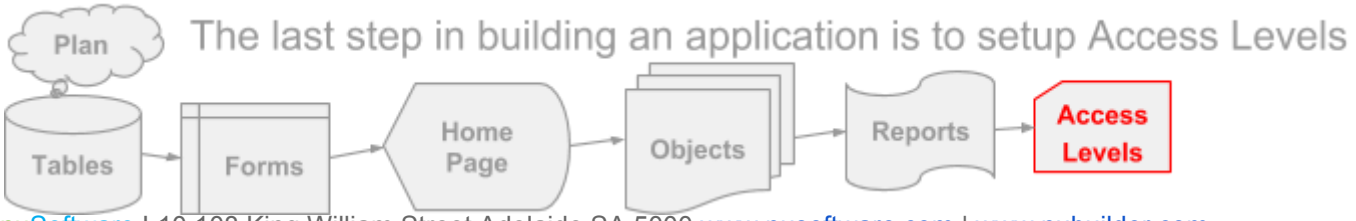
Customers

Region: SA

Code	Name	Email
100004	Donec Corporation	volutpat.Nulla.dignissim@Quisque.co.uk
100009	Fermentum Metus Institute	a@tincidunt.ca
100010	Erat Etiam Vestibulum Corp.	et@dapibus.ca
100024	Dui Lectus Inc.	lacinia.mattis@nuncnulla.co.uk
100031	Duis Volutpat Nunc Limited	eu.dui.Cum@netusetmalesuada.co.uk
100043	Posuere At Corporation	orci@rhoncus.net
100049	Ridiculus Industries	non.lacinia.at@DonecegestasAliquam.edu
100072	Vitae Mauris Company	amet.metus.Aliquam@velit.org

Total: 8

Steps to setup an Access Level



In this section we will learn about the following:

- nuBuilder permissions
- Creating **Access Levels**
- Setting the **Home Page**
- **Form** permissions
- **Report** permissions
- Creating **Users**
- Testing **Access Levels**

nuBuilder permissions

In order to create permissions for **End Users** to log into nuBuilder you need to setup the following two components:

- **Access Levels**
- **Users**

Access Levels

Each **Access Level** defines the grouping of permissions for the following: **Home Page**, **Forms**, **Reports** and **Procedures**.

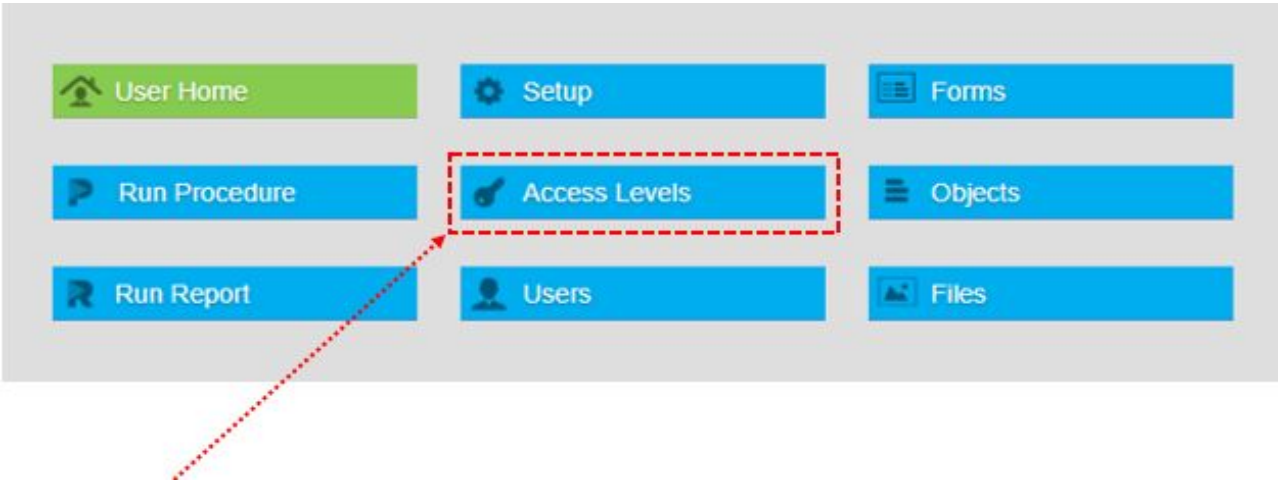
Users

The **User Screen** is where **End Users** are created.

Creating Access Levels

1. Log into nuBuilder with your **globeadmin** account
2. Click on the **Access Levels Button** ⟶ *Figure 104.*
3. Click **Add Button**

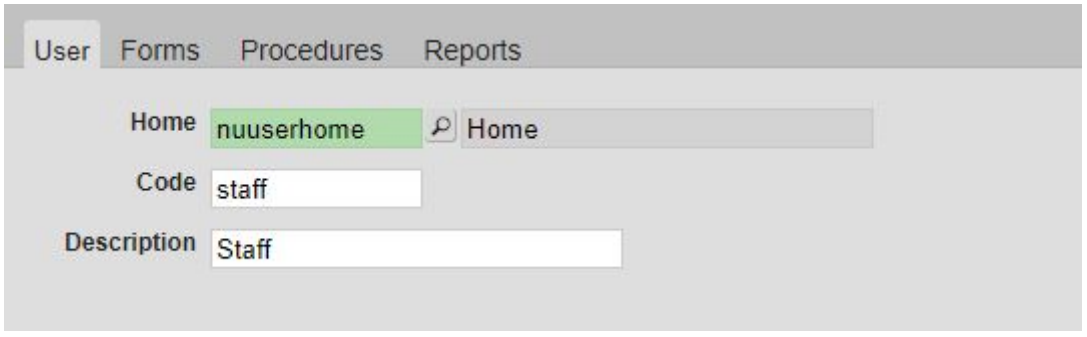
Figure 104.



Setting the Home Page

4. On the **User Tab**, fill in the following items ⟶ *Figure 105.*

Figure 105.



Code	staff	Each **Access Level** needs a unique code so it can be referenced in other sections of nuBuilder
Description	Staff	
Home Page	nuuserhome	Select the default **Home Page**

When you have applications that have several different **Access Levels**, it is helpful to create separate **Forms** for each **Access Level** instead of using the default **Home Page**

Form permissions

Each **Form** you have created for your **End Users** will need to be selected on **Forms Tab** on the **Access Level Screen**. This includes the **Report Criteria Screen** for each **Report**. You also need to remember to include the 'nurunreport - Run Report' **Form** which is pre-built in nuBuilder.

A good way to track what **Forms** you need to include is to start with the **Buttons** that you have added to a **Home Page**. if you examine each of the **Button Objects**, you can find the 'Code' and 'Description' of each **Form**. ⟶ *Figure 106.*

Figure 106.

5. On the **Forms Tab**, fill in the following items ⟶ *Figure 107.*

Figure 107.

cont	This is the **Form** we created earlier for editing 'Contacts'
cus	This is the **Report Criteria Form** we created earlier
org	This is the **Form** we created earlier for editing 'Organizations'
nurunreport	This is the built in **Form** used to search for **Reports**

Report permissions

6. On the **Reports Tab**, fill in the following item ⟶ *Figure 108.*

Figure 108.

cus	This is the **Report** we created earlier

	There will always be two components used to provide permissions to a **Report**, this includes the **Report** and the **Criteria Screen**

7. Click **Save**

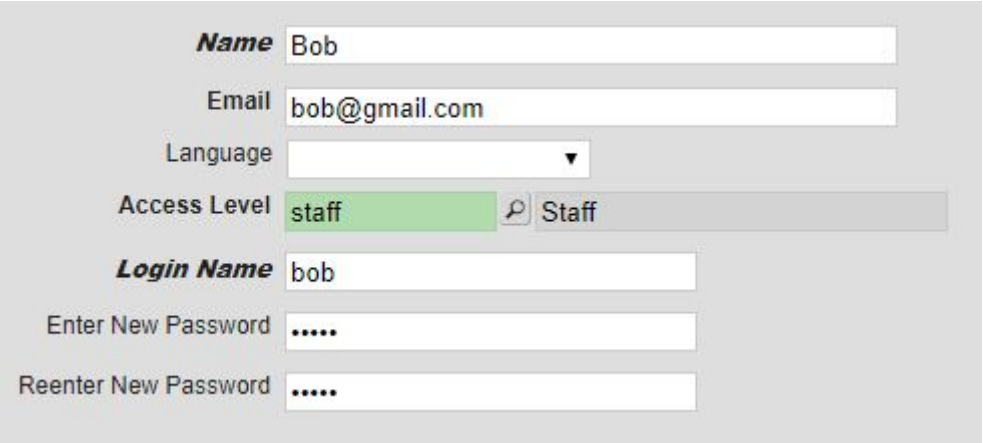
Creating Users

1. Log into nuBuilder with your **globeadmin** account
2. Click on the **Users Button** in the Setup Tab ⟶ *Figure 109.*
3. Click **Add Button**

Figure 109.

4. Fill in the following items ⟶ *Figure 110.*

Figure 110.



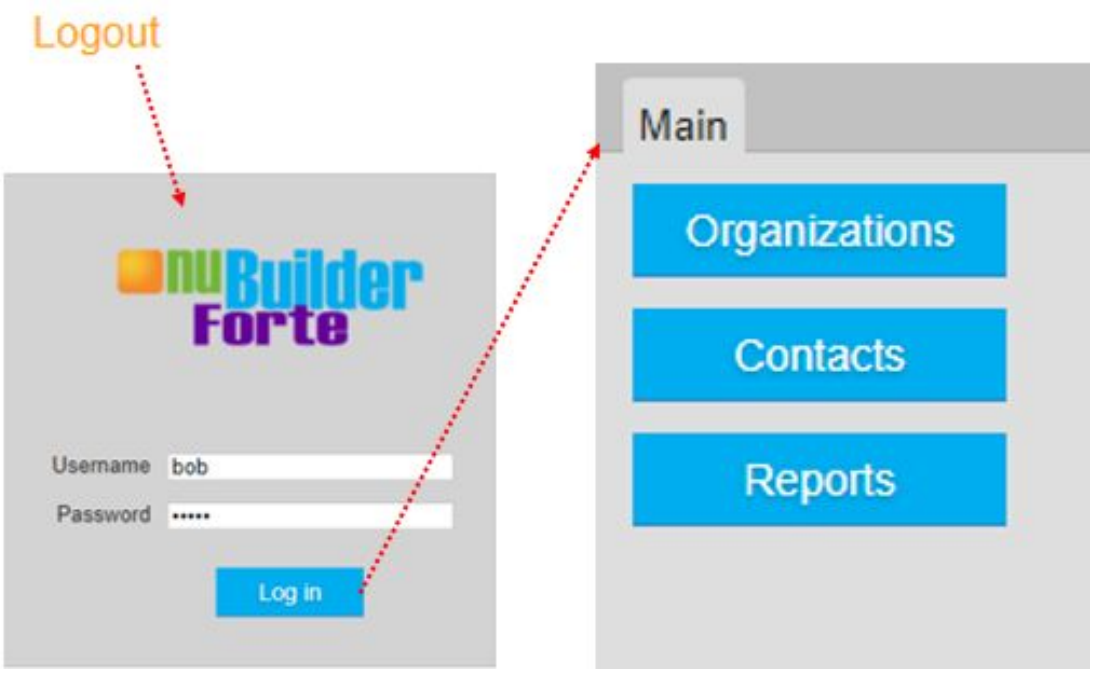
Name	Bob	
Email	bob@gmail.com	
Language		The default language is English if it's left blank
Access Level	staff - Staff	
Login Name	bob	
New Password		
Retype New Password		

5. Click **Save**

Testing Access Levels

It is a good idea to test your **Access Level**. To do this, simply log out of nuBuilder and then log back in with the **Username** and password you created in the previous step. ⟶ *Figure 111.*

Figure 111.


Logout
nuBuilder
Forte
Username
bob
Password
Log in
Main
Organizations
Contacts
Reports