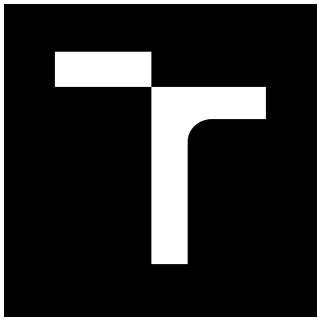
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2018 Bc. Michaela Homzová



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY
DEPARTMENT OF CONTROL AND INSTRUMENTATION

INFORMAČNÍ SYSTÉM PRO PODPORU VÝROBY
DATA STORAGE ARCHITECTURE FOR MANUFACTURING PROCESSES

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Michaela Homzová**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Radovan Holek, CSc.**
SUPERVISOR

BRNO 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Kybernetika, automatizace a měření**
Ústav automatizace a měřicí techniky

Studentka: Bc. Michaela Homzová ***ID:*** 154735
Ročník: 2 ***Akademický rok:*** 2017/18

NÁZEV TÉMATU:

Informační systém pro podporu výroby

POKYNY PRO VYPRACOVÁNÍ:

1. Seznamte se s problematikou měření a ukládání dat ve výrobních linkách osazených měřicími a testovacími stanicemi, popište současný stav.
2. Zdokumentujte strukturu výstupních dat z výrobních linek osazených měřicími a testovacími stanicemi.
3. Navrhněte a realizujte datový a procesní model pro podporu vybraných procesů v architektůře Klient Server, pro zvolený SQL databázový stroj.
4. Navrhněte a realizujte algoritmy pro sběr a vyhodnocení dat z měřicích a testovacích stanic včetně uživatelského rozhraní.
5. Otestujte a vyhodnoťte funkčnost navrženého informačního systému na souboru vzorových dat.

DOPORUČENÁ LITERATURA:

RÁČEK, J. Strukturovaná analýza systémů. 1. vyd. Brno: Masarykova univerzita, 2006, 103 s. ISBN 80-210-4190-0.
ŠIMŮNEK, M. SQL Kompletní kapesní průvodce . Grada ISBN 80-7169-692-7

Termín zadání: 5. 2. 2018 ***Termín odevzdání:*** 14.5.2018

Vedoucí práce: Ing. Radovan Holek, CSc.
Konzultant:

doc. Ing. Václav Jirsík, CSc.
předseda oborové rady

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta elektrotechniky a komunikačních technologií
Ústav automatizace a měřicí techniky

UPOZORNĚNÍ:

Autor diplomové práce nesmí při vytváření diplomové práce porušit autorská práva třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

ABSTRAKT

Tato práce se zabývá návrhem databázového systému pro skladování výsledků měření a testování vyrobených jednotek ze stanic, které tvoří součást výrobní linky.

KLÍČOVÁ SLOVA

Sběr dat, SQL, databáze, datový model, LabVIEW, TestStand

ABSTRACT

This work is covering desing of database system which stores results of measerements performed on test stations which are part of assembly line.

KEYWORDS

data acquisition, data storage, SQL, data model, database, LabVIEW, TestStand

HOMZOVÁ, Michaela. *Informační systém pro podporu výroby*. Brno, Rok, 98 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky. Vedoucí práce: Ing. Radovan Holek, CSc.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Informační systém pro podporu výroby“ jsem vypracoval(a) samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor(ka) uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil(a) autorská práva třetích osob, zejména jsem nezasáhl(a) nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom(a) následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora(-ky)

PODĚKOVÁNÍ

Rád bych poděkovala vedoucímu diplomové práce panu Ing. Radovanu Holkovi, CSc. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

podpis autora(-ky)

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Tento diplomová práce byla zadána a realizována ve spolupráci s firmou Kentigen. Tato firma se zabývá výrobou měřících a testovacích stanic, převážně pro automobilový průmysl. Na uvedených stanicích v rámci výroby probíhají funkční testy zařízení, kdy se testuje konečná funkcionalita zařízení před tím, než je posláno k montáži zákazníkovi. Po provedení těchto finálních testů se nepočítá s jakoukoliv další manipulací ani úpravou otestovaného zařízení. Jedná se tedy o poslední black-box test před finální montáží testovaného zařízení na místo určení, kterým je v případě automobilový průmyslu vozidlo.

V rámci kontroly kvality je v dnešní době stále častěji vznášen požadavek na uchovávání dat o životním cyklu výrobku nejen po jeho expedování, ale po celou jeho životnost, která se u některých dílů odhaduje i na dvacet let. Cílem této práce je nahradit stávající způsob skladování dat na disku testovacího stroje databází, která běží na vzdáleném serveru, který není součást výrobní linky, případně ani výrobní sítě. Proto se na něj nevztahují bezpečnostní omezení kladená na všechna zařízení v produkční síti.

1 SOFTWAROVÉ ŘEŠENÍ TESTOVACÍCH STANIC

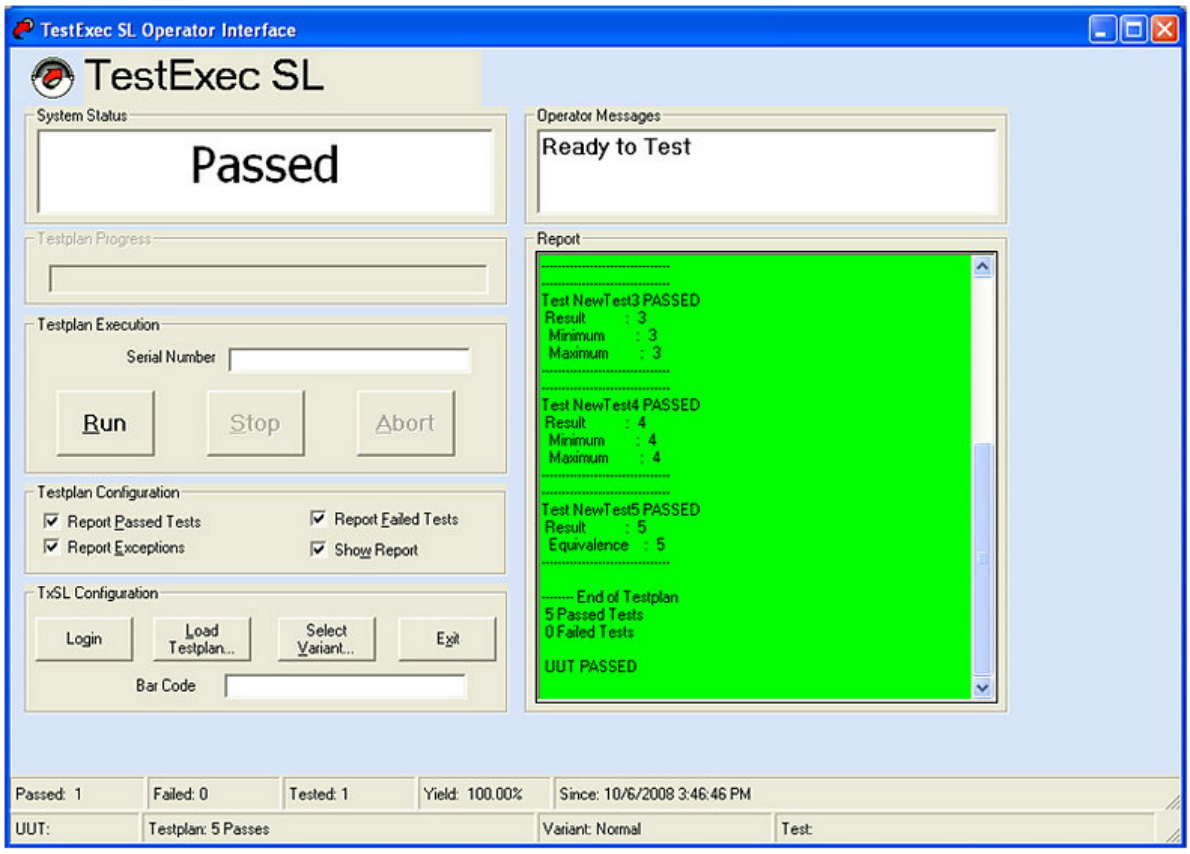
V současné době je snaha omezit podíl lidské práce u testovacích stanic na nezbytné minimum. Jedním ze základních požadavků zákazníka je plně automatizovaný testovací cyklus a jednou z možností je měření a testování na PLC. Toto je ovšem delší a komplikovanější řešení, jelikož se na trhu vyskytuje množství firem, které se zabývají výrobou a prodejem modulálního hardwaru a softwaru určeného přímo pro testovací účely a rapid prototyping. Příkladem takové firmy je National Instruments, Keysight nebo dSpace.

1.1 Dostupný software

V současné době se pro tvorbu automatizovaných testovacích stanic po softwarové stránce používá tzv. sekvencerů, což jsou programy, které spustí předem naprogramovanou sekvenci. Tato sekvence se skládá z kroků, které se vykonávají v daném pořadí a takto probíhá jeden testovací cyklus. Obsah těchto sekvencí je možno programovat v různých jazycích a s různými přístupy podle konkrétního dodavatele software. V následujících podkapitolách zmíním nejznámější z nich.

1.1.1 TestExec

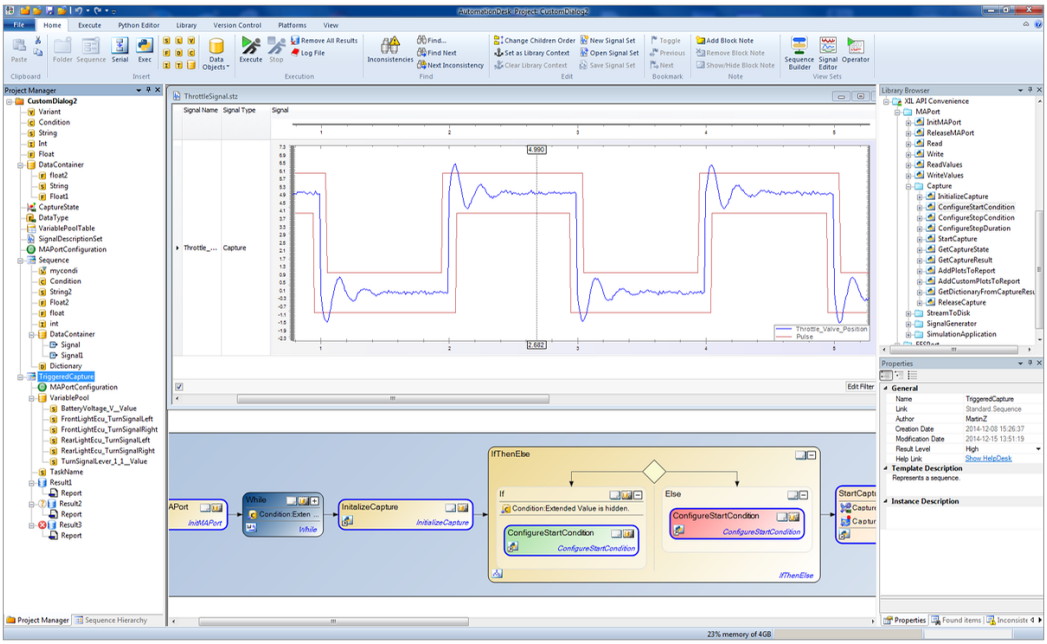
TestExec je sekvencer od firmy Keysight. Na programování kroků sekvence nabízí vlastní grafické vývojové prostředí VEE Pro. Jako další možnost pro vytvoření kroků sekvence lze použít C/C++ a NI LabVIEW. V tomto programu lze přímo pomocí Topology editoru definovat hardwarové zapojení a switche, které se nachází ve fixtuře ovládané aplikací. Data lze sdílet se všemi Keysight aplikacemi, případně vyexportovat ve formátu XML. Poslední verze programu umožňují také paralelní testovaní několika DUT najednou. TestExec neposkytuje tolik možností jako následující dva sekvencery a dále má v porovnání s nimi relativně zastaralé uživatelské prostředí.



Obr. 1.1: Ukázka uživatelského prostředí softwaru TestExec. Převzato z: [4].

1.1.2 AutomationDesk

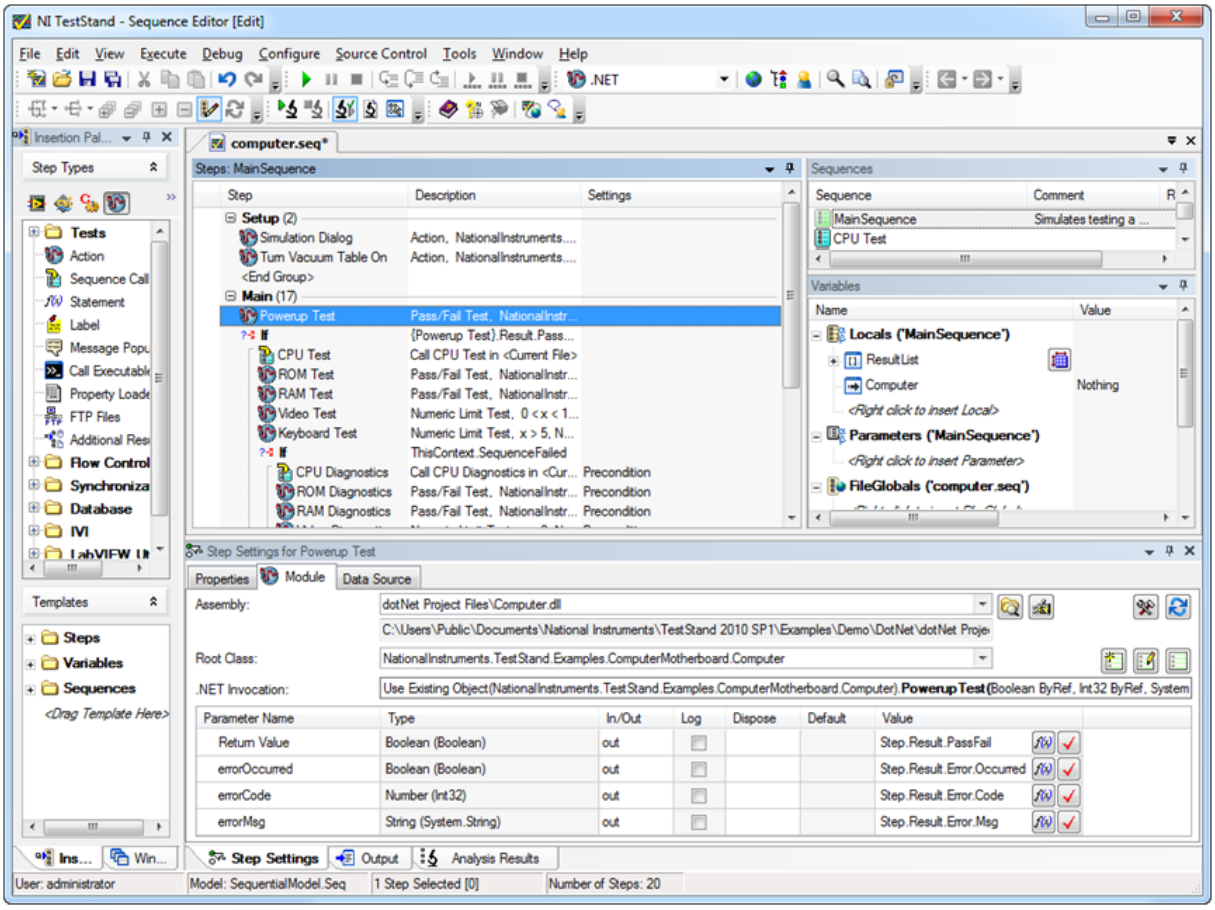
AutomationDesk je program od firmy dSpace. Sequence builder v této aplikaci umožňuje návrh testovacích sekvencí založených na principu UML modelování. Kroky sekvence je možné psát buď v jazyce založeném na jazyce Python a nebo v grafickém programovacím prostředí. Grafický programovací jazyk se většinou používá při přístupu k hardware a nebo řízení toku programu. Python skripty se častěji uplatní při tvorbě algoritmů a nebo uživatelských rozšíření. Tento software a také odpovídající hardware od firmy dSpace je schopen komunikovat s Matlabem a spouštět jeho Mfile. Testovací prostředí také umožňuje testování aplikací běžících v reálném čase. K tomuto využití poskytuje dSpace velké množství knihoven. Dále je možné spustit testovací sekvenci v debugovacím módu a přesně trasovat vykonávání testovací sekvence. Nevýhodou tohoto řešení je vyšší finanční náročnost a nutnost použít originální hardware firmy dSpace. [6]



Obr. 1.2: Uživatelské rozhraní programu AutomationDesk. Převzato z: [5].

1.1.3 TestStand

Posledním zde zmíněným sekvencerem je produkt TestStand od firmy National Instruments. Tento software patří k nejpoužívanějším, hlavně díky kompalibilitě s testovacím hardwarem od NI a grafickým programovacím jazykem LabVIEW také od firmy NI. Jazyk LabVIEW je hojně používán při tvorbě měřicích aplikací. Jelikož tento jazyk bude použit dále k realizaci diplomové práce, budu se mu proto více věnovat v jedné z následujících kapitol. TestStand umožňuje použití modulů naprogramovaných v jazycích C, C++, HTBasic, .Net, dále použití prvků ActiveX a samozřejmě proprietárních jazyků od firmy NI a to LabVIEW a speciálně upravený jazyk C, který se používá v programu CVI/Labwindow. TestStand neumožňuje real-time testování. Tuto oblast pokrývá produkt Veristand, také od firmy NI. TestStand umožňuje volat testovací sekvence z aplikace napsané v LabVIEW a toto se uplatní především při tvorbě uživatelských rozhraní a také tvorbě složitějších testovacích aplikací. Pomocí událostí generovaných v TestStandu je možná integrace sekvencí i do složitějších programovacích celků. Výsledky testů lze vyexportovat ve formátu XML, txt, HTML a do databázového souboru. [12]



Obr. 1.3: Uživatelské prostředí programu TestStand. Převzato z: [7].

2 AKTUÁLNÍ STAV UCHOVÁVÁNÍ DAT

2.1 Popis testovací části linky

V našem modelovém případě se část výrobní linky, kde probíhá testování, skládá ze 4 stanic. Jmenovitě z programovací stanice, dále stanice stanice kontroly PCB a těsnosti kompresoru, dále stanice, kde probíhá funkční test a poslední, kde dochází k měření charakteristik. Všechny tyto stanice jsou v tomto případě obsluhovány operátory výroby, kteří založí nový testovaný kus a po konci testu testovaný kus podle výsledku zařadí mezi úspěšně otestované, případně ty, co testy neprošly. V rámci úspory času, v okamžiku, kdy neprojde jakýkoliv z testovacích kroků na libovolné testovací stanici, je celý test ukončen a chybná jednotka vyřazena a následně poslána na další posouzení do oddělení kvality výroby. V případě, že je jednotka opravena, je opět poslána na celý testovací úsek. Proto je možné, že jedna jednotka se svým unikátním sériovým číslem projde celým testovacím úsekem několikrát. Na následující straně se nachází tabulka se vzorovými daty pro první programovací stanici 2.1 a čtvrtou měřící stanici 2.2. Všechny čtyři stanice jsou přímo přes Ethernet připojeny přes interní síť k serveru, na kterém běží databázový stroj. Z důvodu zachování bezpečnosti sítě není tato síť připojena k internetu.

2.2 Uchovávání naměřených hodnot

Aktuálně se naměřené hodnoty pro dané DUT zapisují do CSV souboru. Tento soubor je uložen na dané testovací stanici, na jejím pevném disku. Aktuálně každá testovací stanice obsahuje svůj vlastní soubor, do kterého zapisuje hodnoty pro jednotlivé jednotky, které byly otestovány. Pro čtyři ukázkové stanice by existovaly čtyři různé soubory, které nejsou vzájemně propojeny a nejsou průběžně zálohovány na vzdáleném serveru.

2.3 Cílový stav

Jako ideální stav pro uchovávání naměřených dat byla vyhodnocena databáze na vzdáleném serveru, který bude součástí produkční sítě.

Tab. 2.1: Data za stanice 1 - Programovací stanice.

Kód kroku	Název kroku	Komentář	Parametry	Dolní limit	Horní limit	Zapisovaná hodnota	Datový typ	Jednotka
P001	Napětí PCB			12	14	Měřené napětí	double	volt
P002	Proud PCB			0,1	0,4	Měřený proud	double	ampér
P003	Kontrola stavových bitů			0	FFFF FFFF		U32	
P004	Datum den			1	31	den	U8	
P005	Datum měsíc			1	31	měsíc	U8	
P006	Datum rok			1	31	rok	U8	
P007	Flashování firmware	Úspěch flashování	HEX file pro flashování	1	1		bool	
P008	Flashování kalibračních dat	Úspěch flashování	HEX file pro flashování	1	1		bool	

Tab. 2.2: Data za stanice 4 - Stanice pro měření charakteristik.

Kód kroku	Název kroku	Komentář	Parametry	Dolní limit	Horní limit	Zapisovaná hodnota	Datový typ	Jednotka
P001	Vymazání chybové paměti		Parametrem je, zda pamět mazat	1	1		bool	
P003	Natlakování měřícího okruhu kapaliny			1000	1300 3		double	Pa
P004	Kontrola teploty vzduchu			20	60		double	Pa
P005	Kontrola napájení kompresoru			12	13	Napětí	double	°C
P006	Měření průtoku v pracovním bodě			115	125	průtok	double	l/s
P007	Měření tlaku v pracovním bodě			6,8	7,2	tlak	double	bar
P008	Měření proudu v pracovním bodě			7	11	proud	double	A
P009	Měření napětí v pracovním bodě			12	13	napětí	double	V
P010	Kontrola, že měření proběhlo v pořádku			1	1		bool	
P011	Kontrola chybové paměti			1	1		bool	

3 TVORBA DATABÁZE

3.1 Výběr vhodné databáze

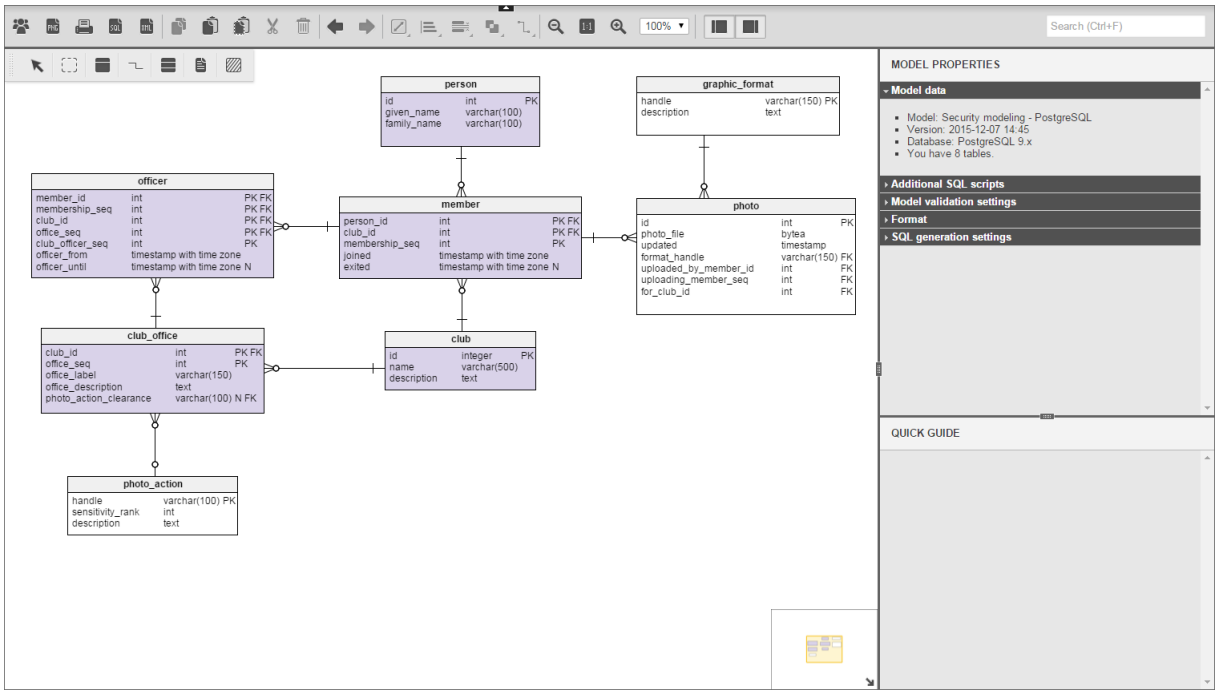
V dnešní době je možné rozdělit databázové technologie na dva hlavní proudy. Prvním jsou relační databáze, někdy referované také jako SQL databáze. Druhou skupinou jsou noSQL databáze. NoSQL zkratka sdružuje alternativy k SQL databázím, NoSQL v originále odkazuje na význam „Not only SQL“, což se dá volně přeložit jako „Nejen SQL.“. Tato skupina obsahuje různé způsoby k přístupu ke skladování dat, které poskytují menší či větší flexibilitu s porovnání s relačními databázemi. [2]

3.1.1 Relační databáze

Jak už jsem zmínila, hlavní rozdíl mezi SQL a NoSQL databázemi je v přístupu k ukládání dat, což je hledisko podle kterého je třeba se rozhodnout jakou databázi využít. Relační databáze skladují veškerá data ve formě dvourozměrných tabulek. Mezi těmito tabulkami je možné vytvářet vzájemné vztahy - relace. Odtud název relační databáze. Tabulky je poté možné spojovat do pohledů, což jsou data v „tabulkách“, které nakonec vidí uživatel ve svém grafickém prostředí. Důležitým faktorem při tvorbě této databáze je návrh vhodného datového modelu, který vystihne všechny vztahy, tedy relace, mezi daty uloženými v tabulkách. Také je nutné vzít v úvahu všechny možné situace a požadavky, které mohou přijít od uživatele a také zadání od zákazníka. Poté, co je datový model jednou vytvořen, je k dispozici efektivní způsob skladování dat s rychlou odezvou. Pokud je jednou datový model vytvořen a do databáze jsou uloženy data, bývá velmi obtížné tento datový model změnit. Toto řešení je vhodné, pokud dopředu známe strukturu uložených dat a neočekáváme, že se bude v průběhu času měnit. Pokud se má struktura dat v průběhu života databáze měnit, je třeba to zahrnout již do datového modelu při tvorbě databáze. Nejznámějšími firmami, které poskytují tento typ databází je mySQL, Oracle a MS SQL Server. [1]

3.1.2 NoSQL

Tato skupina databází přichází do obliby v poslední dekádě a to hlavně v souvislosti s tvz. Big Daty. Tento typ databází je vhodný v případě, že je třeba skladovat velké množství nestrukturovaných nebo obtížně strukturovatelných dat. Dále je to lepší volba v případě, že dopředu neznáme strukturu dat, případně očekáváme, že se bude v průběhu času výrazně měnit. Data v těchto databázích jsou spíše než v



Obr. 3.1: Relační model vytvořený v aplikaci Vertabelo. Převzato z: [8].

tabulkách skladována ve složkách podle jejich vzájemné spojitosti. Jeden tento soubor ve složce databáze může uchovávat různé datové typy, jako například fotografie, text, hypertextové odkazy a další atributy, které se ve velkém využívají například na blogu. Proto jsou NoSQL databáze často označovány jako „dokumentově orientované“. Dále také poskytují lepší přístup k datům pro vývojáře aplikací, kteří nejsou nuceni se učit jazyk SQL a studovat strukturu databáze. Data jsou často skladována v JSON formátu a jemu podobných formátech. Data jsou ukládaná v souborech na principu klíčhodnota, případně na principu dokumentové databáze. Tato databáze rozšiřuje princip klíčhodnota a přidává další stupně komplexnosti. Každý dokument v tomto typu databáze má své data a unikátní klíč, podle kterého je ho možno dohledat. Toto je případ nejpopulárnější databáze z této skupiny a to MongoDB.[2]

3.1.3 Volba databáze

S přihlédnutím k výhodám a nevýhodám obou typů databází jsem se rozhodla k využití databáze typu SQL. Data získávaná z testovacích stanic jsou zatím ukládána ve formě tabulek v programu MS Excel. Tento formát dat je dobře strukturovatelný a rozčlenitelný do tabulek - entit v databázovém modelu SQL databáze. Struktura dat je také známá již nyní a nelze očekávat žádné větší změny ve struktuře datového modelu. Dále mám možnost ovlivnit formát dat získávaných ze zmiňovaných stanic a v případě potřeby mohu data upravit do potřebného formátu v aplikaci nahráva-

```
{
    '_id' : 1,
    'name' : { 'first' : 'John', 'last' : 'Backus' },
    'contribs' : [ 'Fortran', 'ALGOL', 'Backus-Naur Form', 'FP' ],
    'awards' : [
        {
            'award' : 'W.W. McDowell Award',
            'year' : 1967,
            'by' : 'IEEE Computer Society'
        }, {
            'award' : 'Draper Prize',
            'year' : 1993,
            'by' : 'National Academy of Engineering'
        }
    ]
}
```

Obr. 3.2: Příklad dokumentu v JSON formátu používaném databází MongoDB. Převzato z: [3].

jící data z testovací stanice na server. V tomto případě je výhodné použít relační databázi, protože nám poskytne rychlou odezvu a efektivní způsob skladovaní dat, což jsou dva z hlavních požadavků na vznikající databázi.

Z vyjmenovaných poskytovatelů databázového software byl vybrán MS SQL Server. Jelikož se daná databáze bude používat v komerční sféře, je nutno přihlédnout k požadavkům zákazníků. MS SQL Server je již u valné většiny zadavatelů projektů používán a jejich IT oddělení jsou schopna novou databázi velmi rychle integrovat do své stávající infrastruktury, bude použit i v této diplomové práci. Další výhodou využití tohoto software je podpora od firmy National Instrument, která garantuje kompatibilitu Labview Connectivity toolkitu s MS SQL server a Oracle, ale negarantuje kompatibilitu s mySQL. [13]

3.2 Datový model

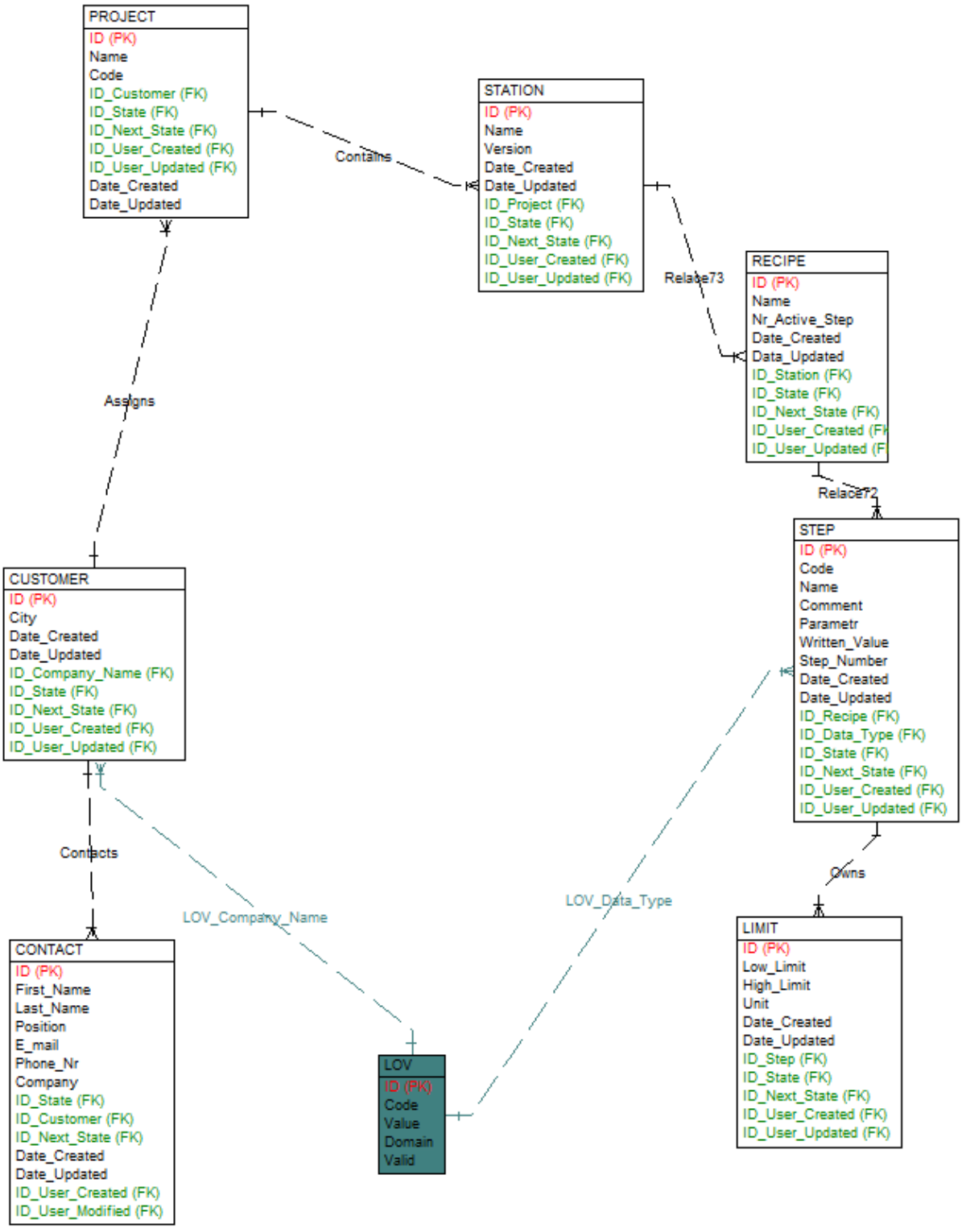
S ohledem na současnou funkcionalitu testovacích stanic a strukturu dat, která se budou uchovávat v databázi, byl vytvořen následující datový model. Celkový pohled na tento model je v příloze této práce. Záznamy do tabulek s bílým pozadím budou plněna manuálně pomocí uživatelského prostředí a uchovávána v databázi pro celou dávku testovaných jednotek. K těmto tabulkám se pro každé DUT testované na

stanici programově vytvoří záznamy pro tabulky se zeleným pozadím. Žlutě podbarvené tabulky zabezpečují správu databáze a to uživatele, uživatelské role a povolení. Modře podbarvené entity jsou tabulka stavů a tabulka přechodů. Poslední tabulka je šedě podbarvený číselník. Dále je popsán návrh datového modelu podrobněji.

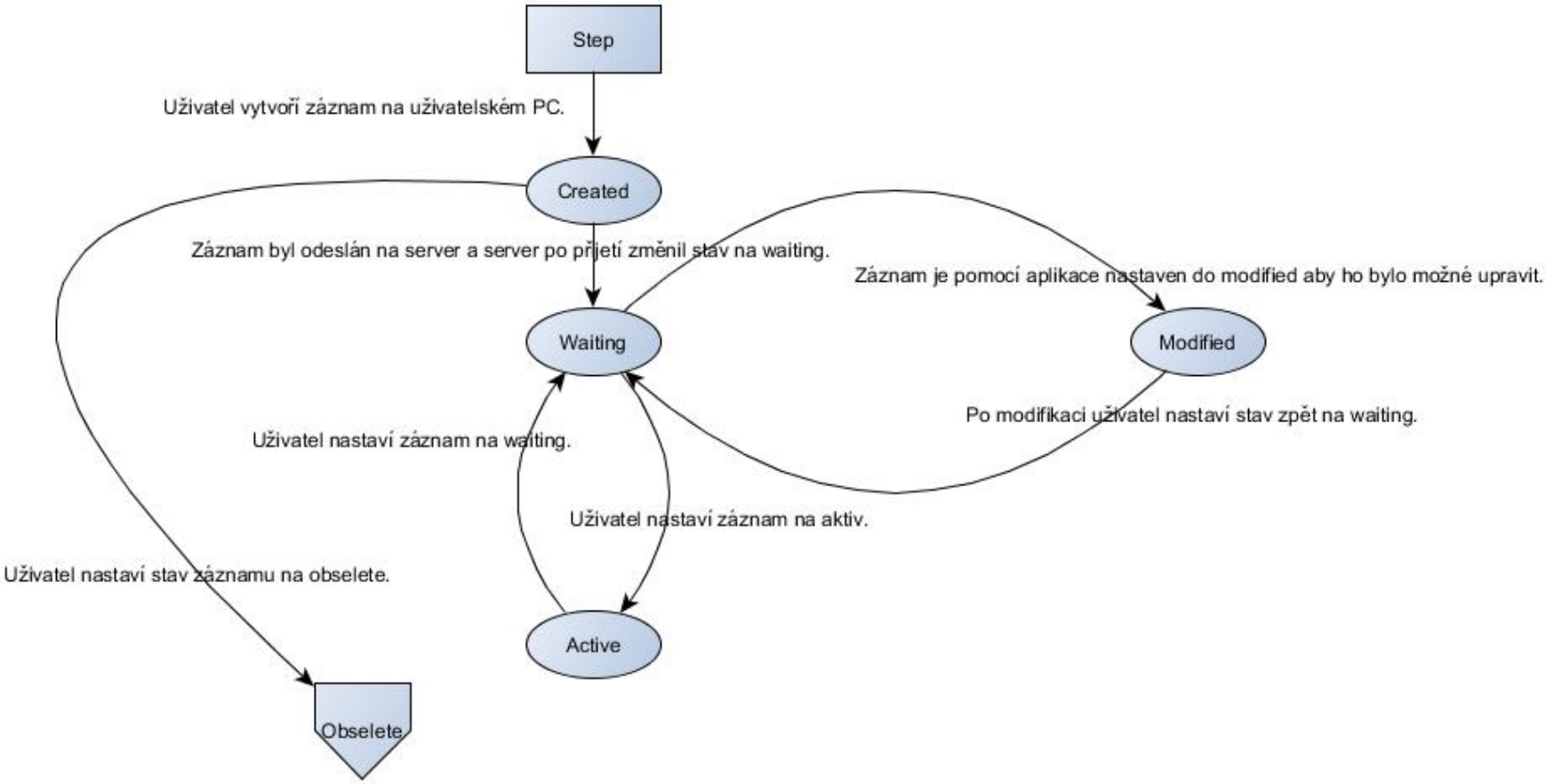
3.2.1 Životní cyklus záznamů tvořících metadata

Zákazník zadává projekt, který se skládá z testovacích stanic. Zákazník má také přiřazen záznam kontakt, kde jsou uloženy informace o kontaktních osobách pro daný projekt. Tabulka stanice reprezentuje testovací stanici, na které běží testovací sekvence. Testovací sekvence má přidělen svůj recept a ten se skládá z kroků sekvence, které jsou zapsány v tabulce Step. Každá stanice má přidělené kroky sekvence v daném pořadí a dohromady tvoří aktuální předpis testu. Každý krok má přiřazené odpovídající limity, tedy horní a dolní přípustnou mez naměřených hodnot, které určují úspěch testovacího kroku. Všechny tabulky v této části datového modelu obsahují atributy State a Next state, tedy v jakém se aktuálně nachází stavu a do jakého následujícího stavu se dostanou. Stavový a přechodový diagram těchto entit zachycuje obrázek 3.4.

Jako vzorová entita je uvedena tabulka Step. Tento přechodový diagram platí pro všechny tabulky s bílým pozadím. Jako první je záznam v tabulce vytvořen a po kontrole uživatelem označen jako „Created“. Poté je pomocí aplikace zapsán do databáze na serveru a přejde do stavu „Waiting“. Aby mohl být tento krok použit při testovaní na testovacích stanicích, je nutno záznam ručně nastavit do stavu „Active“. Poté, co předpis který obsahuje není dále používán v testovací stanici, je možno záznam nastavit do stavu „Obselete“ a ten již není zobrazen v uživatelské aplikaci, ale je pouze uložen na serveru.



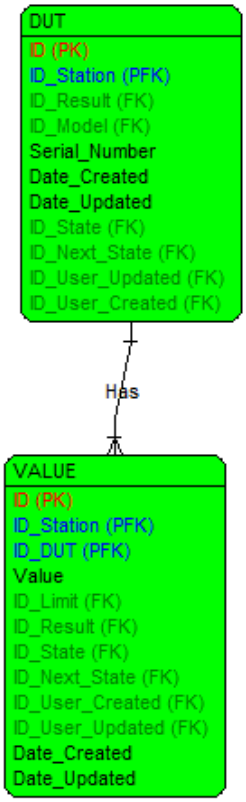
Obr. 3.3: Část datového modelu zachycující předpis testů.



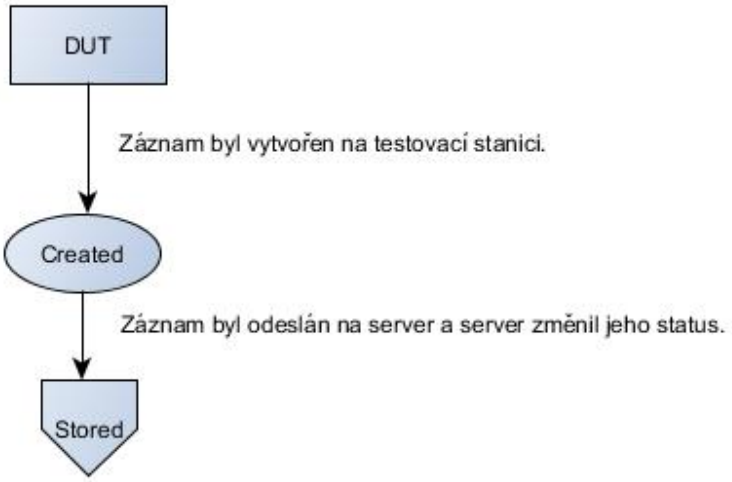
Obr. 3.4: Diagram stavů a přechodů pro záznam v tabulce Step.

3.2.2 Záznamy tvořené na testovacích stanicích

Záznam pro tabulku DUT je vytvořen na každé testovací stanici pro každé nové DUT a je tvořen bez přímého spojení s databází. Jelikož testovaná jednotka prochází postupně několika testovacími stanicemi a v případě neúspěšného testu je možný i opakovaný průchod testovací stanicí, je primární klíč autoinkrementující se číslo na příslušné stanici. Druhý segment primárního klíče je cizí klíč, který tvoří primární klíč příslušné stanice. Takto lze jednoznačně identifikovat průchod testované jednotky stanicemi. Každý záznam obsahuje také atribut sériové číslo, pomocí kterého se filtruje konkrétní záznam pro testovanou jednotku v uživatelských pohledech. Toto zaručí, že průchod konkrétní testované jednotky konkrétní testovací stanicí bude z hlediska záznamů v databázi unikátní. Na podobném principu probíhá i identifikace naměřené hodnoty Value. Tento záznam má tří segmentový primární klíč, který se skládá ze segmentu unikátní ID automaticky přiřazené testovací stanicí, primárního klíče testovací stanice a primárního klíče záznamu DUT. Jako cizí klíč také tabulka obsahuje primární klíč záznamu v tabulce Limit, tedy podle jakých limitů byla vyhodnocena úspěšnost testu.



Obr. 3.5: Entity DUT a Value, zapisované na stanici.



Obr. 3.6: Diagram stavů a přechodů pro entitu DUT.

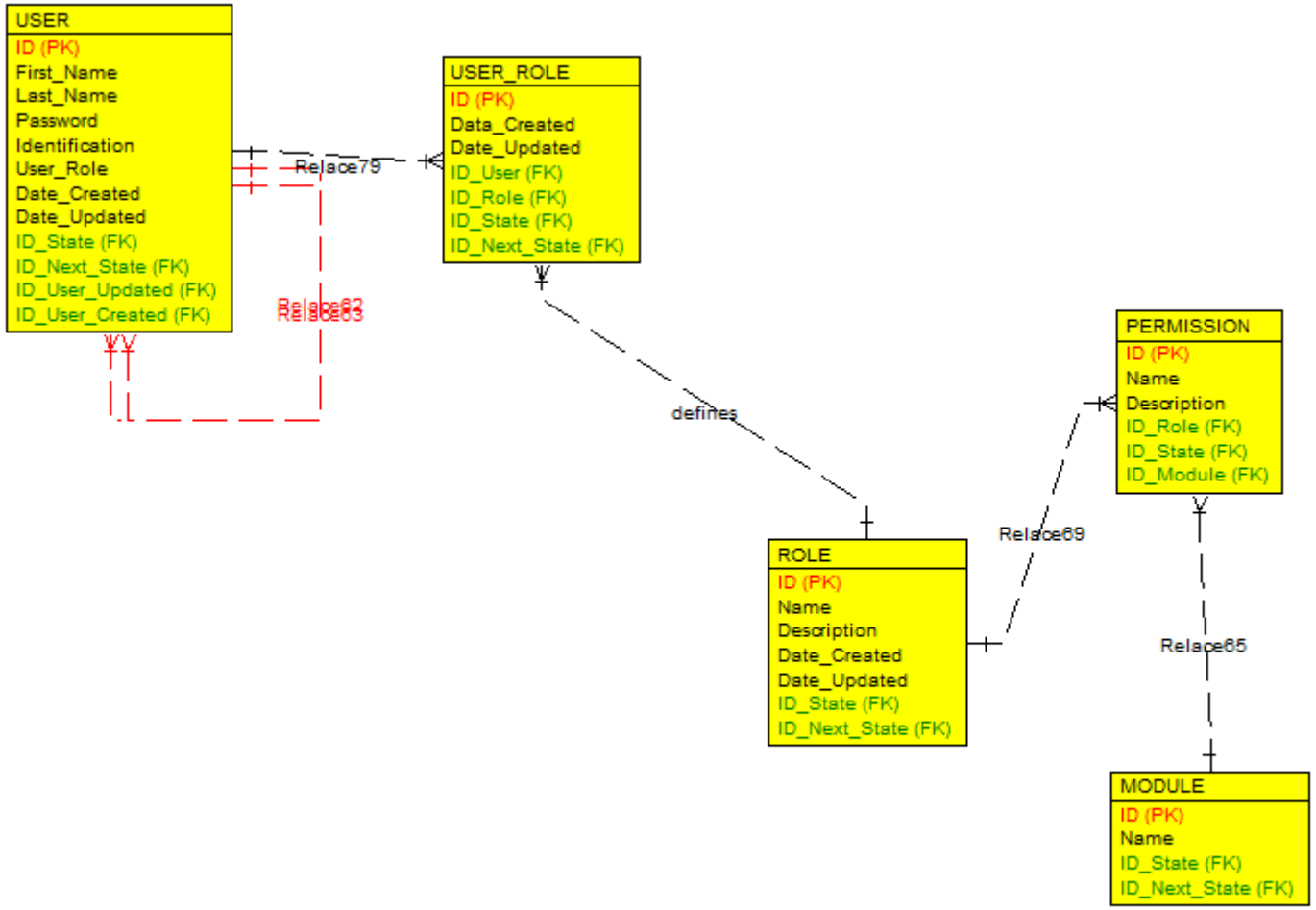
Cílem této databáze je uchovávání naměřených hodnot pro jednotlivé testované jednotky, tedy zápis záznamů do tabulek DUT a Value. Tyto záznamy nabývají stavů „Created“, kdy jsou vytvořeny na testovací stanici a poté jsou zapsány na server do databáze. Při zápisu do databáze server změní stav záznamu na „Stored“. Po vytvoření není možná žádná další úprava ani manipulace s naměřenými daty.

3.2.3 Správa uživatelů a uživatelských oprávnění

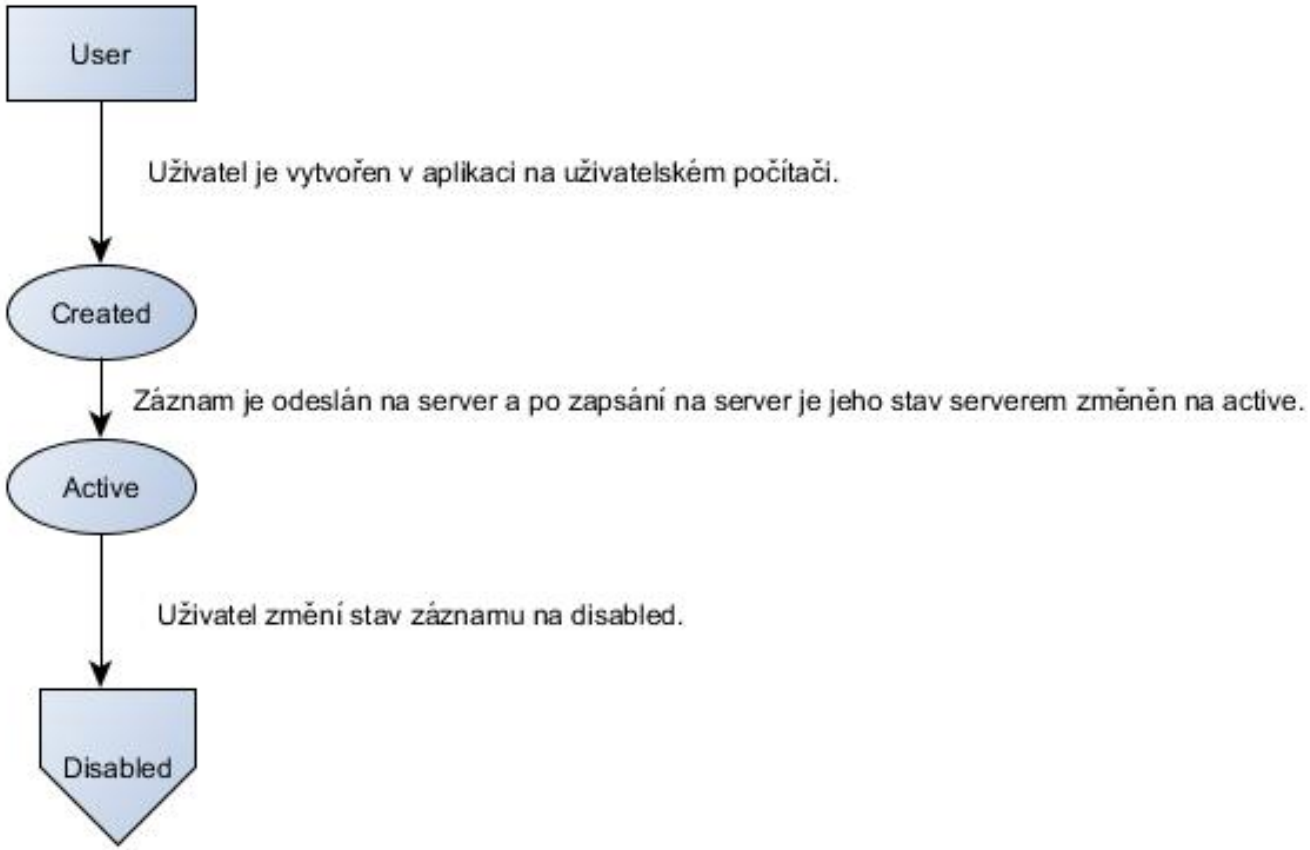
Přístup do databáze, ať už prostřednictvím aplikace pro zapisování dat na testovací stanici a nebo přes uživatelské rozhraní má větší množství uživatelů s různými oprávněními. Dále z důvodu dohledatelnosti každý záznam v databázi, se kterou může přijít uživatel do styku, obsahuje atribut kdo ji vytvořil a také který uživatel ji naposledy editoval. Z důvodu lepší přenositelnosti datového modelu mezi různými databázovými stroji je správa uživatelů a jejich oprávnění zahrnuta v datovém modelu. Následující obrázky zobrazují tabulky datového modelu zodpovědné za správu uživatelů a diagram stavů a přechodů pro tabulku Uživatel.

3.3 Procesní model

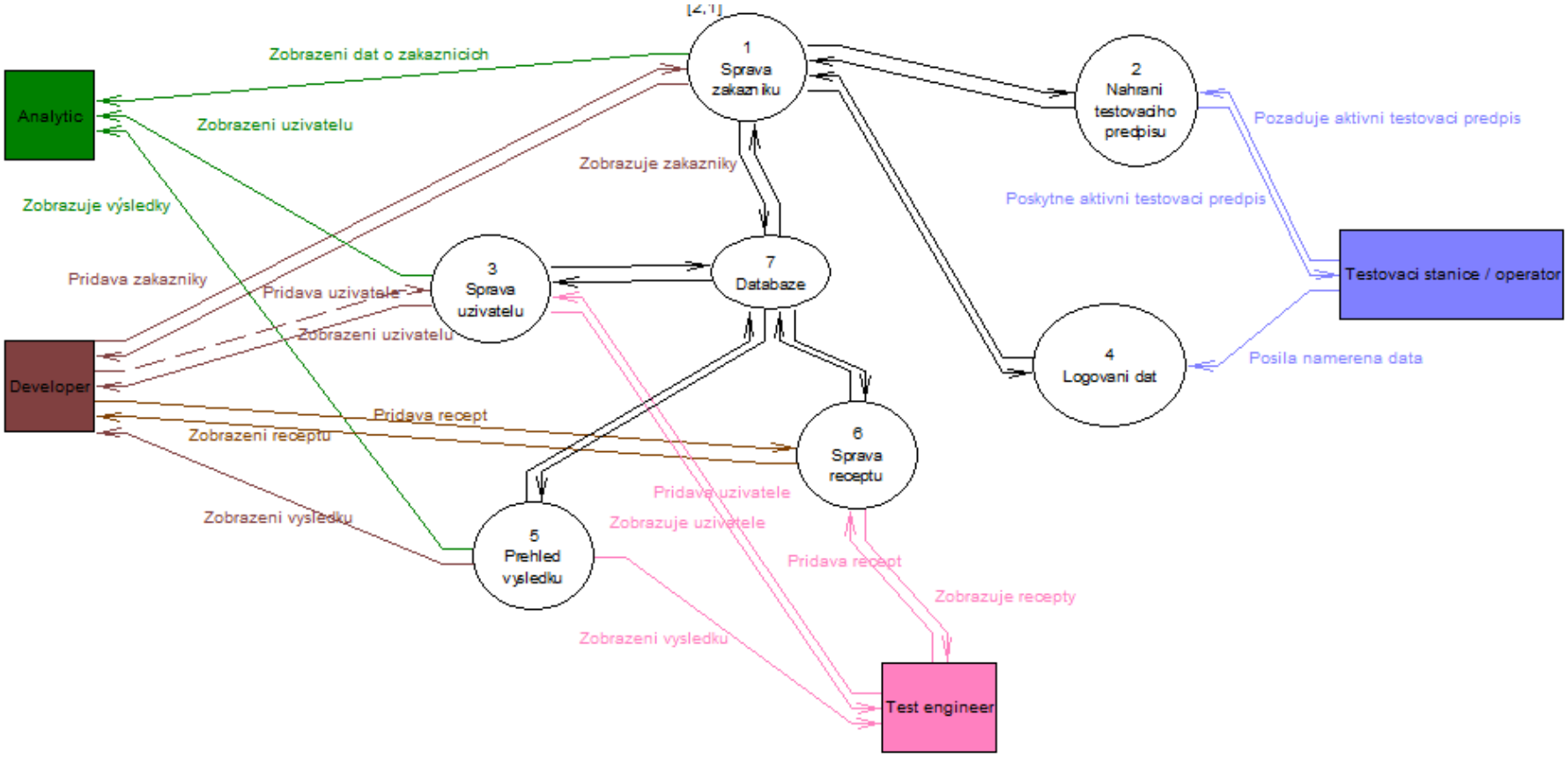
Následující obrázek 3.9 zachycuje procesní model pro vytvořenou databázi.



Obr. 3.7: Entity pro správu uživatelů a uživatelských povolení.



Obr. 3.8: Diagram stavů a přechodů pro tabulku User.



Obr. 3.9: Procesní model.

4 APLIKACE PRO PŘÍSTUP K DATABÁZI

Cílem bylo vytvoření dvou aplikací, které budou fungovat nezávisle na sobě. První je zodpovědná za vyčítání naměřených dat ze sekvencí a jejich zápis do databáze. Tato aplikace bude běžet na libovolné testovací stanici. Druhá aplikace slouží jako uživatelský interface pro náhled do databáze a zobrazení výsledků. Tato aplikace je spustitelná na jakémkoliv počítači. Primárním jazykem pro vývoj obou aplikací bude LabVIEW od firmy National Instruments.

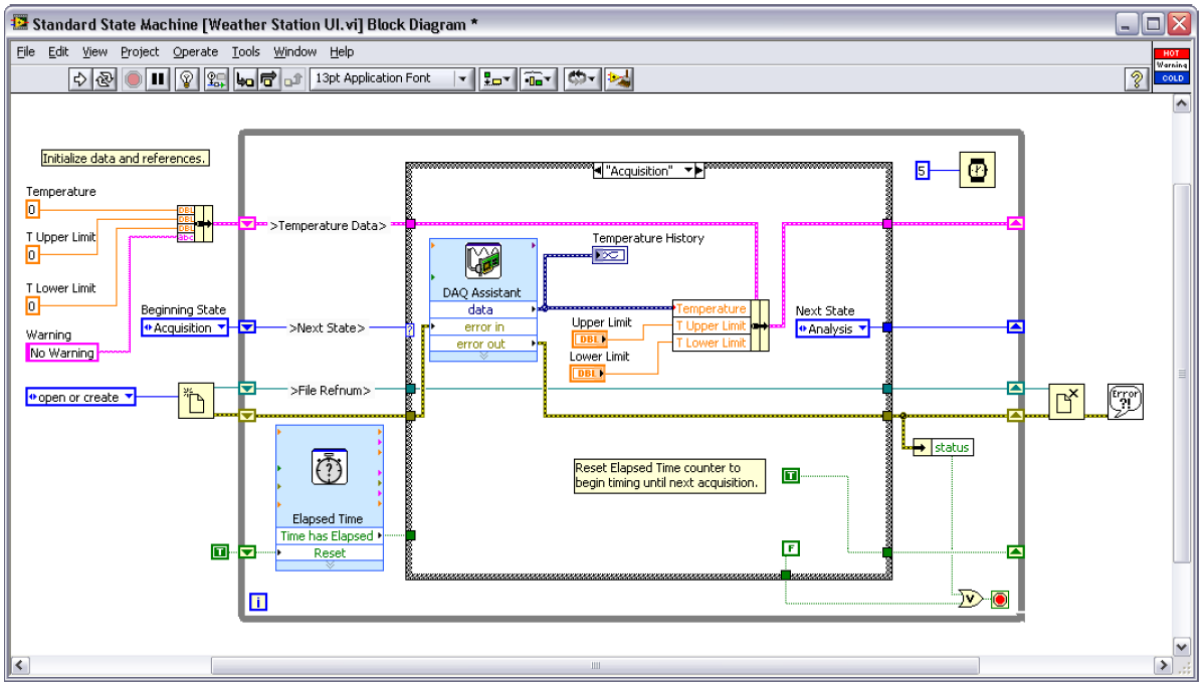
4.1 LabVIEW

LabVIEW je grafický programovací jazyk, který byl původně vyvinut pro tvorbu měřicích řetězců. V dnešní době má širokou škálu využití a to programování FPGA, sběr a vyhodnocování dat, obsluha měřicích přístrojů, tvorbu uživatelských prostředí a nebo správu hardware. Hlavní výhoda tohoto jazyka je, že se jedná o programovací jazyk určený pro inženýry a vědce a umožňuje jednoduchou správu vstupů a výstupů, cyklické vyčítání a zpracování dat a obsluhu měřicích zařízení. Umožňuje i tvorbu vícevláknových aplikací. Nová nástavba Actor Framework umožňuje tvorbu modulů a jejich vzájemnou komunikaci a tím podporuje tvorbu složitějších a více komplexních aplikací. V LabVIEW můžou být použity také moduly vytvořené v jiných programovacích jazycích jako je C/C++, VisualBacis a nebo Fortran.[9]

LabVIEW také umožňuje objektově orientovaný design jak je známý z jiných objektově-orientovaných jazyků. Lze vytvořit třídu, která obsahuje cluster(společnou strukturu proměnných) privátních dat třídy, která je přístupná pouze pro metody třídy. Dále je možno vytvářet „private“, „protected“ a „public“ metody třídy. Oproti jiným jazykům umožňuje i tvorbu tvz. „communite metod“, což znamená, že přístup mají pouze VI ve stejné třídě, ve stejné knihovně a nebo ve „friend“ třídě a nebo knihovně. Je možné využít mechanismus podobný přetěžování metod, který je v LV znám jako „dynamic dispatch“. Tento mechanizmus umožňuje vytvořit nové VI jako potomka jiného VI se stejným jménem, který se liší vstupními parametry. Až za běhu programu je podle vstupních dat rozhodnuto, které konkrétní VI bude použito[14].

4.1.1 Řízení toku dat

Základním způsob vykonávání programu v LV je paralelní, program se tedy v blokovém diagramu vykonává po směru drátu zleva doprava. Pokud je třeba zajisti vykonání určitých operací, jako třeba volání VI, v pevně daném pořadí, je třeba zapojit uvchybový(error) drát postupně všemi VI. „Chybový drát“ zajistí postupnou



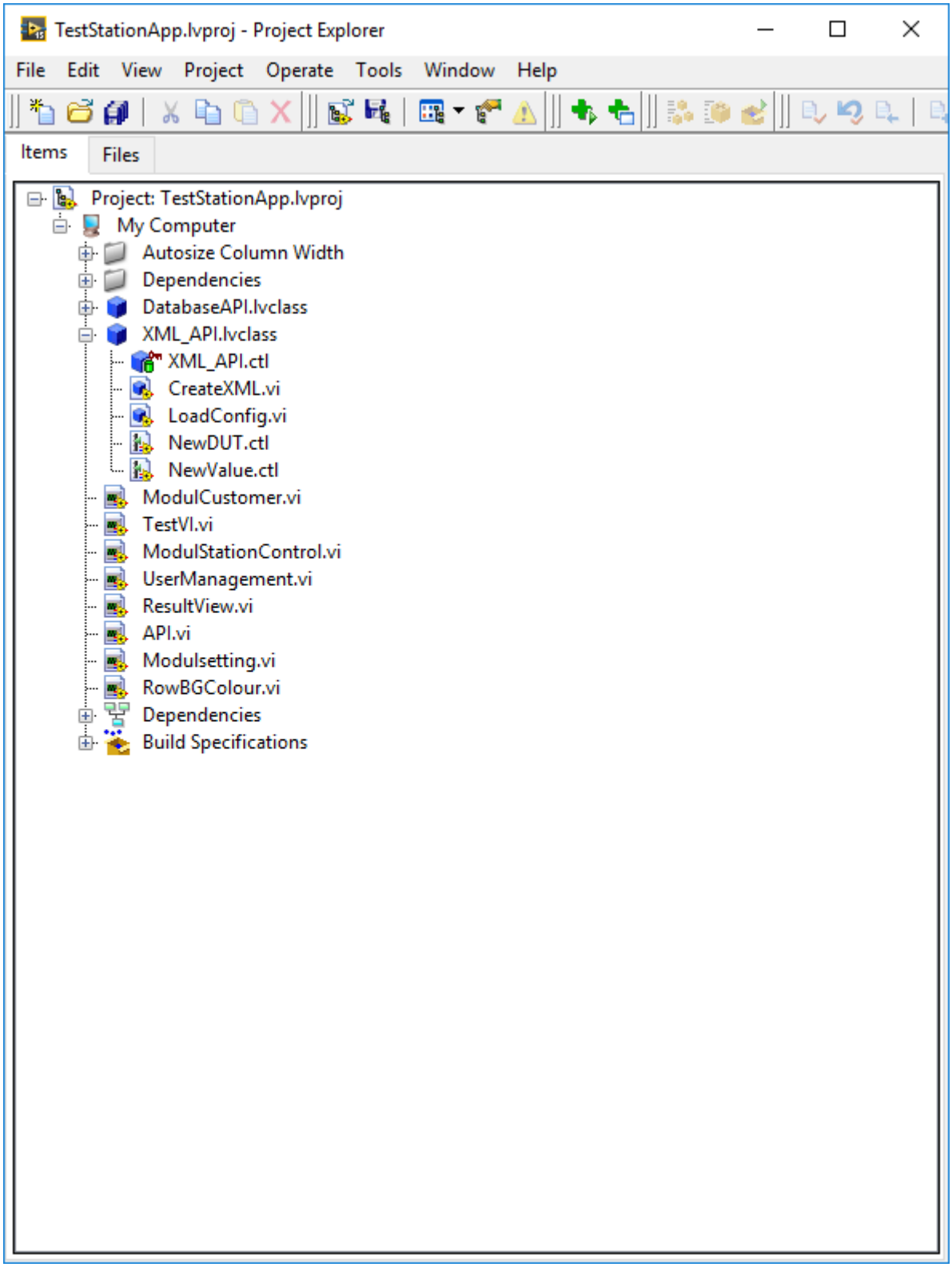
Obr. 4.1: Ukázka zdrojového kódu aplikace v LabVIEW. Převzato z:[10].

exekuci v pevně daném pořadí. Dále existují lokální proměnné pomocí kterých lze asynchronně za běhu programu vyčíst či nastavit hodnotu proměnné. Dále každý grafický prvek na Front panelu má vlastní property node, který nám umožňuje nastavovat jeho vlastnosti jako třeba velikost, vzhled, případně hodnotu a simulovat události na daném prvku za běhu programu[16].

4.1.2 Actor Framework

Jak již bylo zmíněno, Actor Framework je nástavba, která umožňuje tvorbu komplexních aplikací. Základním principem je, že vše je zpráva. Architektura je následující: vše se skládá z tzv. „actorů“. Actor je prvek, který obsahuje „main VI“, které by se dalo přirovnat k funkci main u strukturovaného programování. Zde se umístí vlastní kód, který má tento modul vykonávat. Dále existují VI, které se provedou před spustěním „main VI“ a po jeho ukončení. Každý actor vlastní frontu pro zprávy a může komunikovat se svým mateřským actorem a dále s actory na stejné úrovni. Komunikuje také svým dětským actorem, tedy tím actorem, který řídí. Celou tuto komplexní architekturu poskytuje Actor Framework.

Díky dědičnosti v LV se nový actor vytvoří tak, že podědí vše výše zmíněné VI z templatu a programátor doplní vlastní fukncionalitu, např. měření napětí pomocí analogové karty, do „main VI“. Do inicializačního VI se doplní inicializace analo-



Obr. 4.2: LabVIEW projekt obsahující dvě třídy vytvořen v průběhu této diplomové práce.

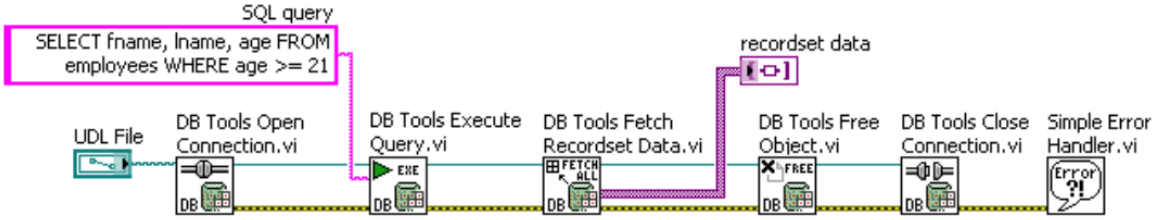
gové karty a do clean-up VI se vloží uzavření komunikace s kartou. Tento modul komunikuje s okolím pomocí normovaných zpráv a např. odesílá naměření hodnoty a diagnostické zprávy. Takto vytvořený a propojený systém actorů umožňuje tvorbu modulárních aplikací, které jsou schopny fungovat paralelně, nezávisle na sobě a je možné moduly libovolně skládat a nebo odebírat. Vše je řízeno hlavním actorem, který je spustěn uživatelem a ten pomocí zpráv předá instrukce svým potomkům, kteří zase spustí své potomky a tak dále [15].

Bylo plánováno využít Actor Framework pro logovací aplikaci, ale bohužel to nebylo možné, protože v LabVIEW verze 2015, která je použita pro tuto práci i ve verzi o rok novější jsou problémy v knihovnou obsahující jádro Actor Frameworku a tato knihovna není spustitelná. Navíc byly reportovány problémy se stabilitou již vytvořených spustitelných aplikací. Proto bylo od využití tohoto Frameworku upuštěno.

4.2 Požadavky na aplikace

4.2.1 Aplikace pro zápis naměřených dat do databáze - logovací aplikace

Tato aplikace bude použita jako rozšíření stávajících systémových řešení firmy Kentigen pro sběr dat z testovacích sekvencí. Pro zápis do databáze je využito LabVIEW rozšíření Connectivity toolkit.



Obr. 4.3: Příklad použití LabVIEW pro Select Statement. Převzato z:[11].

Tato aplikace se skládá z následujících VI:

- VI pro načtení aktuálního testovacího receptu pro danou testovací stanici.
- VI pro načtení primárních klíčů záznamů Station, Step a Limit, které budou figurovat jako cizí klíče u vytvořených záznamů v tabulkách DUT a Value.
- VI, které bude komunikovat s databází pomocí SQL jazyka.
- VI pro ukládání dat do XML souboru pro případ, že není dostupné spojení s databázových serverem.
- VI, který po obnovení spojení načte naměřené hodnoty a odešle do databáze.

- VI pro korektní ukončení aplikace ukončení spojení s databází, uložení aktuálních hodnot počítadel.

4.2.2 Uživatelské prostředí

Druhou vytvořenou aplikací bude User Interface, který umožní přístup do databáze. Toto bude také v LabVIEW. Aplikaci bude možné spustit na jakémkoliv počítači s nainstalovaným LabVIEW Runtime engine. Zde je výčet modulů pro danou aplikaci:

- Přihlášení/odhlášení uživatele do databáze.
- Výběr modulů podle oprávnění uživatele.
- Moduly pro vkládání/úpravu/mazání dat.
- Statistika výsledků testování.

4.2.3 Podpora aplikací na straně databáze

Na straně databáze budou data nachystány pro zobrazení uživateli pomocí pohledů. Data z těchto dohledů budou potom zobrazeny v záložkách uživatelských modulů.

4.2.4 Seznam uživatelských oprávnění

Zde je výčet uživatelských oprávnění, které je možné přidělit uživatelům databáze.

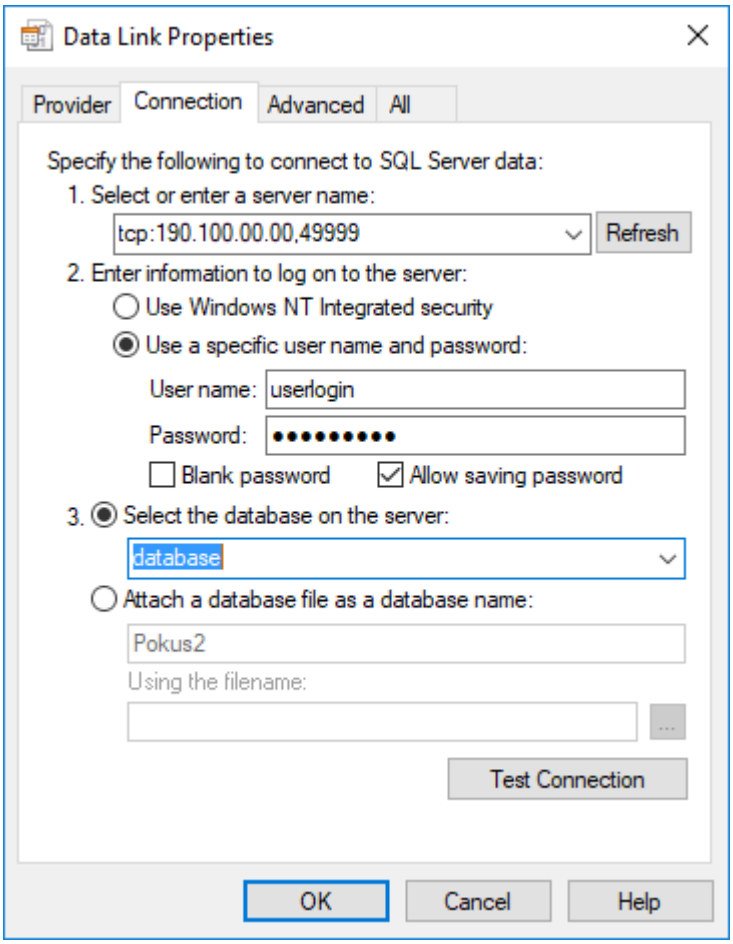
- Testovací stanice - operátor se smí přihlásit do databáze před spuštěním testovacího programu na testovací stanici. Nemá přístup k modulům uživatelského rozhraní.
- Manager má přístup k modulům pro zobrazení výsledků.
- Test Engineer má přístup k modulům pro nastavení předpisu pro testovací stanici, výsledkům a správě uživatelů. Smí přidat uživatele se všemi oprávněními kromě oprávnění Developer.
- Developer má plný přístup ke všem modulům a smí vytvořit uživatele s jakýmkoliv oprávněním.
- Analytic má read-only přístup ke všem modulům.

5 UŽIVATELSKÉ ROZHRANÍ

Uživatelské prostředí umožňuje vložit a upravit, případně smazat metadata tvořící předpis pro testovací stanici. Zároveň umožňuje i zobrazení výsledků. Dále popsané moduly jsou realizovány jako stavový automat, který je řízen událostmi a je možno se dostat z jakéhokoliv stavu do stavu dalšího. Po spuštění se podle uživatelských oprávnění povolí a nebo zakáží tlačítka, poté se začne používat stavový automat, který čeká na vstup od uživatele. Poté co uživatel vyvolá událost, je tato událost obsloužena.

5.1 Navázání spojení s databází

Pro připojení do databáze je využit Microsoft OLE DB Provider for SQL server. Ten je nakonfigurován pomocí Data link properties souboru.



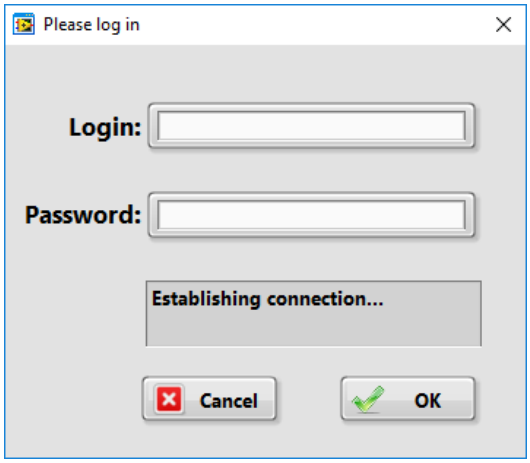
Obr. 5.1: Detail Data link properties souboru.

Tento soubor má příponu „udl“ a je potřeba, aby byl umístěn ve stejné složce jako

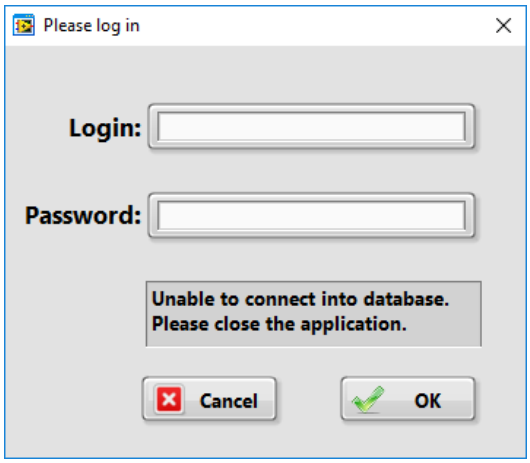
spouštěná aplikace. Podle budoucích požadavků zákazníků je možno umístit tento soubor společně s ostatními konfiguračními souboru do libovolné složky a zadat k ní cestu ve zdrojovém kódu. V tomto souboru se nastaví login a heslo pro přihlášení do databázového serveru. Tento login a heslo bude pro každou stanici unikátní. Poté následuje pokus o přihlášení do databáze. Toto proběhne bezprostředně po kliknutí na „exe soubor“. Uživateli se zobrazí přihlašovací okno, kde je viditelný aktuální stav přihlášení k databázi. Mohou nastat celkem tři možnosti: probíhá připojování k databázi, spojení s databází bylo navázáno a nebo se navázání spojení nezdařilo. Časový limit pro navázání spojení s databází je nastaven na maximálně 15 sekund. Poté je v přihlašovacím dialogu uživatel vyzván, aby aplikaci ukončil.

Aplikace se v tomto případě neukončí automaticky, ale očekává se interakce uživatele, protože z hlediska user experience nejsou vhodné automaticky se ukončující aplikace bez jakéhokoliv vysvětlení.

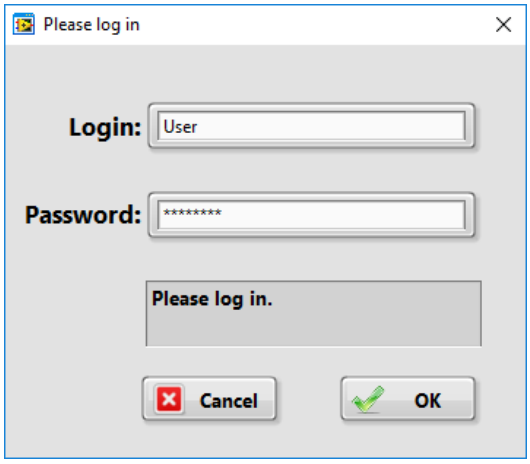
V případě, že spojení bylo navázáno, je třeba se přihlásit pomocí loginu a hesla pro databázového uživatele. Aplikace nereaguje na stisknutí tlačítka „Enter“ po zadání hesla, je nutno kliknout na tlačítko „OK“. Zde se jedná o login do databáze a ne k databázovému stroji jako v případě MS OLE DB Provideru. Jedná se o přihlašovací údaje které byly nastaveny pomocí uživatelského prostředí v modulu „User management“. Heslo je po zadání zašifrováno pomocí MD5 hash funkce. Toto VI bylo převzato z fora pro podporu produktů firmy National Instruments [19]. V případě, že nebyl v databázi nalezen příslušný pár údajů login a heslo, je pole pro zadání hesla smazáno a uživatel je upozorněn, že byl zadán špatný login a nebo heslo.



Obr. 5.2: Modul pro přihlášení navazuje spojení s databází

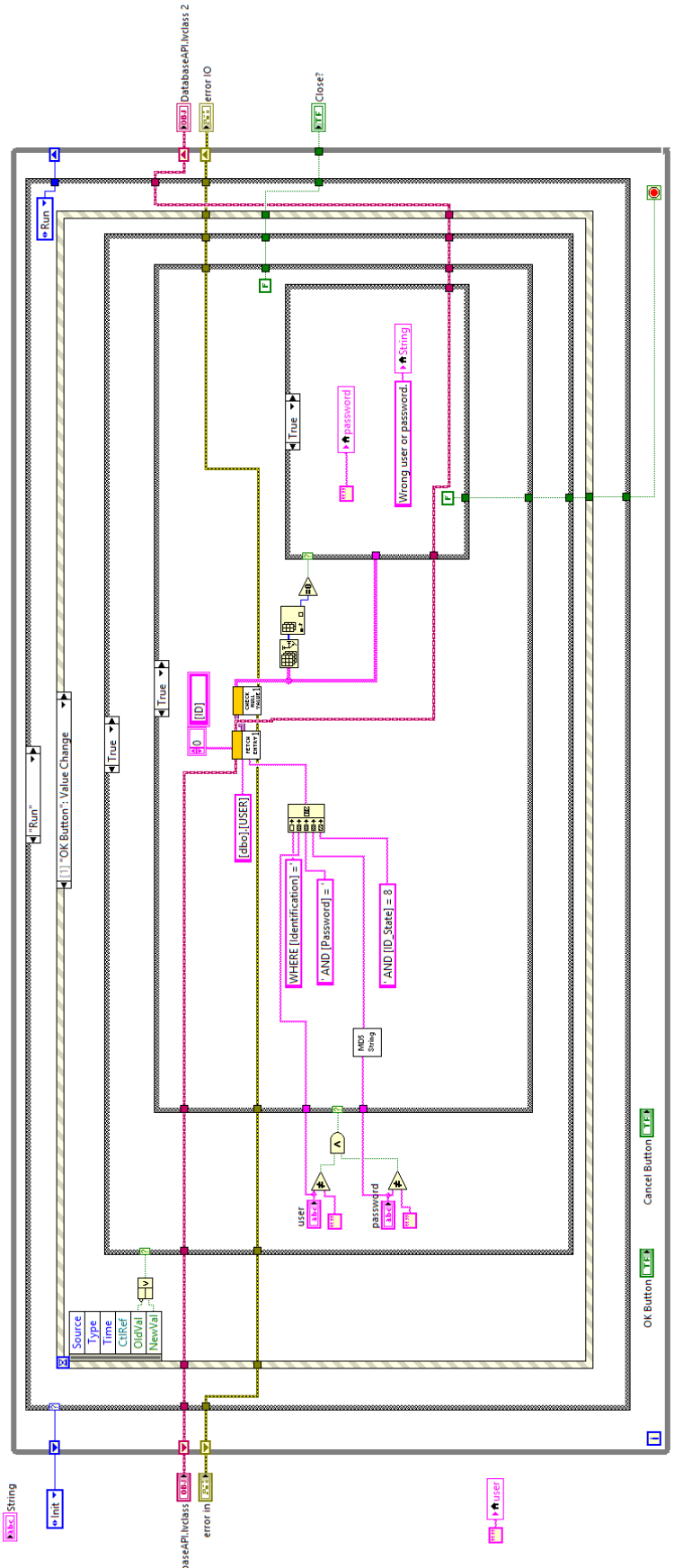


Obr. 5.3: Modul pro příhlášení uživatele po selhání spojení s databází



Obr. 5.4: Modul pro přihlášení připraven k přihlášení uživatele.

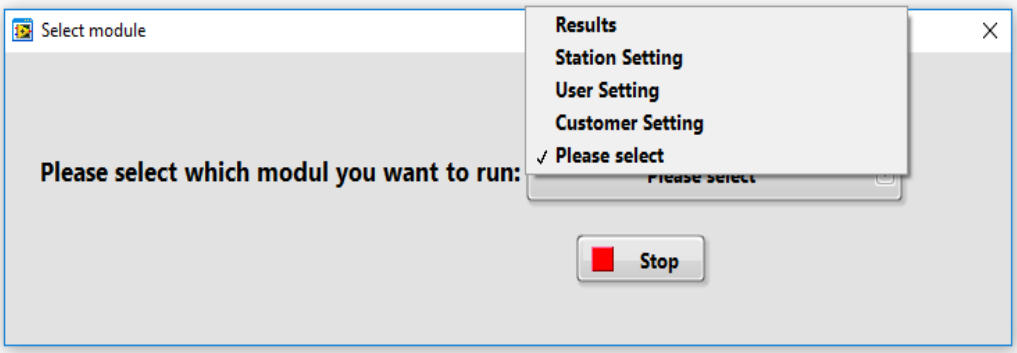
Obr. 5.5: Modul pro přihlášebí uživatele v případě zadání chybného hesla.



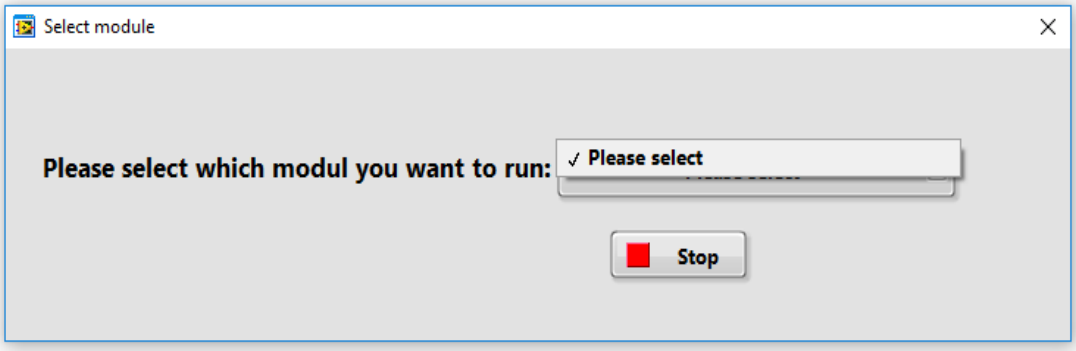
Obr. 5.6: Block diagram pro modul Login.

5.2 Výběr modulu

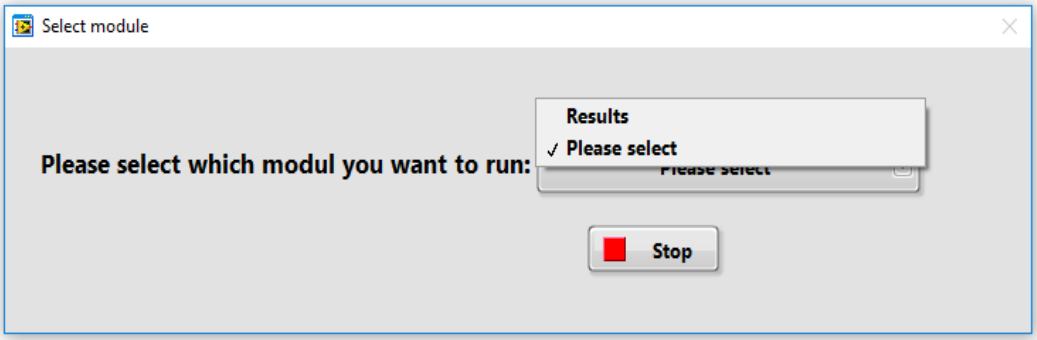
Přihlášení probíhá tak, že se hledá v databázi uživatel s příslušným loginem. Ten byl zadán při vytvoření uživatele jako „Identification“ a musí být unikátní. V databázi má nastaven příznak unique, proto pokus o vytvoření uživatele se stejným loginem není povoleno. V případě, že je poslána taková žádost do databáze, bude zápis ukončen chybovou hláškou. Uživatel musí být nastaven se do stavu „Active user“. Poté, co je nalezen požadovaný uživatel, jsou vylistovány moduly, ke kterým má podle jeho oprávnění přístup. Existují celkem 4 moduly: „Customer Setting“, „User Management“, „Station Setting“ a „Result View“. Zároveň existuje 5 typů uživatelů a to: Operátor/testovací stanice, Test Engineer, Developer, Manager a Analytic. Jejich pravomoci jsou popsány výše v návrhu. Uživatel může vybrat z rozbalovací nabídky, který modul chce spustit. Uživatel vidí pouze ty moduly, ke kterým má přístup, ať už read/write a nebo pouze read. Poté, co uživatel vybere modul, je modul automaticky spuštěn. V případě, že uživatel má read-only přístup, jsou tlačítka pro přidávání, úpravu a mazání záznamů šedá a neaktivní.



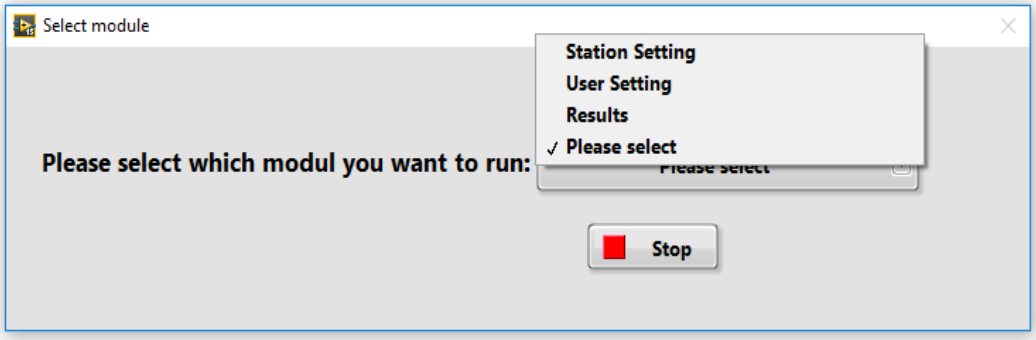
Obr. 5.7: Výběr modulů po přihlášení uživatele s oprávněním Developer.



Obr. 5.8: Výběr modulů pro přihlášení uživatele s oprávněním Operátor. Tento uživatel v ideálním případě nemá přístup k uživatelskému prostředí.



Obr. 5.9: Výběr modulů pro přihlášení uživatele s oprávněním Manager.



Obr. 5.10: Výběr modulů pro přihlášení uživatele s oprávněním Test Engineer.

5.3 Modul Customer Control

Zde je možno přidat novou firmu, nového zákazníka a nebo přiřadit zákazníkovi projekt a kontaktní osoby.

První pohled na tento modul vidíme na obrázku 5.11. Po otevření modulu je pokaždé otevřena první záložka a zobrazeny všechny záznamy kromě těch ve stavu „Obselete“. Toto je realizováno pomocí umělého vyvolání události kliknutí myši na tlačítko „List all entries“, proto je při pomalejším připojení viditelné zlaté podbarvení daného tlačítka.

V okamžiku, kdy uživatelské prostředí zpracovává konkrétní dotaz, který může kvůli rychlosti připojení k databázi trvat i několik sekund, je dané tlačítko podbarveno zlatě a pomocí grafiky vykresleno jako zmáčknuté. Uživatelské prostředí v tom okamžiku nereaguje na podměty ze strany uživatele. Toto řešení bylo zvoleno z toho důvodu, že nejpomalejší článek řetězce je komunikace s databází a není žádoucí, aby uživatel zmáčknul tlačítko na vylistování pouze aktivních záznamů a v mezičase začal upravovat položku na čtvrtém řádku, kdy není přesně určeno, jestli položka na čtvrtém řádku je stále ještě původní čtvrtý záznam v pořadí a nebo už nově vylistovaný čtvrtý aktivní záznam v pořadí, který zatím nebyl zobrazen v tabulce. Hrozí tak nebezpečí, že uživatel změní jiný záznam než původně chtěl.

Podle práv uživatele jsou aktivní všechny tlačítka a nebo pouze tlačítka pro zobrazení záznamů. K tomuto modulu má přístup plný přístup pouze Developer a read-only přístup uživatel s právy Analytic, který si může pouze zobrazit záznamy 5.12.

Z nabídky tlačítek vlevo lze pomocí prvního tlačítka vylistovat všechny záznamy, které nejsou ve stavu „Obselete“ a pomocí druhého tlačítka pouze aktivní záznamy. Po zmáčknutí třetího tlačítka automaticky vyskočí okno pro přidání nového zákazníka, jak lze vidět na obrázku 5.13.

V levé části okna je možno vybrat z rozbalovací nabídky jméno společnosti, které je zákazníkem firmy Kentigen a dále vyplnit město, kde se pobočka firmy nachází. V případě, že daná firma ještě není na seznamu zákazníků, lze ji přidat v pravé části okna. Výběr jména firmy z rozpalovací nabídky a uložení jména firmy v databázi v tabulce LOV - výčtu hodnot je kvůli možné změně názvu firmy v budoucnosti a snadné editaci tohoto názvu ve všech záznamech.

V celém uživatelském rozhraní jsou pole s červeným popisem povinná a pole s černým popisem dobrovolná. V případě, že záznam musí být unikátní, je označen hvězdičkou. Aktuální stav nedovoluje používat českou diakritiku. Jelikož nejčastěji je v zadání požadavek na anglický jazyk, nebyla podpora českých znaků implementována.

Po stisknutí tlačítka „Save“ je uložení záznamu potvrzeno vyskakovacím oknem

a poté je okno automaticky zavřeno. V tabulce obsahují záznamy dojde k opětovnému vylistování všech záznamů, které nejsou ve stavu „Obselete“ pomocí umělého vyvolání události stisku tlačítka „List all entries“.

V případě, že nastane chyba při zápisu do databáze je na to uživatel upozorněn. Pokud se jedná o chybné zadání dat, například, že nejsou vyplněna všechna požadovaná pole, je možné doplnit chybějící záznam a opět stisknout tlačítko uložit 5.15. V případě, že došlo ke ztrátě spojení s databází je na tuto skutečnost upozorněn uživatel pomocí automaticky vyskakovacího okna, které je třeba potvrdit stisknutím tlačítka „OK“ 5.15. Poté se okno pro přidání zákazníka automaticky zavře a v případě, že se nepodařilo opět navázat spojení s databází je uživatel upozorněn, že nastala interní chyba aplikace a modul bude uzavřen. Toto nastane i v případě, když dojde k neočekávané, neošetřené chybě v aplikaci 5.16. Aplikace komunikuje přímo s databází a tedy žádná data nejsou uložena lokálně, proto při ztrátě spojení s databází jsou všechny do té doby potvrzené záznamy uloženy na serveru.

Dále je možno záznam upravit. To se provádí zmáčknutím tlačítka „Edit Entry“. Po zmáčknutí tohoto tlačítko proběhne kontrola, jestli byl vybrán řádek označující záznam. Řádek tabulky se vybere tak, že se na něj klikne dvakrát myší, označený řádek se podbarví zlatou barvou. V případě, že nebyl vybrán žádný záznam a nebo byl vybrán řádek tabulky, který je prázdný, je uživatel vyzván, aby vybrat řádek, který obsahuje platná data 5.17.

Obrázek 5.18 ukazuje okno pro upravení detailu záznamu zákazníka. V levé části jsou vidět aktuální hodnoty a v pravé části okna je možno vyplnit hodnoty nové. Není nutno vyplnit všechna pole, což také indikuje černá barva textu popisků. Při editaci záznamu se jako první jeho stav nastaven na „Modified“, tedy, že záznam byl modifikován. Pokud záznam není možno nastavit do stavu „Modified“, je na to uživatel upozorněn.

Poslední tlačítko umožňuje smazat záznam. Není ovšem možné smazat záznam, na kterém jsou závislé další záznamy. Pokud vybraný záznam obsahuje nějaké závislé záznamy, ke kterým by byl poté ztracen přístup, je uživatel upozorněn, že z výše uvedeného důvodu není možné záznam smazat. Vše probíhá následovně: po stisknutí tlačítka „Delete“ je uživatel vyzván k potvrzení, jestli daný záznam chce opravdu smazat 5.19. Poté je zkontrolováno, zda pro zvolený záznam existují nějaké závislé záznamy a pokud ne, je možno záznam smazat 5.20.

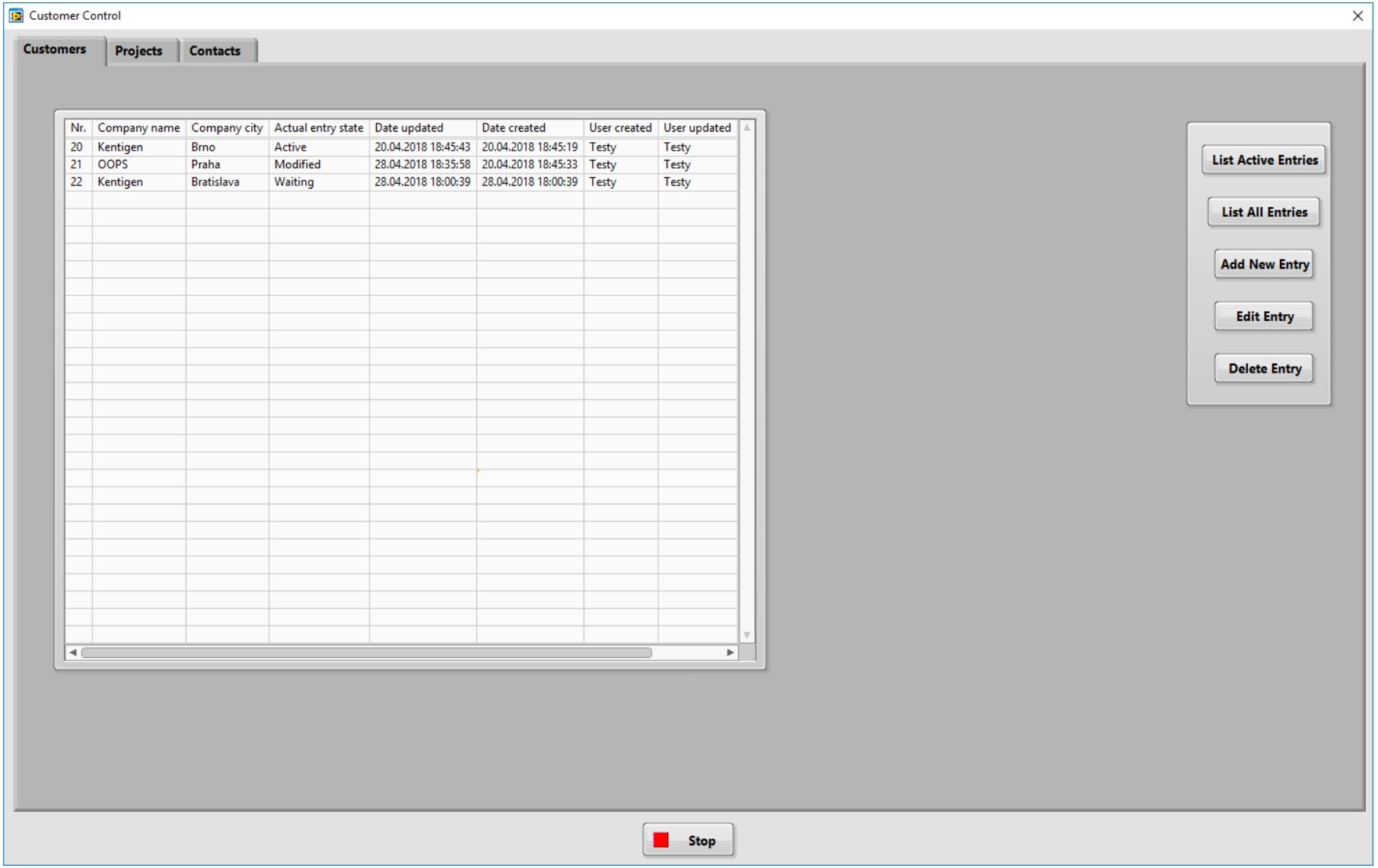
Pokud je potřeba, aby existující záznam nebyl déle viditelný v aplikaci, je možno ho nastavit do stavu „Obselete“. Poté již není zobrazován v aplikaci, ale stále je uložen v databázi na serveru. Poté, co je záznam nastaven do stavu „Obselete“, není možné již v aplikaci vylistovat jeho závislé záznamy.

Pokud je označen řádek, jehož závislé záznamy chci rozbalit, stačí kliknout na další záložku. Ta je poté zobrazena a opět jsou vylistovány všechny záznamy, kromě

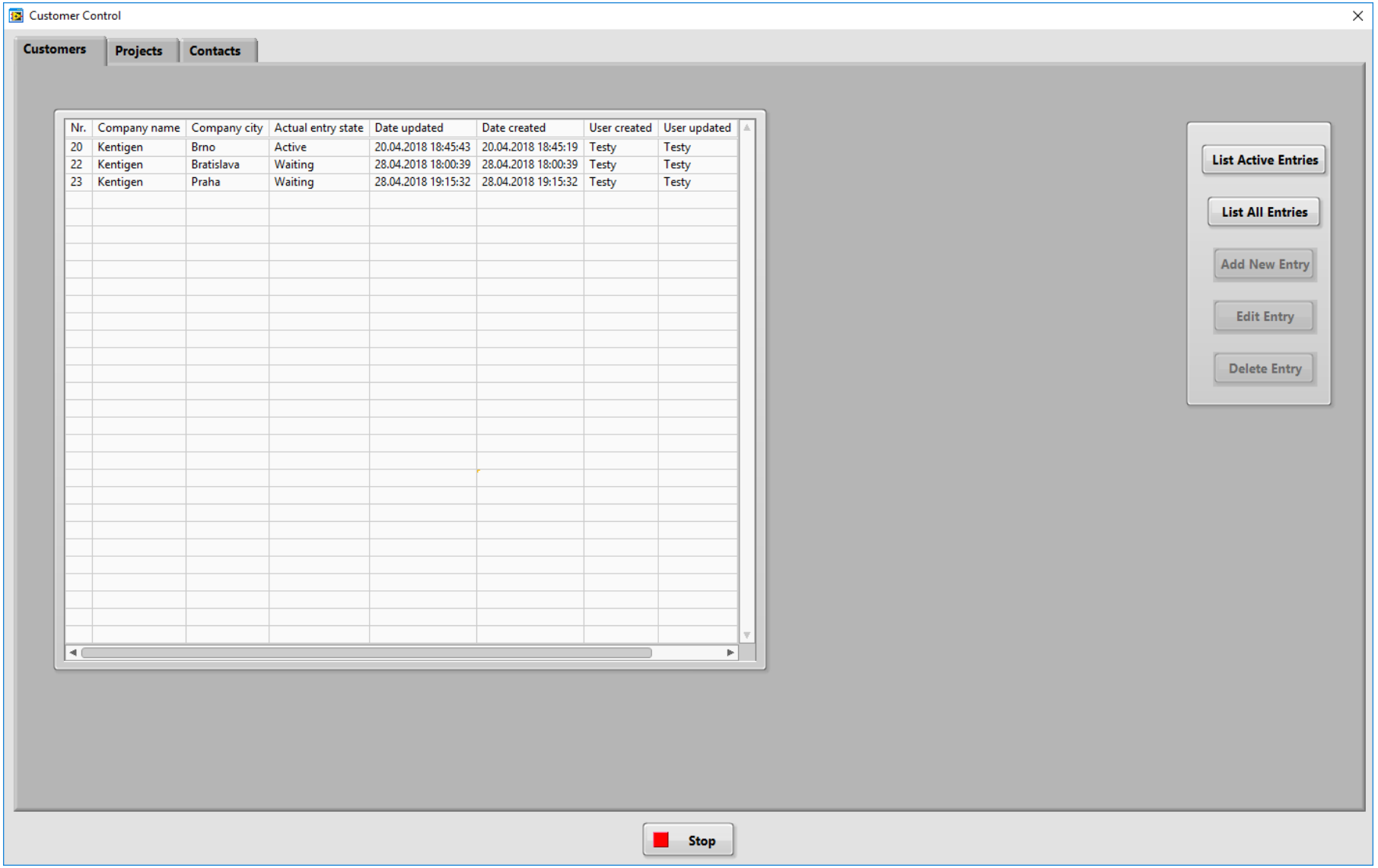
těch ve stavu „Obselete“. Pro lepší orientaci slouží navigační řádek „Actual position“. Pokud není vybrán žádný záznam jehož podzáznamy chceme zobrazit, je na to uživatel upozorněn a výsledná tabulka je prázdná.

Jak můžeme vidět na obrázku 5.22, který ukazuje okno pro přidání nového projektu je pole „Project code“ nejenom doprovázeno červeným popiskem, který znamená, že je to povinná položka, ale také hvězdičkou, která znamená, že kód projektu musí být v databázi unikátní. Pokud je zadán již existující kód projektu, skončí pokus o zápis do databáze chybou a uživatel je o ní informován pomocí samo vyskakujícího okna.

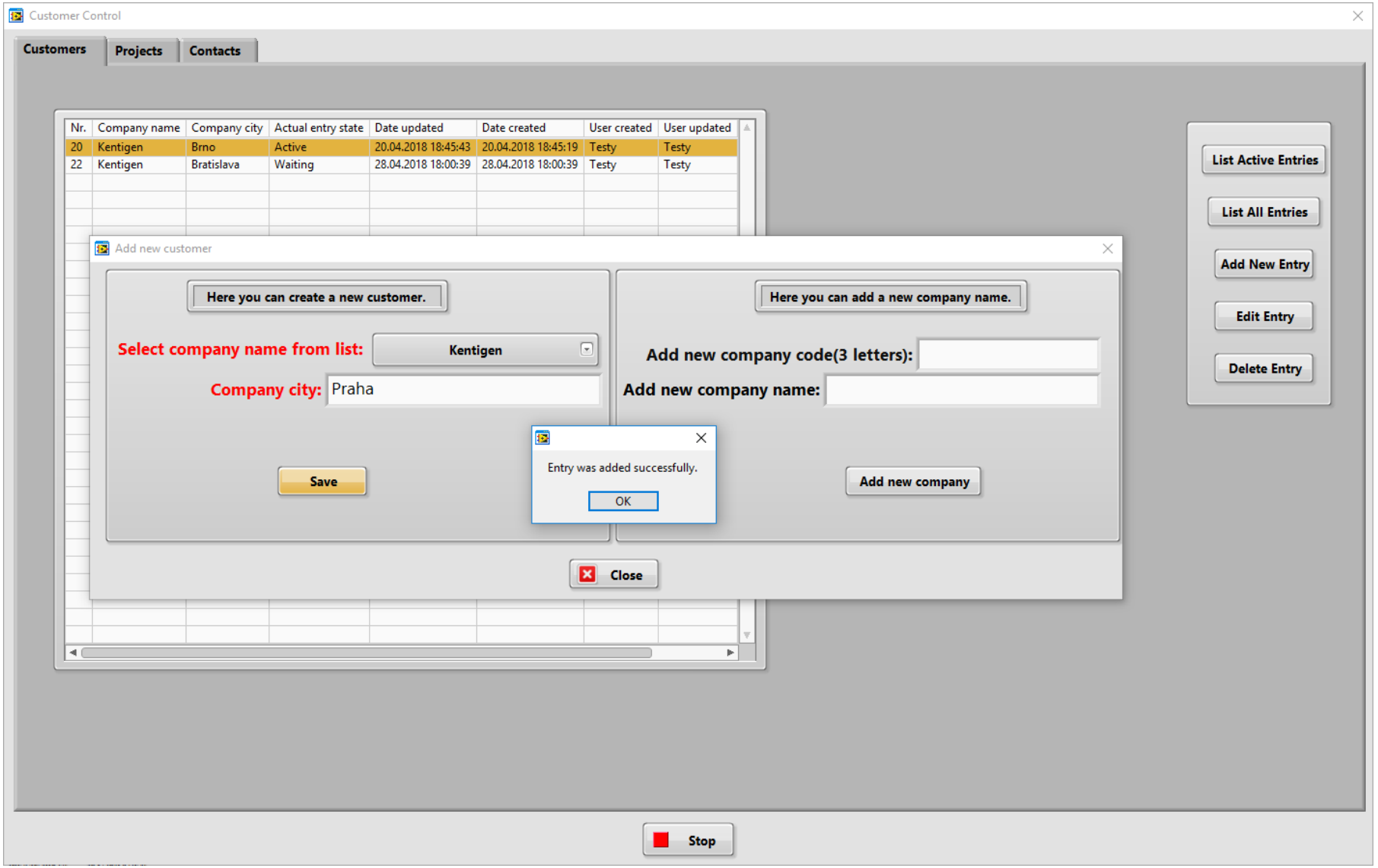
Následující obrázky zobrazují okna pro editaci projektu a dále pohled na záložku „Contacts“ společně s okny pro úpravu jejich záznamů. Jak si můžeme povšimnout na obrázku 5.26 při zadávání telefonního čísla při tvorbě nového kontaktu je možno vložit pouze čísla. Další kontrola na platnost formátu telefonního čísla neprobíhá. Na obrázku si lze také povšimnout, že název společnosti se zadává jako text a ne výběrem z rozbalovací nabídky zákazníků. Zde se budou vyplňovat kontaktní osoby pro konkrétní zákazníky, jejich dodavatelé, subdodavatelé a servisní pracovních a není tudíž žádoucí, aby se objevovali v celkovém výčtu zákazníků firmy Kentigen. Zároveň je ošetřeno, že nelze omylem vytvořit projekt a poté testovací stanici pro úklidovou firmu.



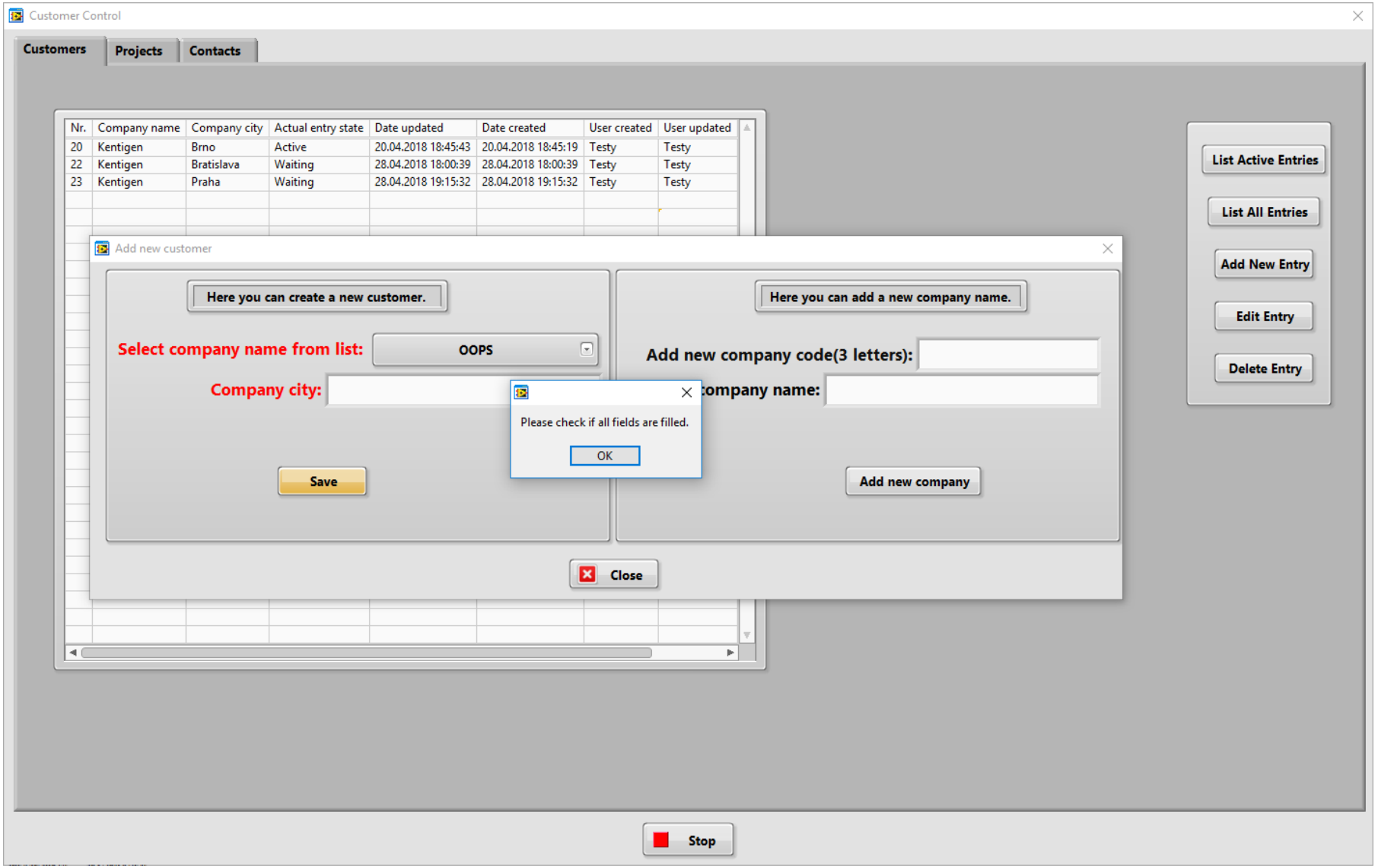
Obr. 5.11: Modul pro správu zákazníku bezprostředně po spuštění pro uživatele s oprávněním Developer.



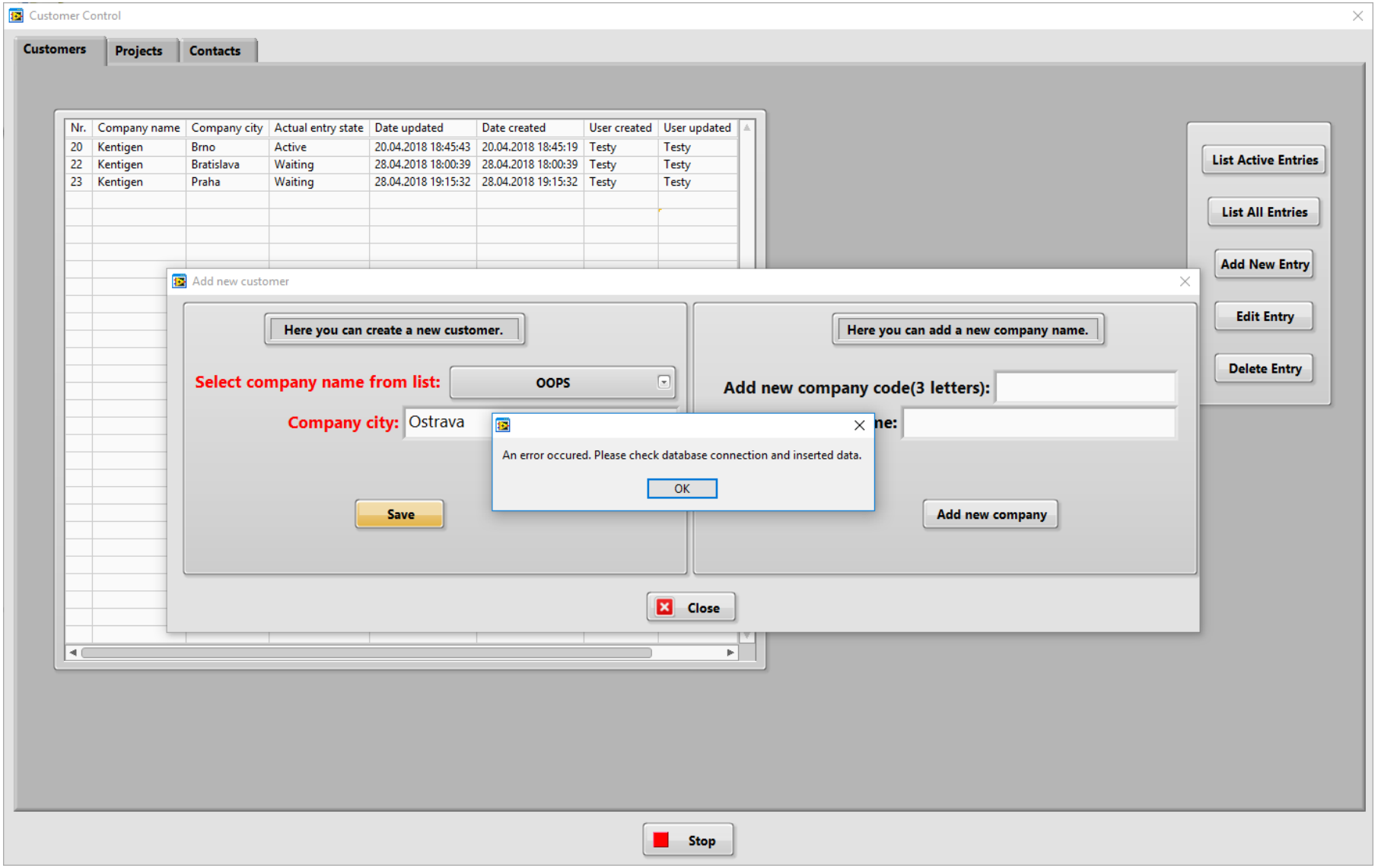
Obr. 5.12: Modul customer po spuštění uživatelem s právy Analytic.



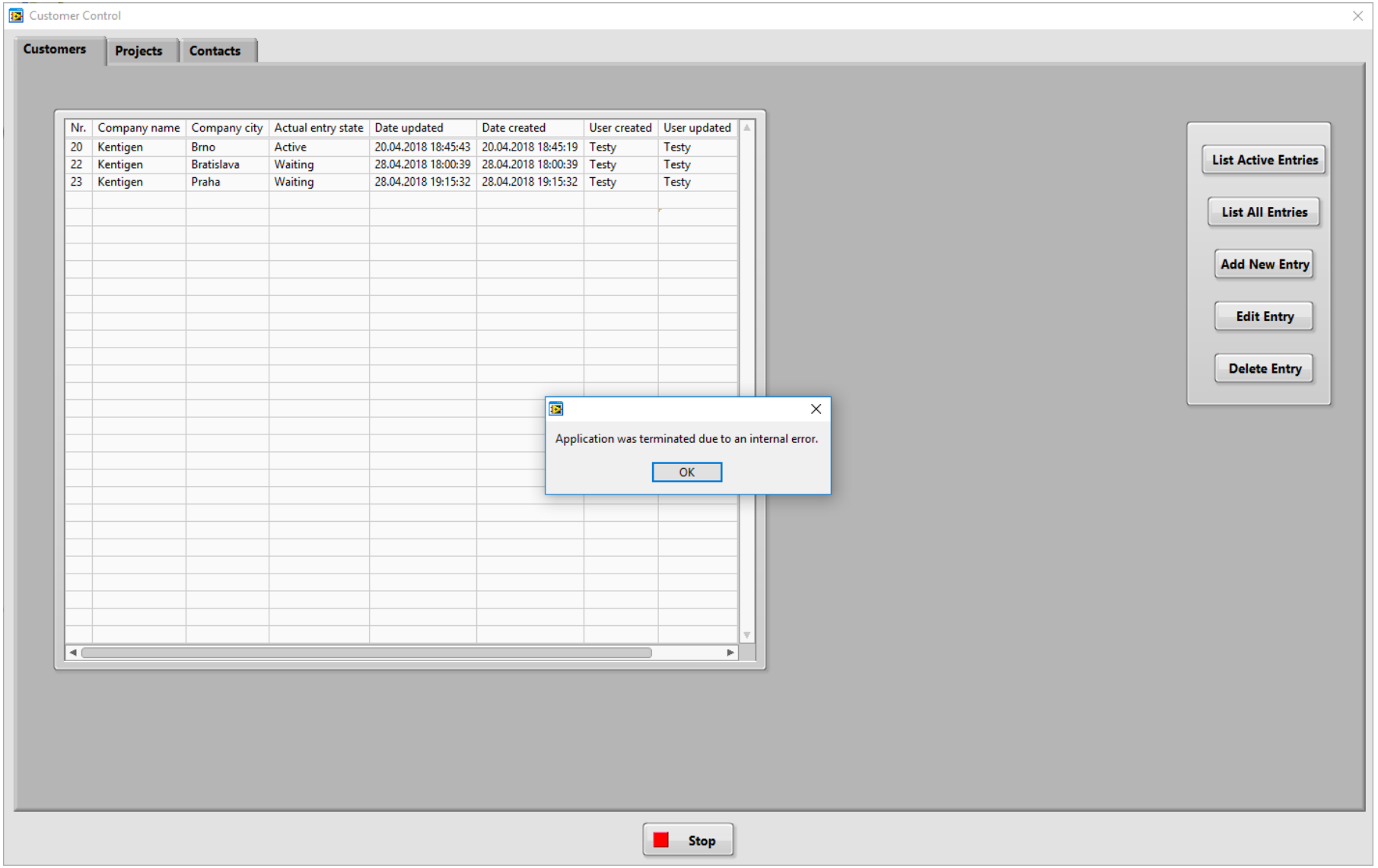
Obr. 5.13: Potvrzení úspěšného přidání nového záznamu.



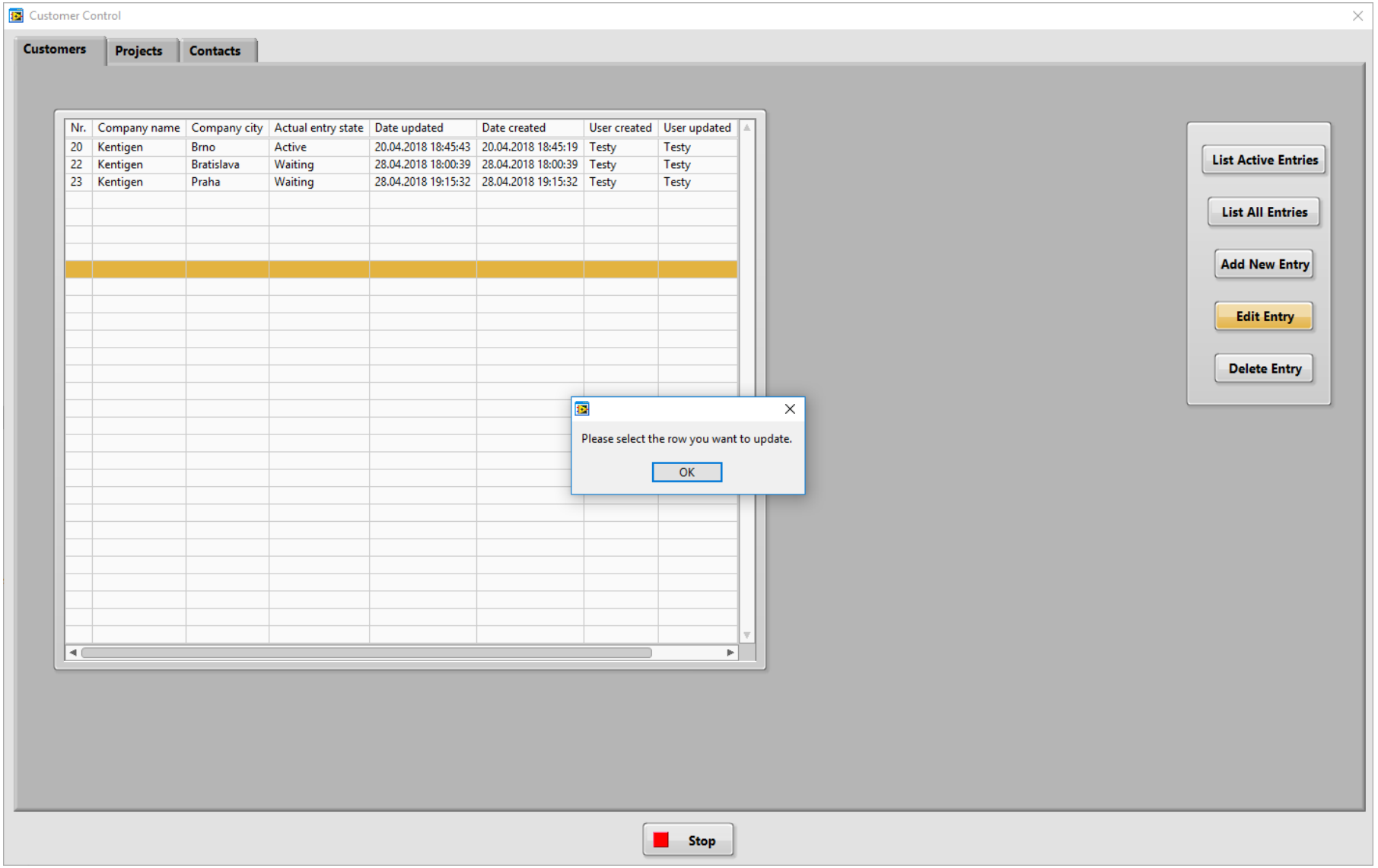
Obr. 5.14: Upozornění na chybu při vkládání záznamu - nejsou vyplněna všechna požadovaná pole.



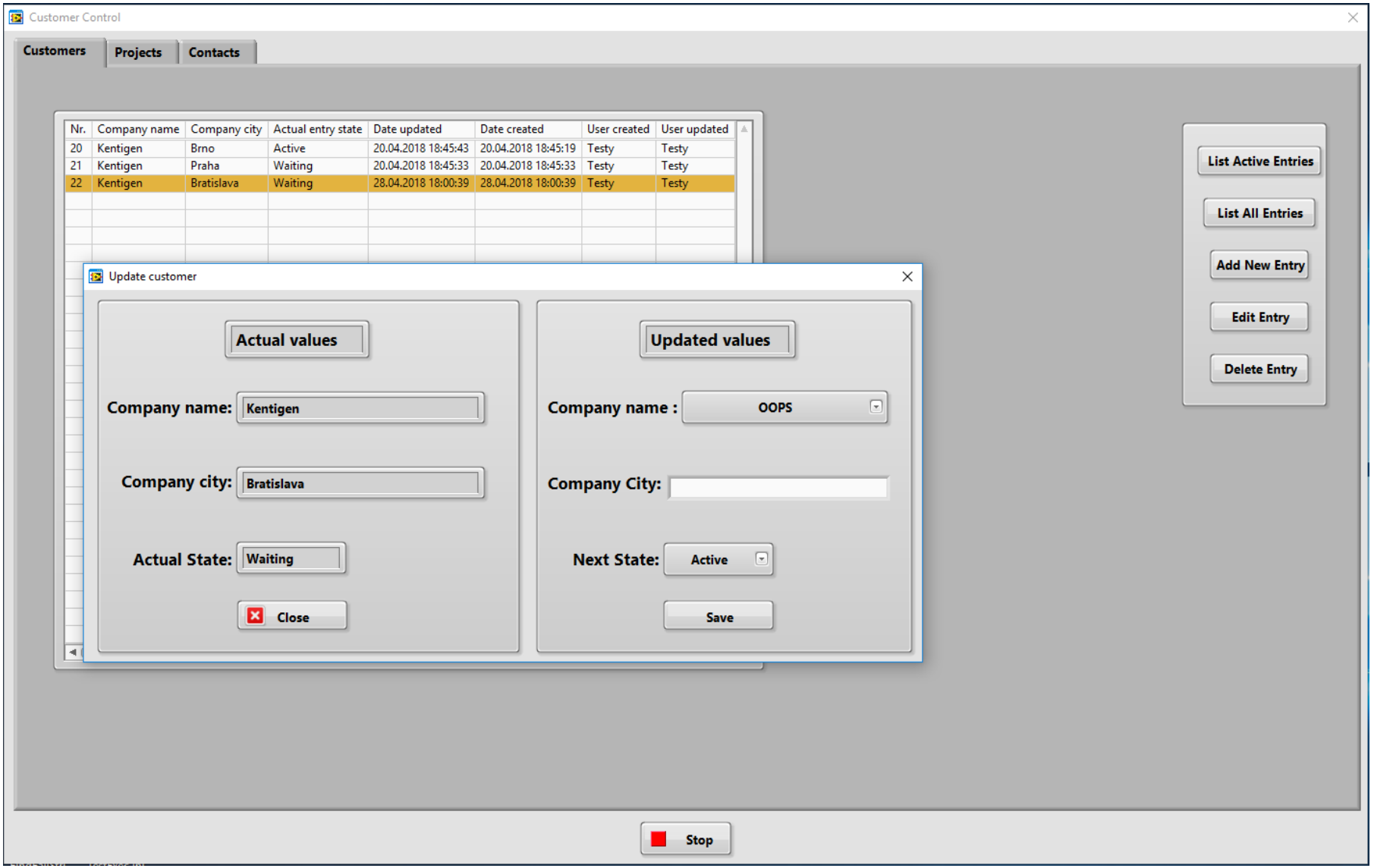
Obr. 5.15: Chyba při zápisu nového záznamu způsobená na straně databáze: buď bylo ztraceno spojení s databází a nebo je zadaný řetězec příliš dlouhý.



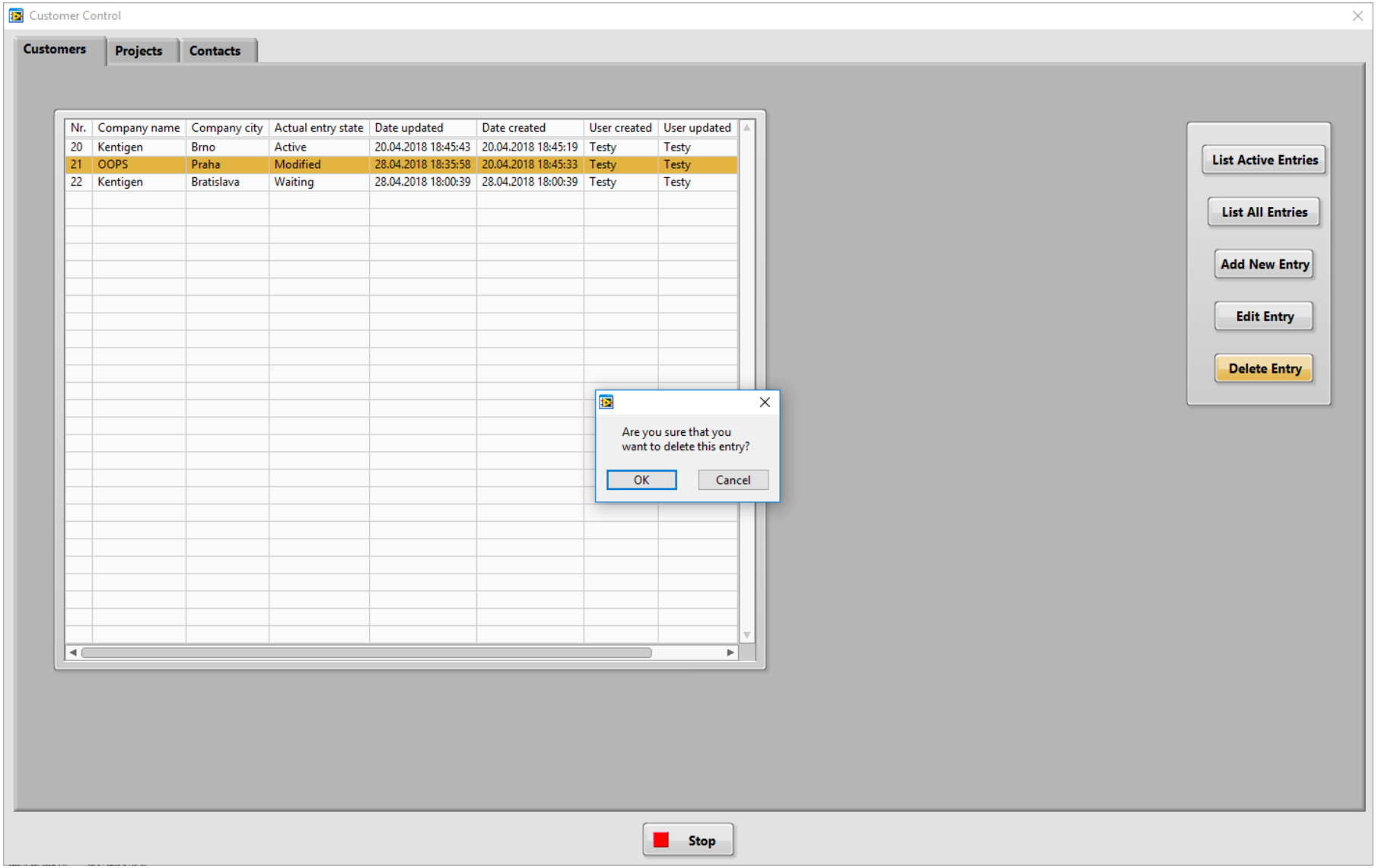
Obr. 5.16: Upozornění na interní chybu aplikace před uzavřením modulu.



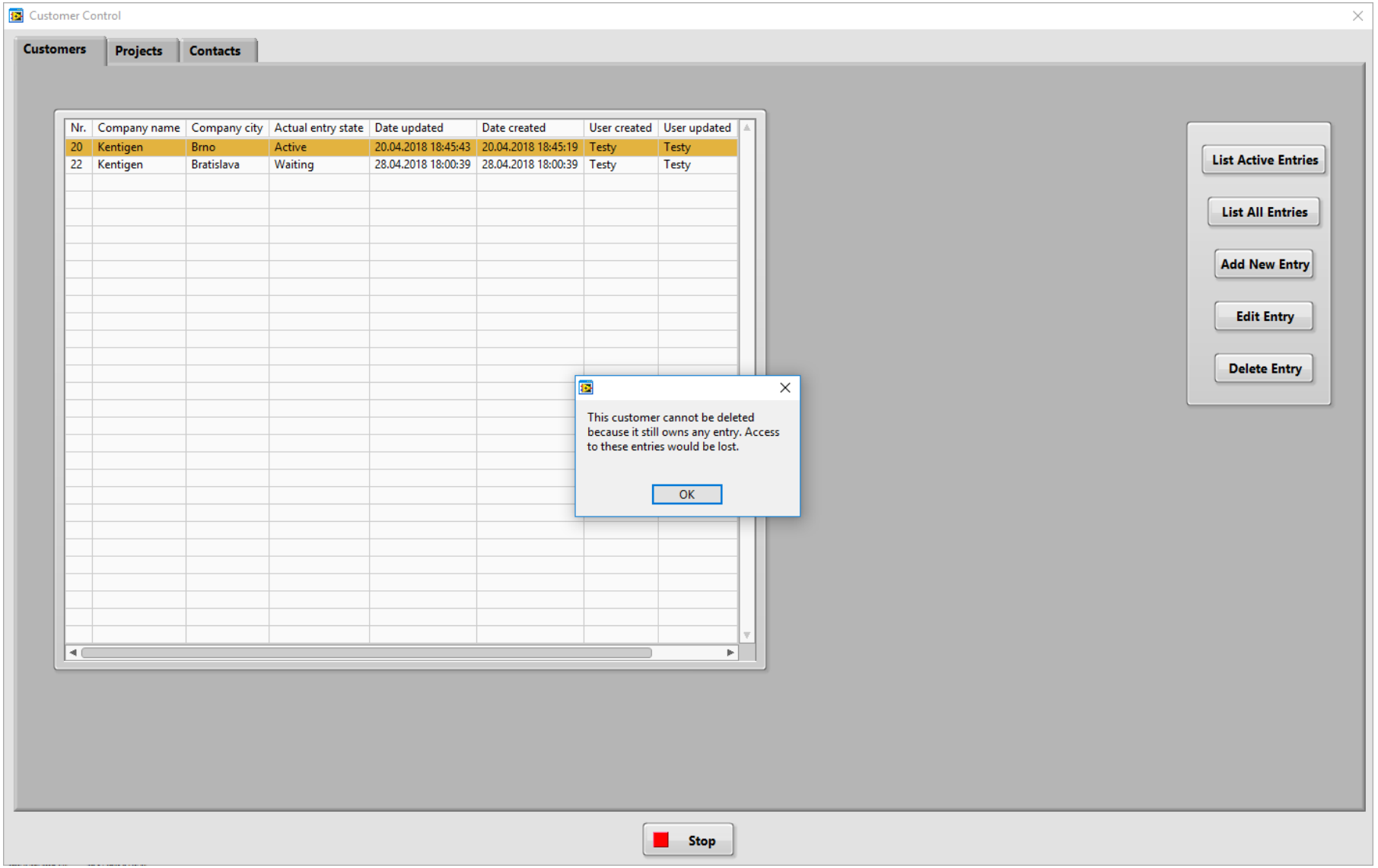
Obr. 5.17: Upozornění uživatele v případě, že chce upravit řádek, který neobsahuje záznam.



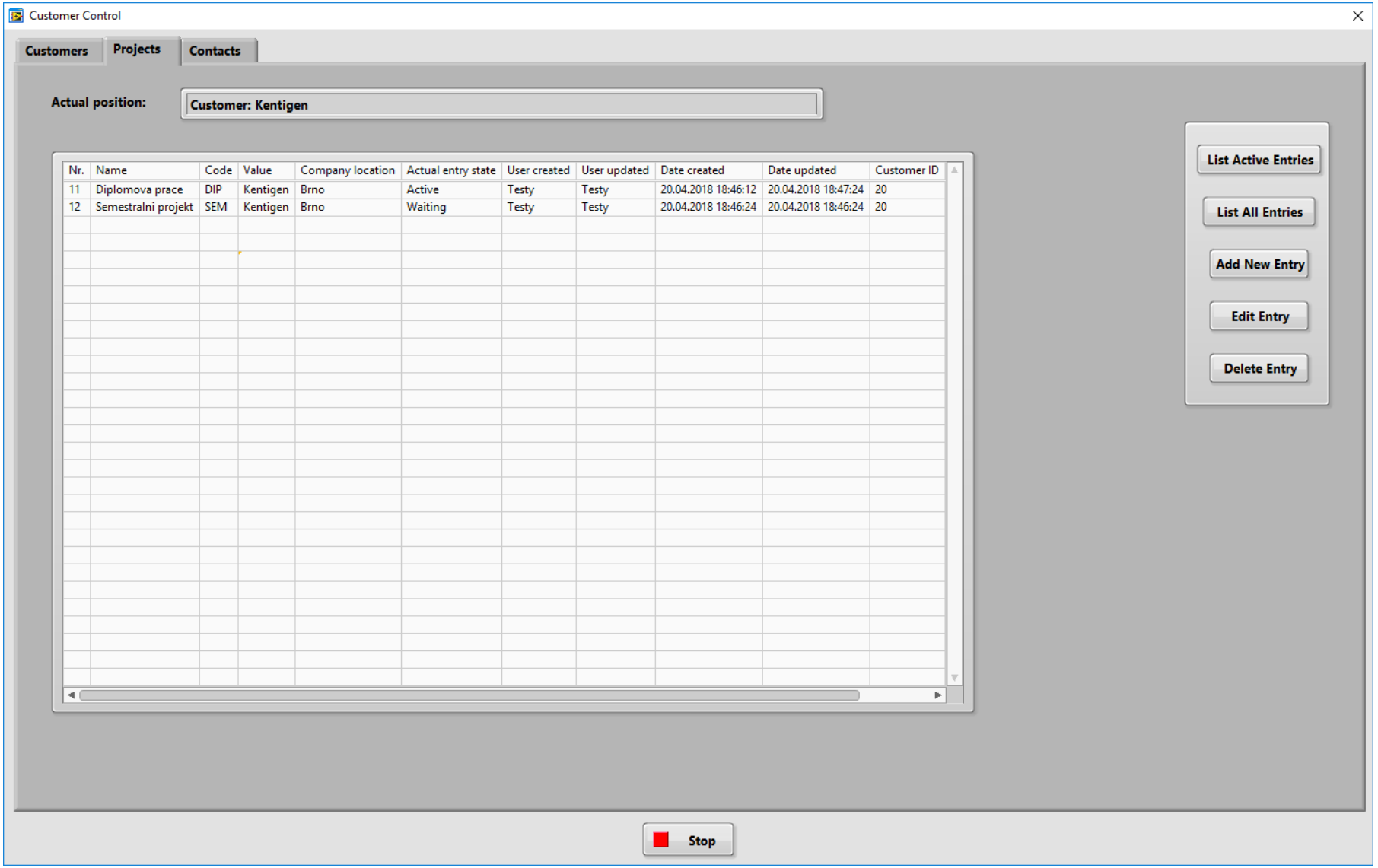
Obr. 5.18: Okno pro editaci záznamu.



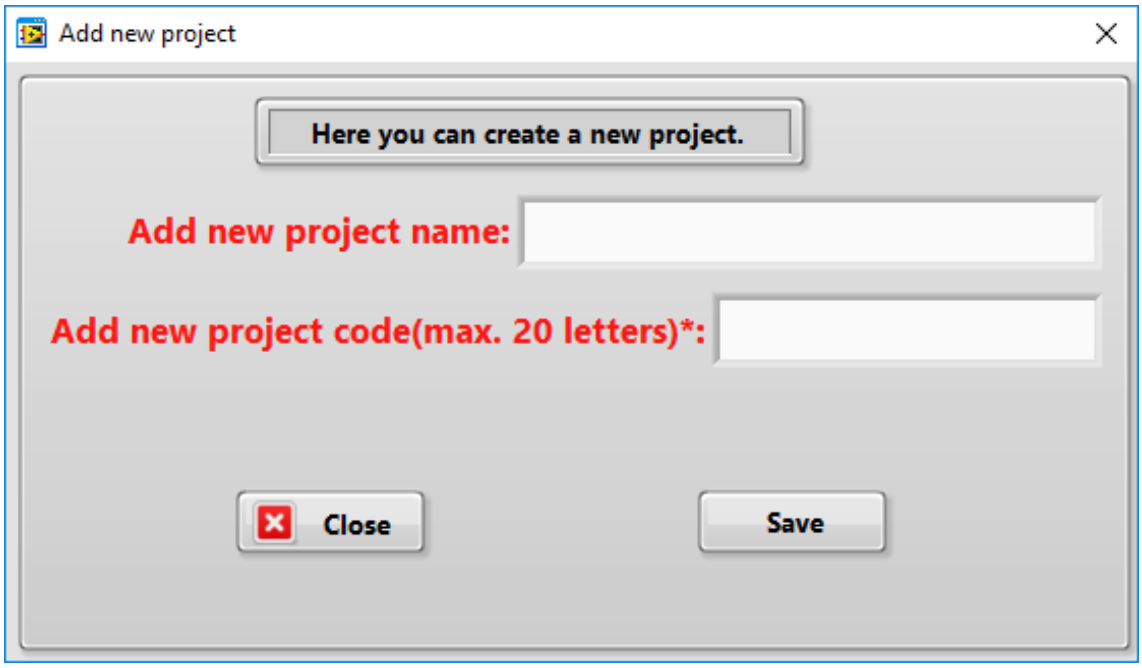
Obr. 5.19: Výzva uživatele k potvrzení, jestli chce daný záznam opravdu smazat.



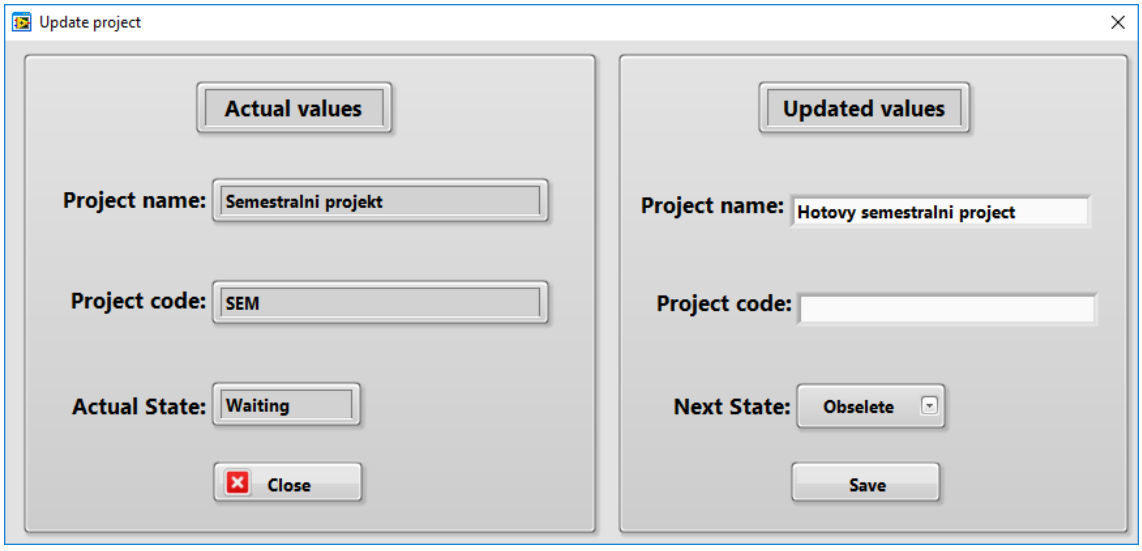
Obr. 5.20: Upozornění uživatele, že není možno smazat vybraný záznam.



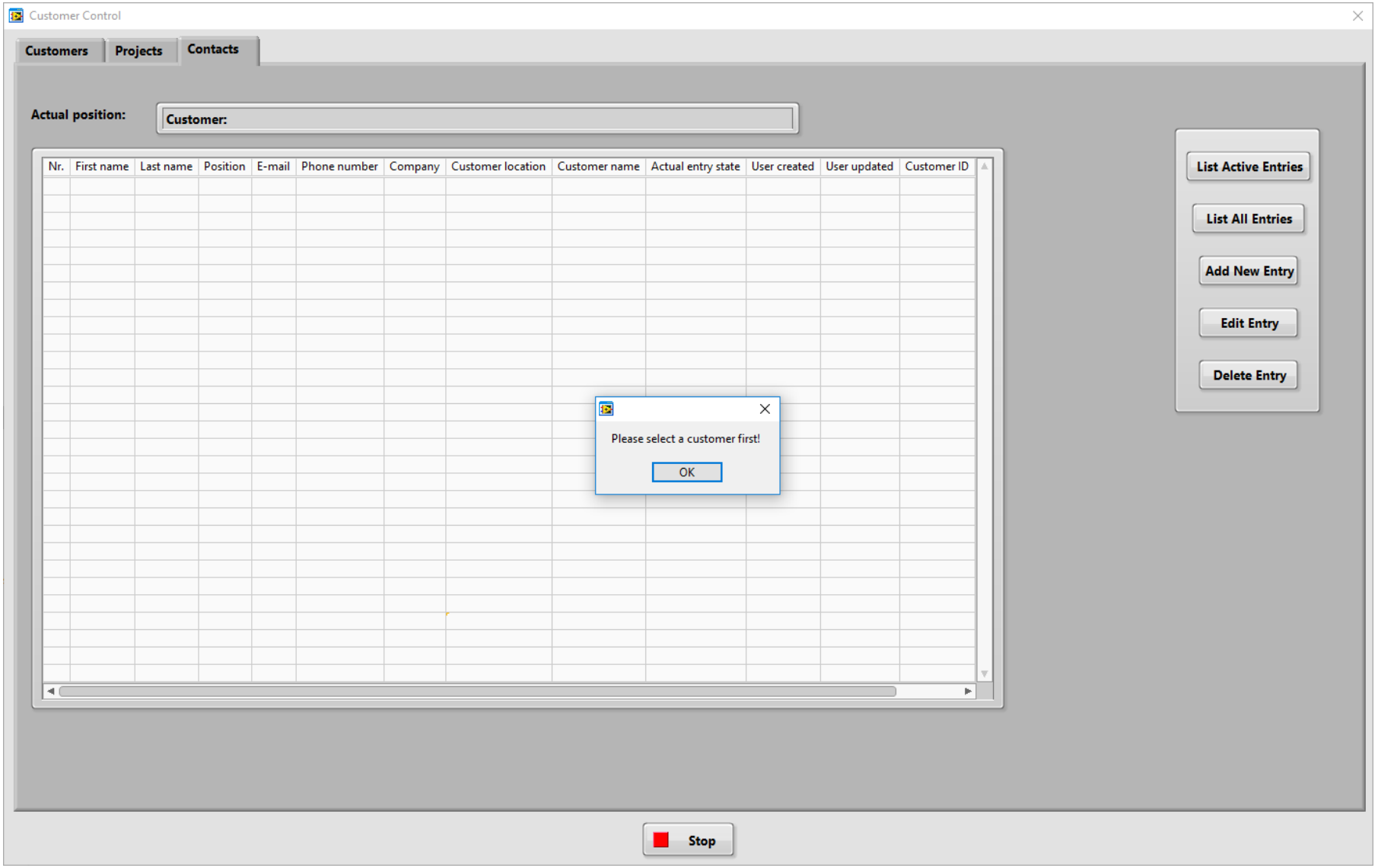
Obr. 5.21: Pohled na záložku „Projects“.



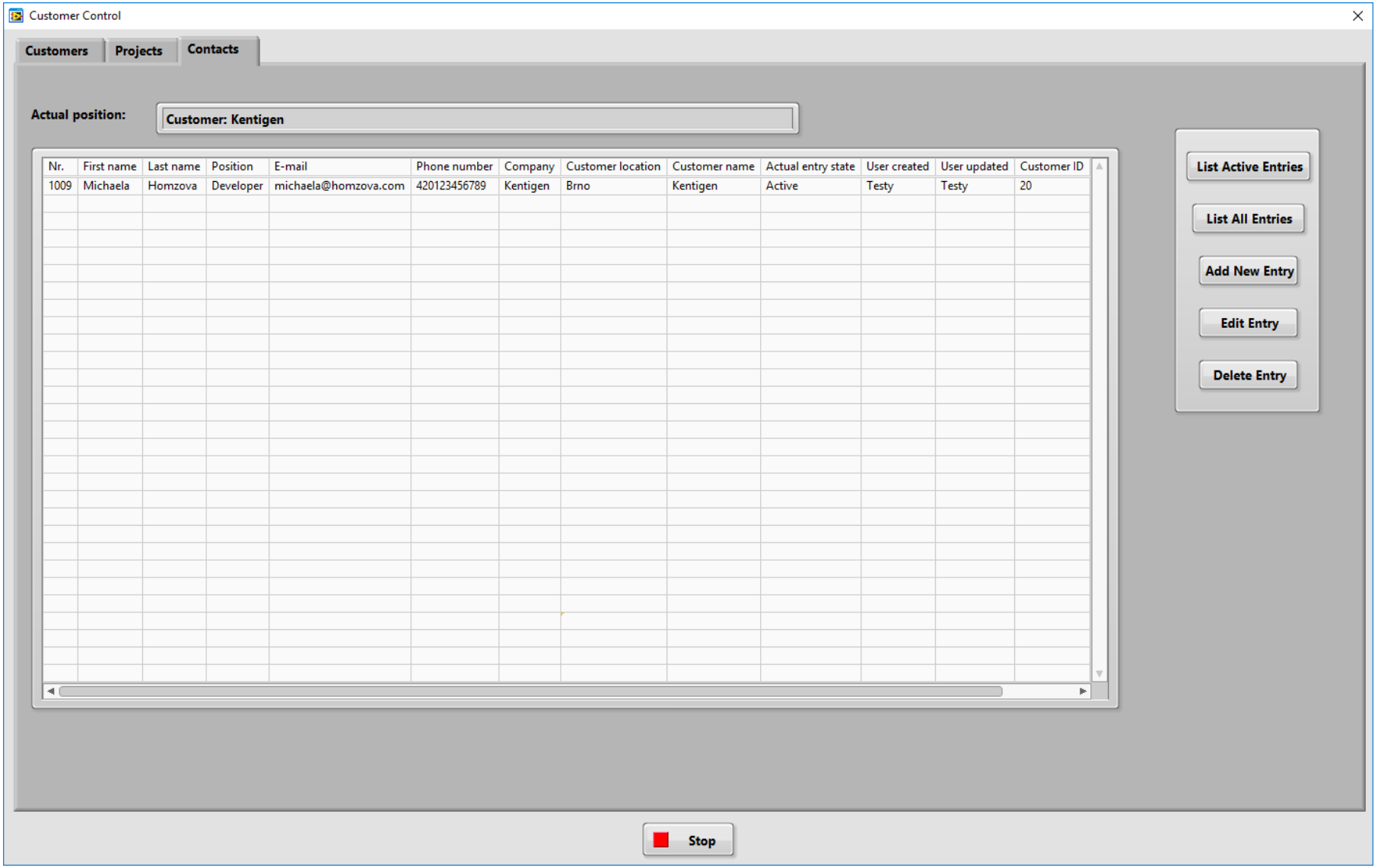
Obr. 5.22: Okno pro přidání nového projektu.



Obr. 5.23: Okno umožňující editaci stávajícího projektu.

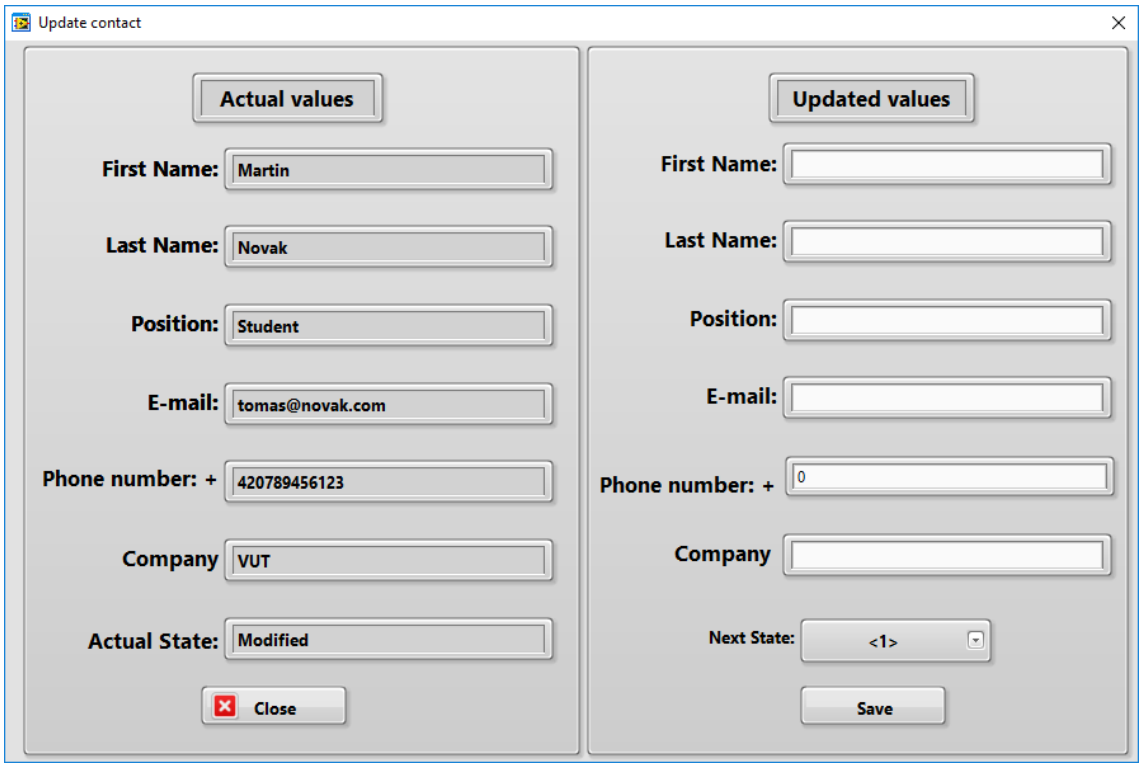


Obr. 5.24: Výzva uživateli, aby vybral zákazníka, jehož kontakty chce zobrazit.



Obr. 5.25: Pohled na záložku uvContacts, kde jsou zobrazeny pouze aktivní záznamy.

Obr. 5.26: Detail okna pro přidání nového kontaktu.



Obr. 5.27: Detail okna pro úpravu stávajícího kontaktu.

5.4 Modul Station Setting

Plný přístup k tomuto modulu má Developer a také Test Engineer. Dále pouze read-only přístup má uživatel s právy Analytic. Zde se nastavují předpisy pro danou testovací stanici a proto první záložka je seznam testovacích stanic 5.29.

Aby mohl být záznam v tabulce Station, a s tím celý předpis této stanice použit pro testování, musí být stanice ve stavu „Active“. Jak je patrno z přechodového diagramu záznamů, které tvoří metadata zde 3.4, při vytvoření v uživatelské aplikaci je záznam nastaven do stavu „Created“. Po zapsání na server je převeden do stavu „Waiting“. Poté je nutno manuálně přepnout záznam do stavu „Active“. Toto opatření je nezbytné, protože pouze jedna stanice s daným názvem může být aktivní. Jak bude dále popsáno v testovací aplikaci, při inicializaci testovací stanice jsou použity pouze aktivní záznamy a je kontrolován jejich počet. Záznamy v modulu „Station Setting“ také nemohou být editovány kromě změny jejich stavu reffig:updateentrywaiting.

Kvůli ochraně dat v testovacích předpisech je možno po přidání záznamu pouze změnit jeho stav. V případě, že záznamu ještě nebyli přiřazeny žádné podřízené záznamy, je ho možno smazat. V případě, že záznam vlastní podřízené záznamy, je možno ho nastavit do stavu „Obselete“ a nebude se již v aplikaci zobrazovat. Před tím je třeba zvážit, jestli nebude v budoucnu potřeba přistupovat k jeho podřízeným záznamům a to receptu stanice, jejím testovacím krokům a limitům. Implementaci přechodového stavu záznamů tvořících metadata ilustrují obrázky 5.30 a 5.31.

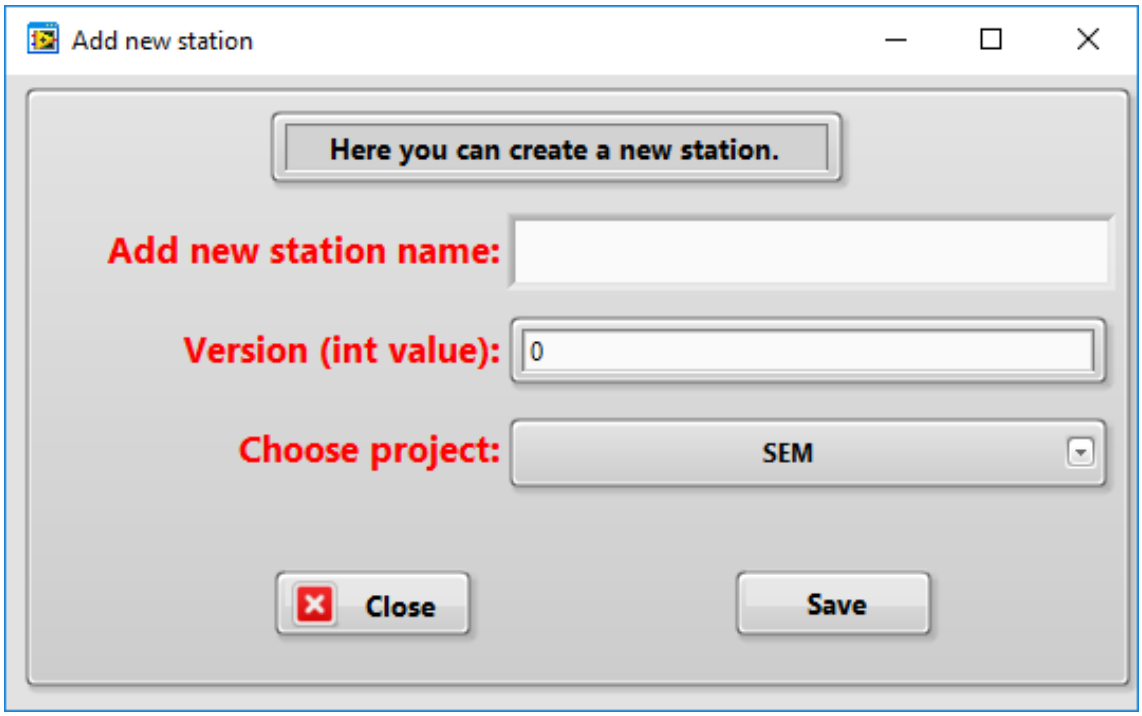
Podle předpisu uvedeného v tabulce 2.2 si lze povšimnout, že limity ke konkrétním krokům nemají jednotný datový typ. Proto se při zadávání nového kroku vybírá datový typ z rozbalovaní nabídky, jak ukazuje obrázek 5.34. Všechny hodnoty limitů se poté ukládají jako plovoucí datový typ. Ten má v případě použitého MS SQL serveru velikost 32-bitů, která se při ukládání automaticky rozšíří na velikost 64-bitů, je-li to potřeba [17].

Informace z nabídky datových typů poté slouží jako reference pro případnou další práci s daty. Dále limity, které jsou zadané v hexadecimální soustavě jsou převedeny na celé číslo a to je poté uloženo ve float formátu. Celkový rozsah float formátu nám umožňuje uložit i reprezentace velkých čísel. Toto řešení bylo zvoleno z následujících důvodů: cílem bylo zachovat číselný formát se kterým se dá relativně jednoduše pracovat v rámci dalšího zpracování a vyhodnocování výsledků pomocí rutin MS SQL serveru, kdyby to bylo v budoucnu potřeba. Druhý důvod je úspora místa při ukládání dat. V případě velkých čísel či čísel s velkým počtem desetinných míst by při ukládání naměřených hodnot do proměnné typu string došlo ke zbytečnému obsazení paměti. Při uložení čísla s pěti digity, tedy pět číslic - znaků, který by zabrali celkově 5 bytů, šetříme místo, protože ve float reprezentaci zaberou pouze 4

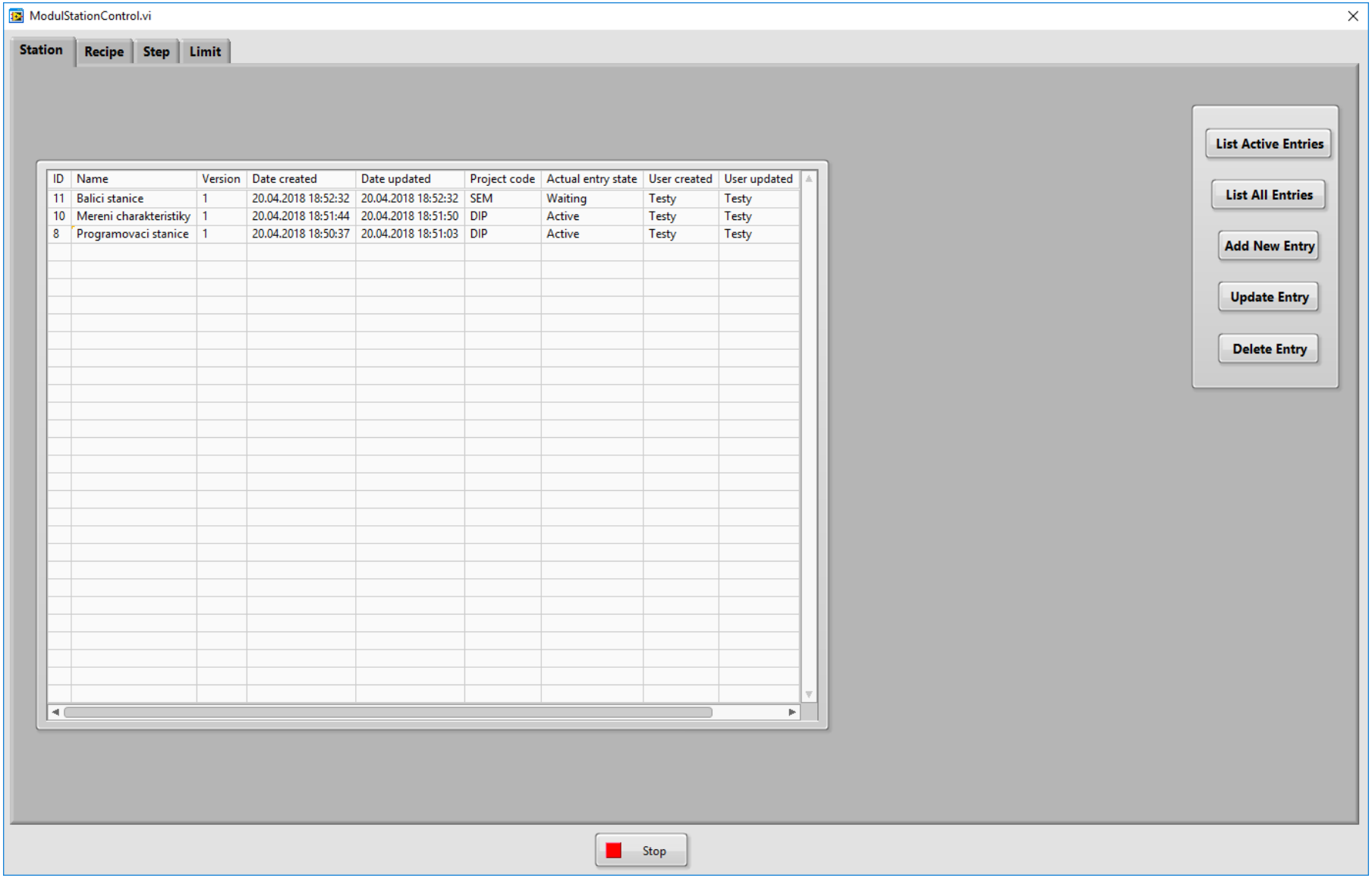
byty.

Pokud chceme hodnoty limitů zobrazit v hexadecimální soustavě je potřeba při zadávání limitů za pole jednotka vyplnit řetězec „HEX“, jak ukazuje obrázek 5.37.

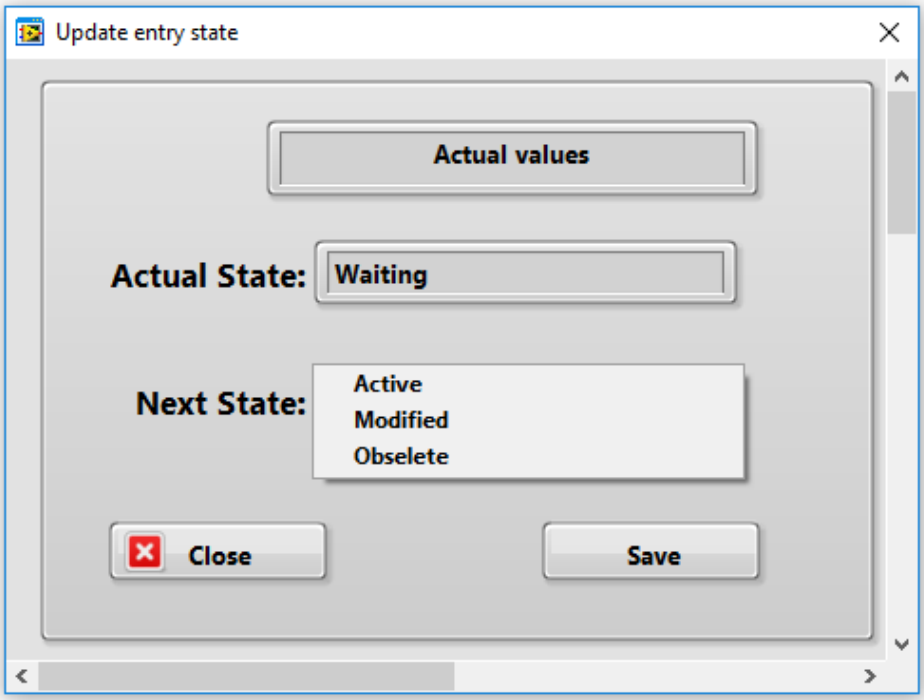
Jak si lze povšimnout při bližším zkoumání předchozích modulů, tabulka pro listování záznamů nemá pevnou velikost, ale její šířka i šířka jednotlivých sloupců se přizpůsobuje obsahu. V případě, že je počet sloupců tak velký, že se nevejdou všechny do nastaveného maximálního viditelného rozměru tabulky, objeví se automaticky posuvníky, které umožňují zobrazit další sloupce **??**. To samé platí při větším než maximálním zobrazitelném počtu řádků tabulky. VI pro automatické nastavení šířky sloupce bylo převzato z následujícího odkazu z diskuzního fóra pro podporu produktů firmy National Instruments [18] a poté upraveno autorem této práce.



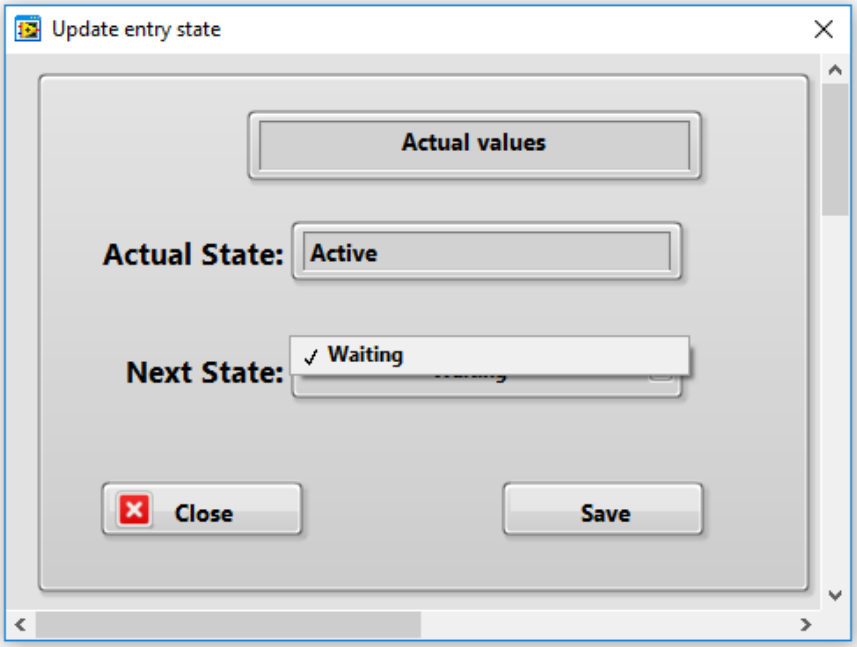
Obr. 5.28: Okno pro přidání nové stanice.



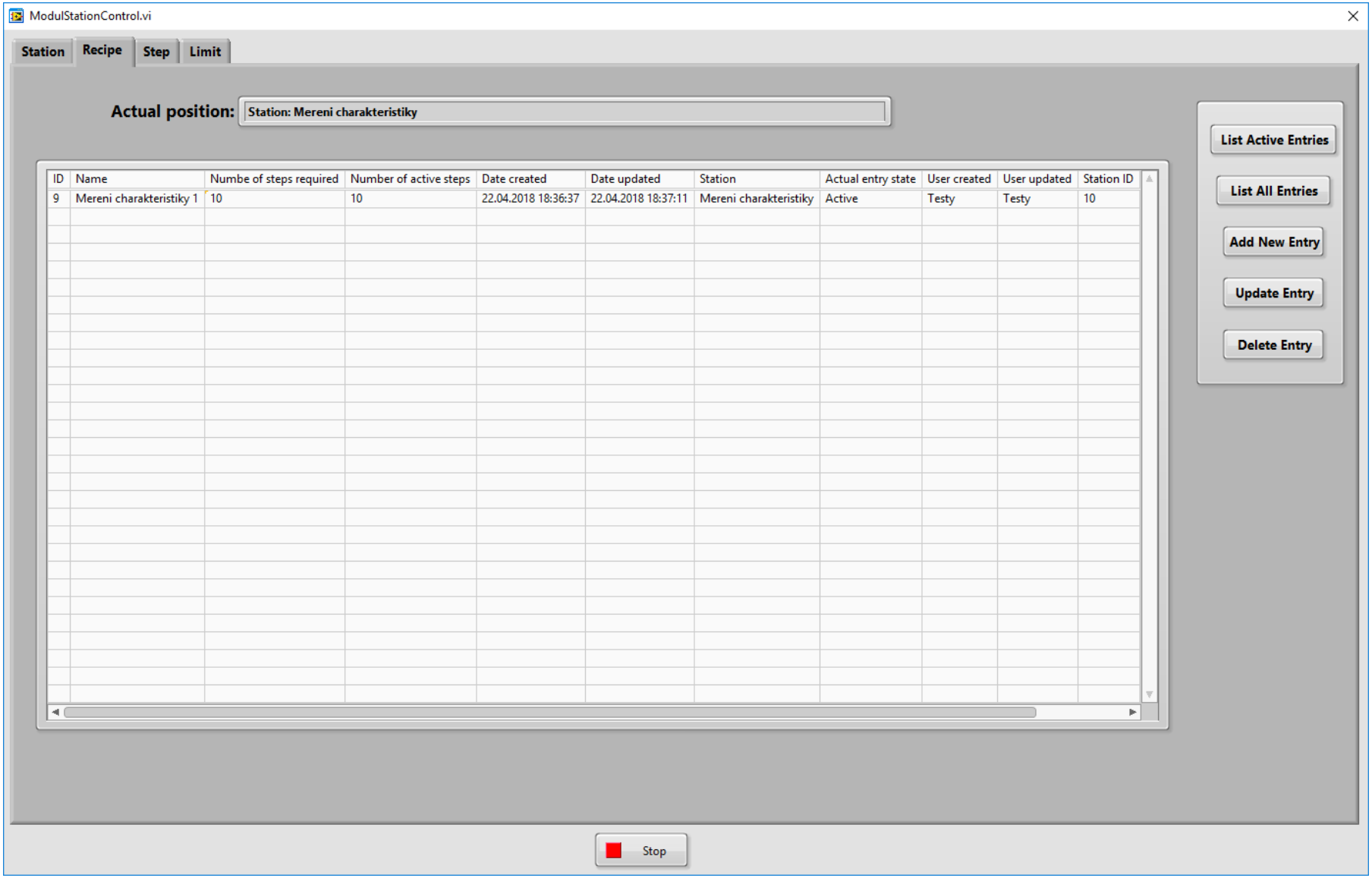
Obr. 5.29: Pohled na modul „Station Setting“ po spuštění.



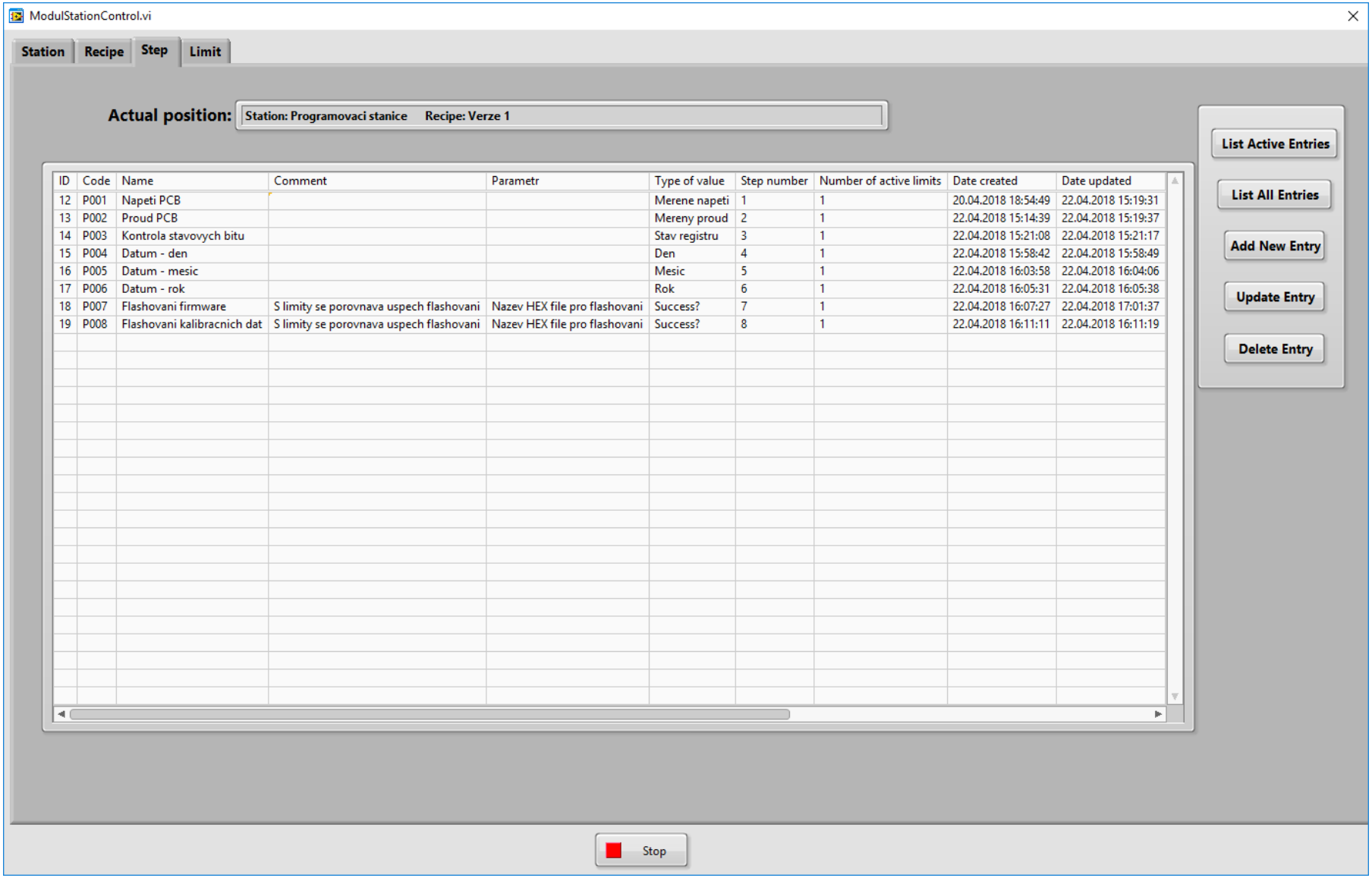
Obr. 5.30: Okno pro změnu stavu záznamu z výchozího stavu „Waiting“.



Obr. 5.31: Okno pro změnu stavu záznamu z výchozího stavu „Active“.

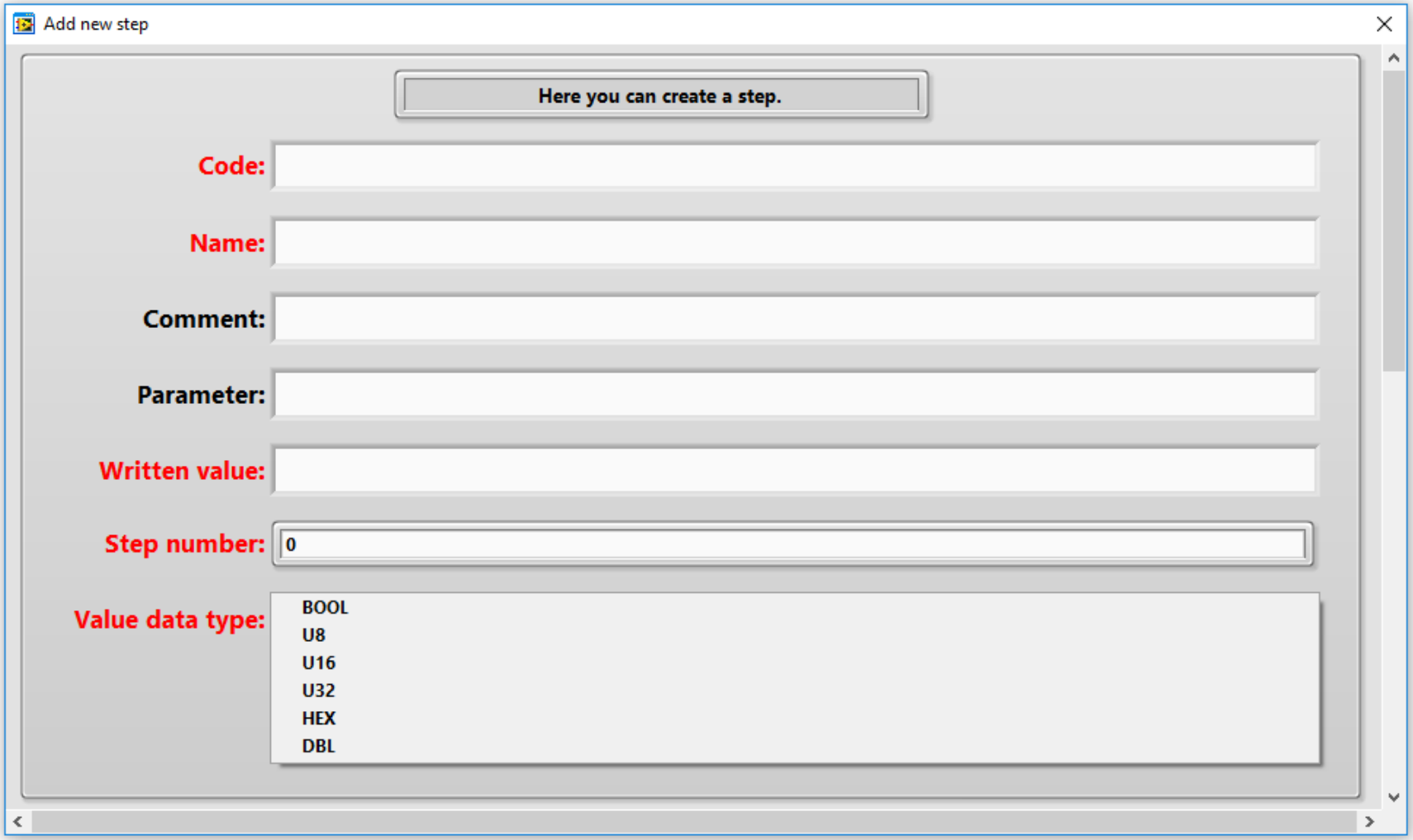


Obr. 5.32: Detail záložky „Recipe“.

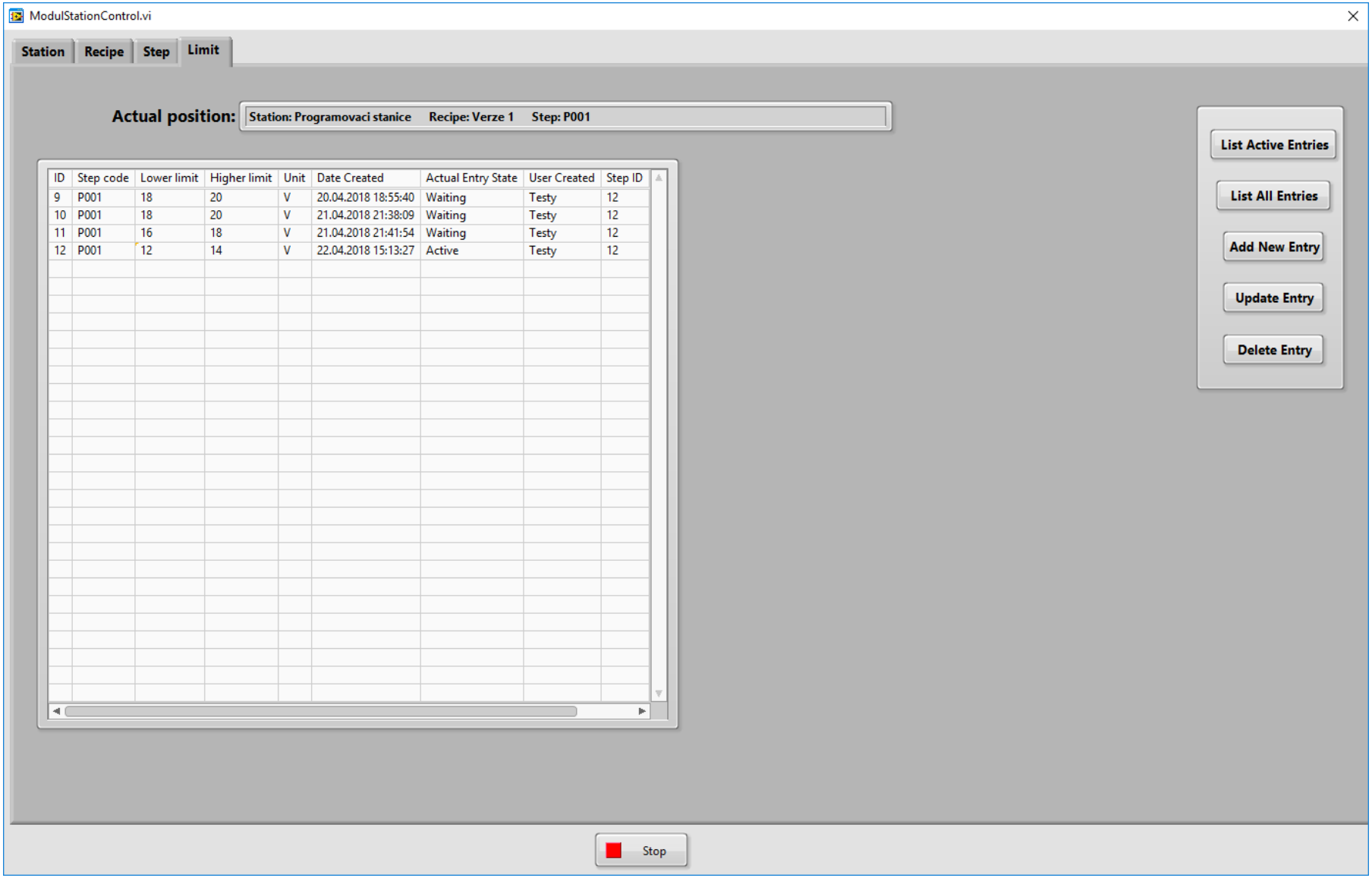


ID	Code	Name	Comment	Parametr	Type of value	Step number	Number of active limits	Date created	Date updated
12	P001	Napeti PCB			Merene napeti	1	1	20.04.2018 18:54:49	22.04.2018 15:19:31
13	P002	Proud PCB			Mereny proud	2	1	22.04.2018 15:14:39	22.04.2018 15:19:37
14	P003	Kontrola stavovych bitu			Stav registru	3	1	22.04.2018 15:21:08	22.04.2018 15:21:17
15	P004	Datum - den			Den	4	1	22.04.2018 15:58:42	22.04.2018 15:58:49
16	P005	Datum - mesic			Mesic	5	1	22.04.2018 16:03:58	22.04.2018 16:04:06
17	P006	Datum - rok			Rok	6	1	22.04.2018 16:05:31	22.04.2018 16:05:38
18	P007	Flashovani firmware	S limity se porovnava uspech flashovani	Nazev HEX file pro flashovani	Success?	7	1	22.04.2018 16:07:27	22.04.2018 17:01:37
19	P008	Flashovani kalibracnich dat	S limity se porovnava uspech flashovani	Nazev HEX file pro flashovani	Success?	8	1	22.04.2018 16:11:11	22.04.2018 16:11:19

Obr. 5.33: Pohled na záložku „Steps“.

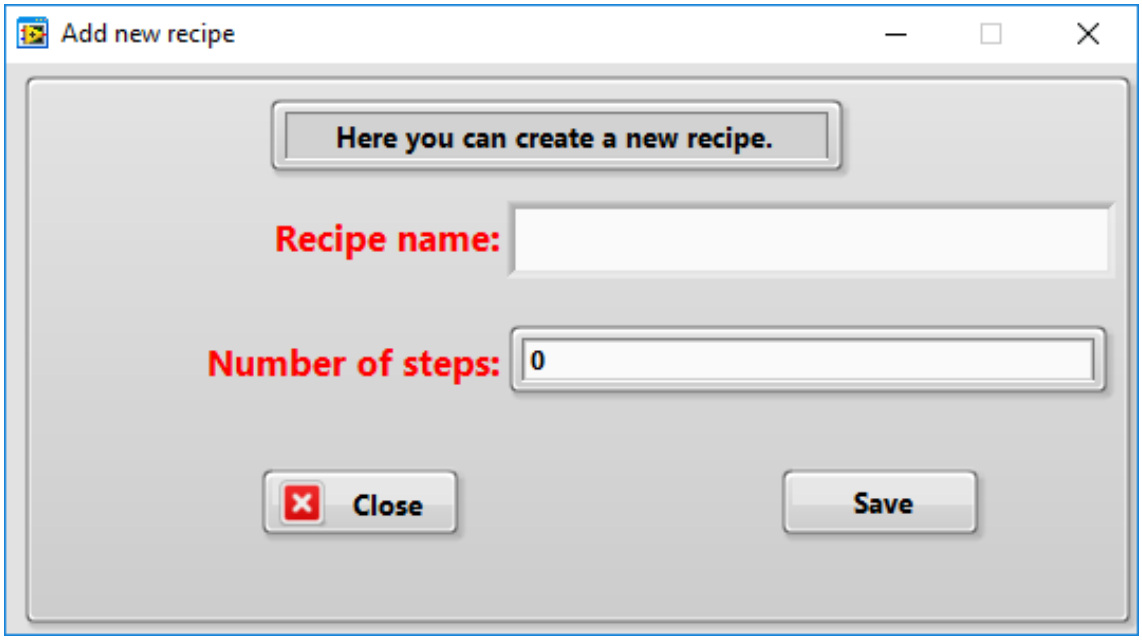


Obr. 5.34: Okno umožňující přidání nového kroku.

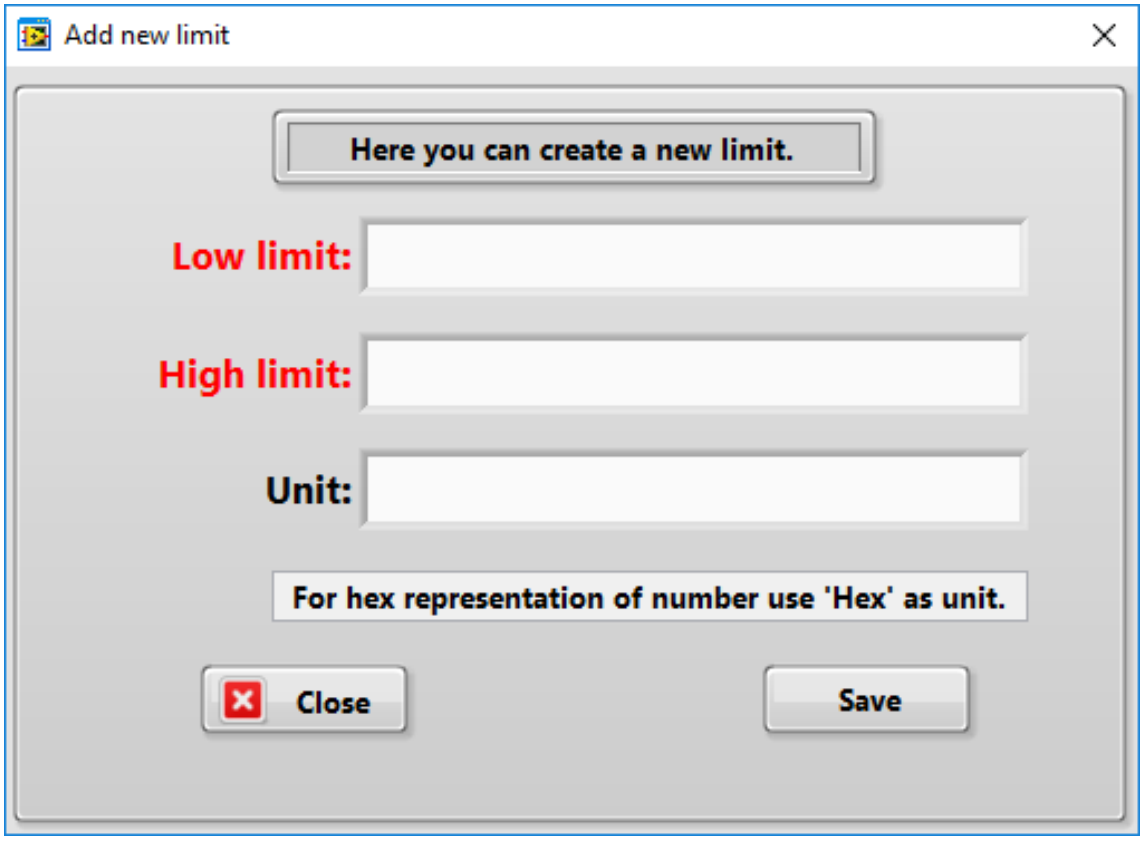


ID	Step code	Lower limit	Higher limit	Unit	Date Created	Actual Entry State	User Created	Step ID
9	P001	18	20	V	20.04.2018 18:55:40	Waiting	Testy	12
10	P001	18	20	V	21.04.2018 21:38:09	Waiting	Testy	12
11	P001	16	18	V	21.04.2018 21:41:54	Waiting	Testy	12
12	P001	12	14	V	22.04.2018 15:13:27	Active	Testy	12

Obr. 5.35: Pohled na záložku „Limits“.



Obr. 5.36: Okno pro přidání nového receptu.



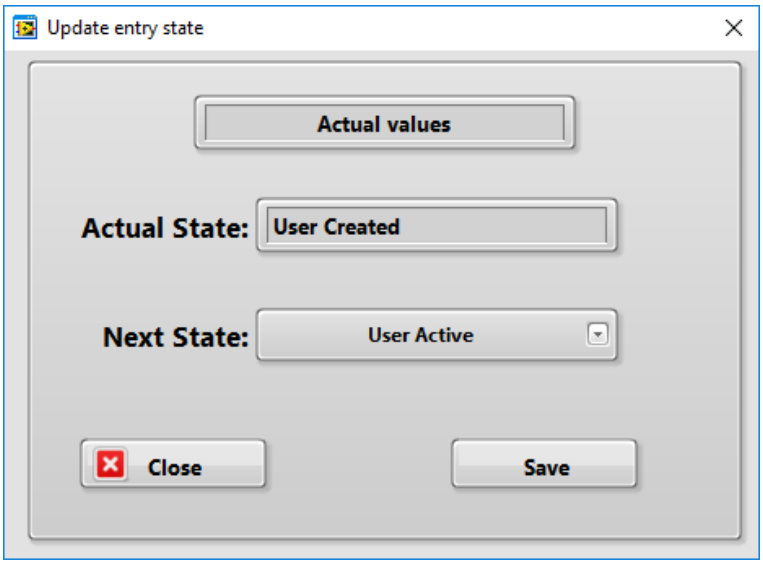
Obr. 5.37: Okno umožňující přidání nového limitu.

5.5 Modul User Setting

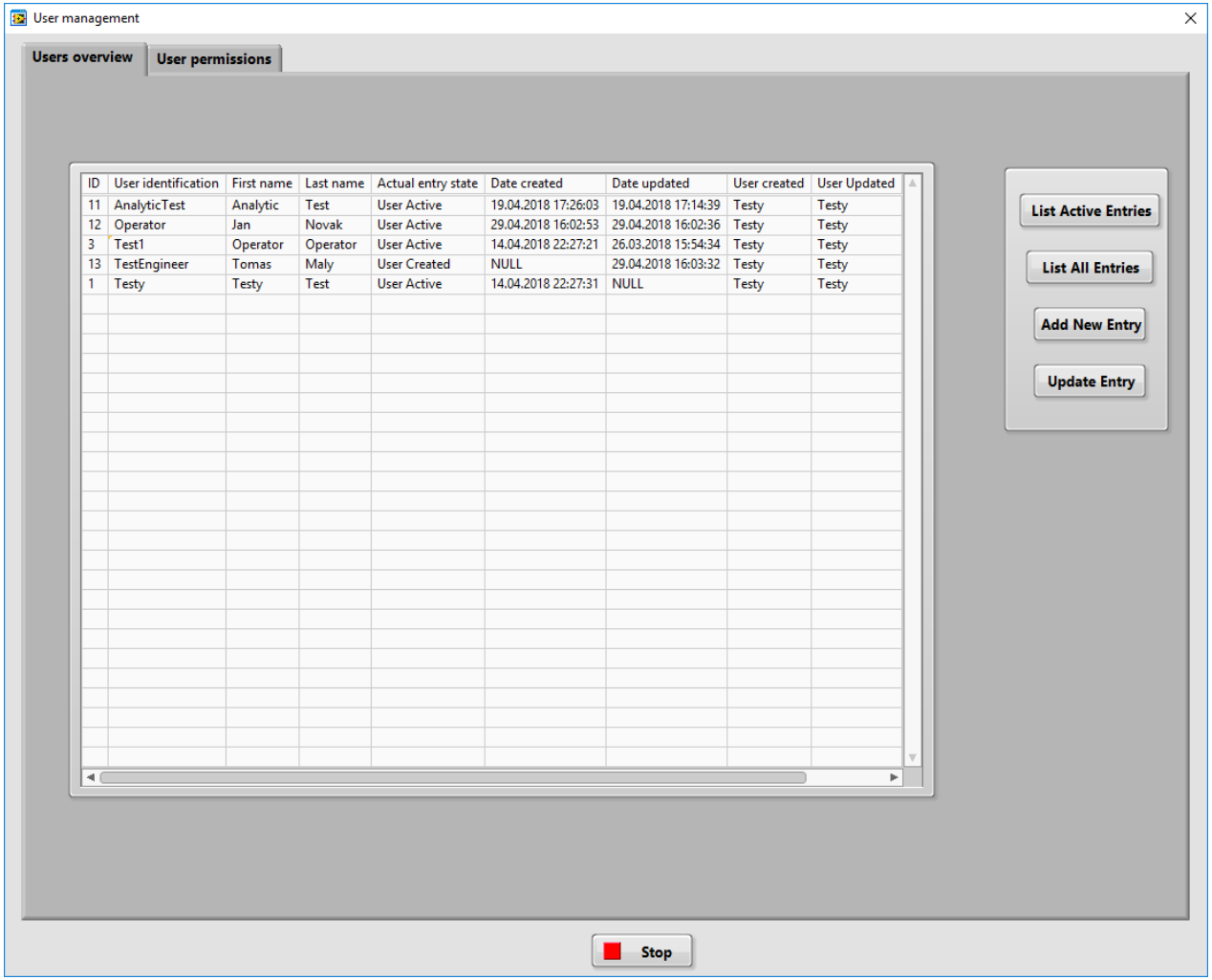
Modul pro správu uživatelů je dostupný po přihlášení uživatele s právy Developer, Test Engineer a pouze ke čtení pro uživatele s právy Analytic. Po přihlášení je zobrazen pohled na záložku „Users Overview“. Přidání nového uživatele je možno stiskem tlačítka „Add New Entry“. Detail okna pro přidání uživatele je na obrázku 5.41. Jak ukazuje obrázek, záznam v poli „Identifikacation“ musí být unikátní v celé databázi. Z rozbalovací nabídky dole je třeba vybrat roli pro nově vytvořeného uživatele.

Developer může vytvořit uživatele s jakoukoliv rolí, Test Engineer smí vytvořit uživatele pouze s právy opetor - station a nebo Test Engineer 5.42. Po přidání uživatele je status přidaného záznamu nastaven na „User Created“ a je nutné aby uživatel obsluhující uživatelské prostředí změnil status na „User Active“. Detail okna pro změnu stavu záznamu můžeme vidět na obrázku 5.38.

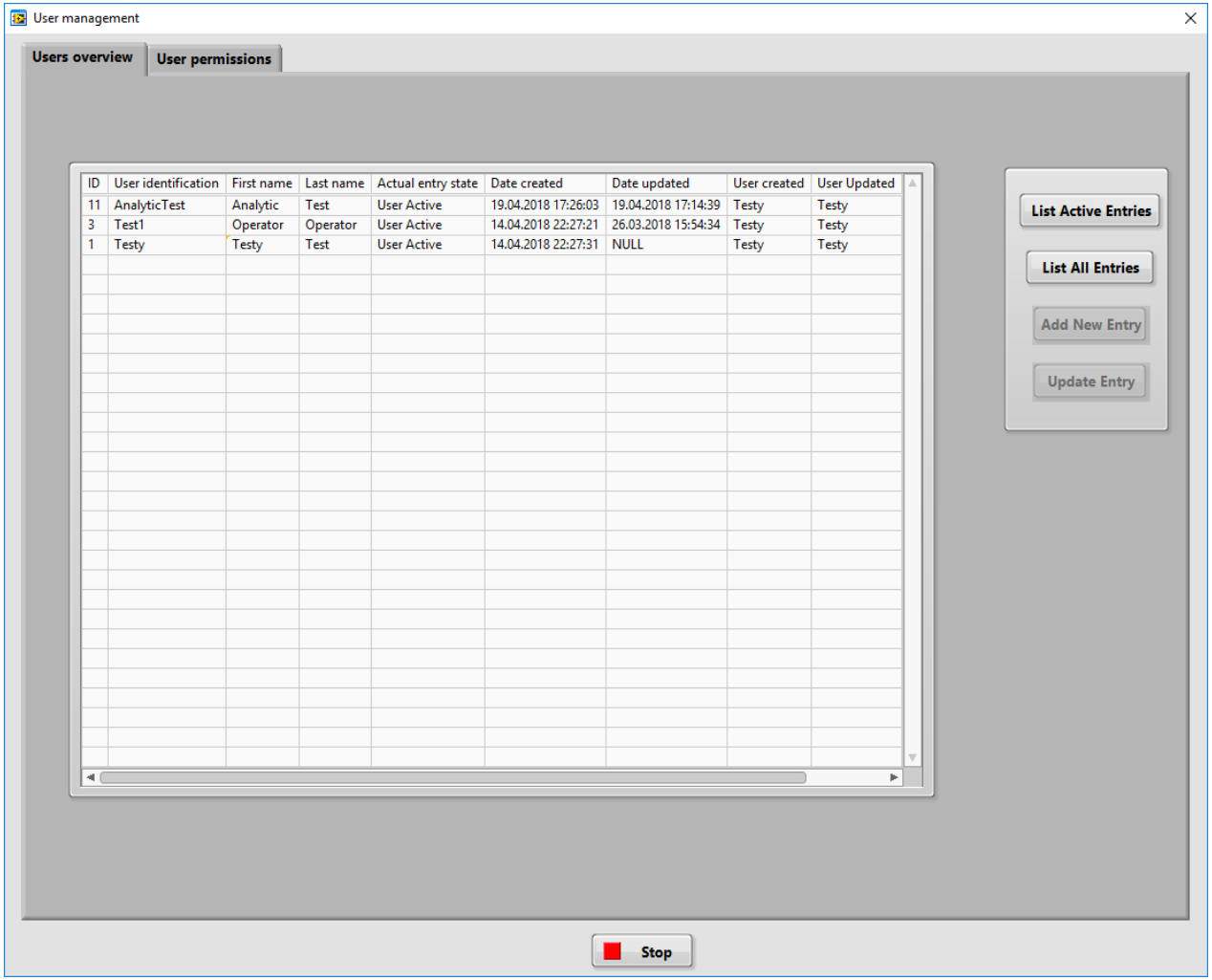
Dále je možno vybrat uživatele, jehož práva chceme zobrazit a kliknout na záložku „User Permission“ 5.43. Zde vidíme oprávnění pro zvoleného uživatele, které jsem vybrali v okně pro přidání nového uživatele. Uživatelské oprávnění je ve stavu „Waiting“ a je třeba ho nastavit pomocí aplikace na „Active“. Uživatel může mít více rolí, ale pouze jedna může být ve stavu „Active“. To, že uživatel může více rolí ve stavu „Active“ není programaticky ošetřeno. Problém nastane při přihlašování, když se z databáze načítá role příslušného uživatele po přihlášení. Pokud je jich více ve stavu „Active“, bude použita role s nižším identifikačním číslem záznamu, tedy se starším datem vytvoření.



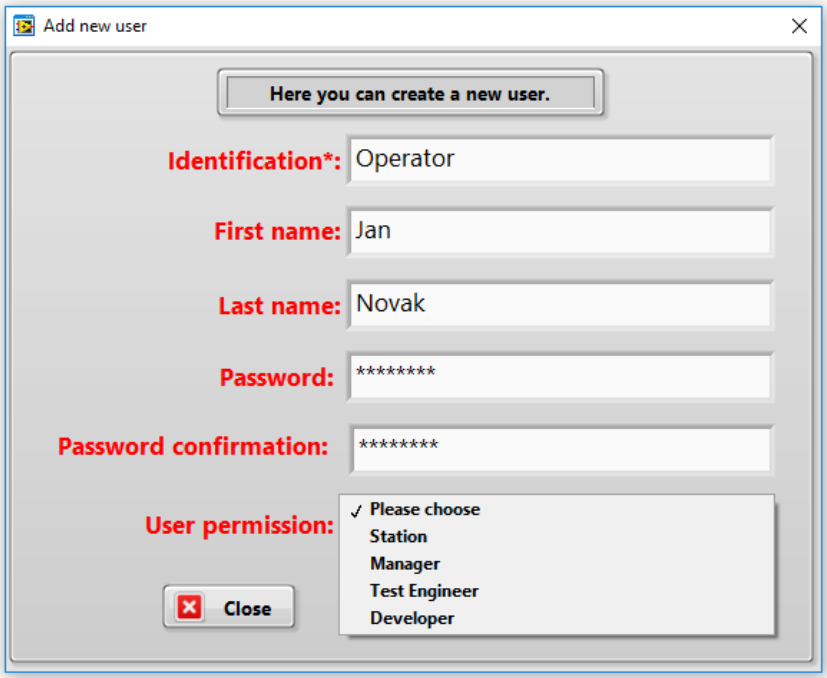
Obr. 5.38: Okno umožňující změnit stav záznamu v tabulce Uživatel.



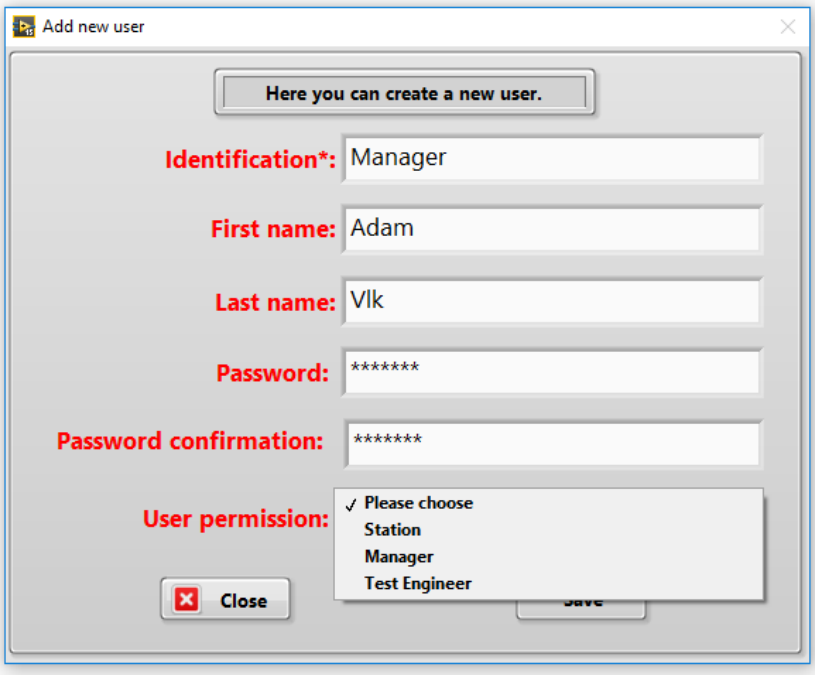
Obr. 5.39: Pohled na modul „User Setting“ po otevření.



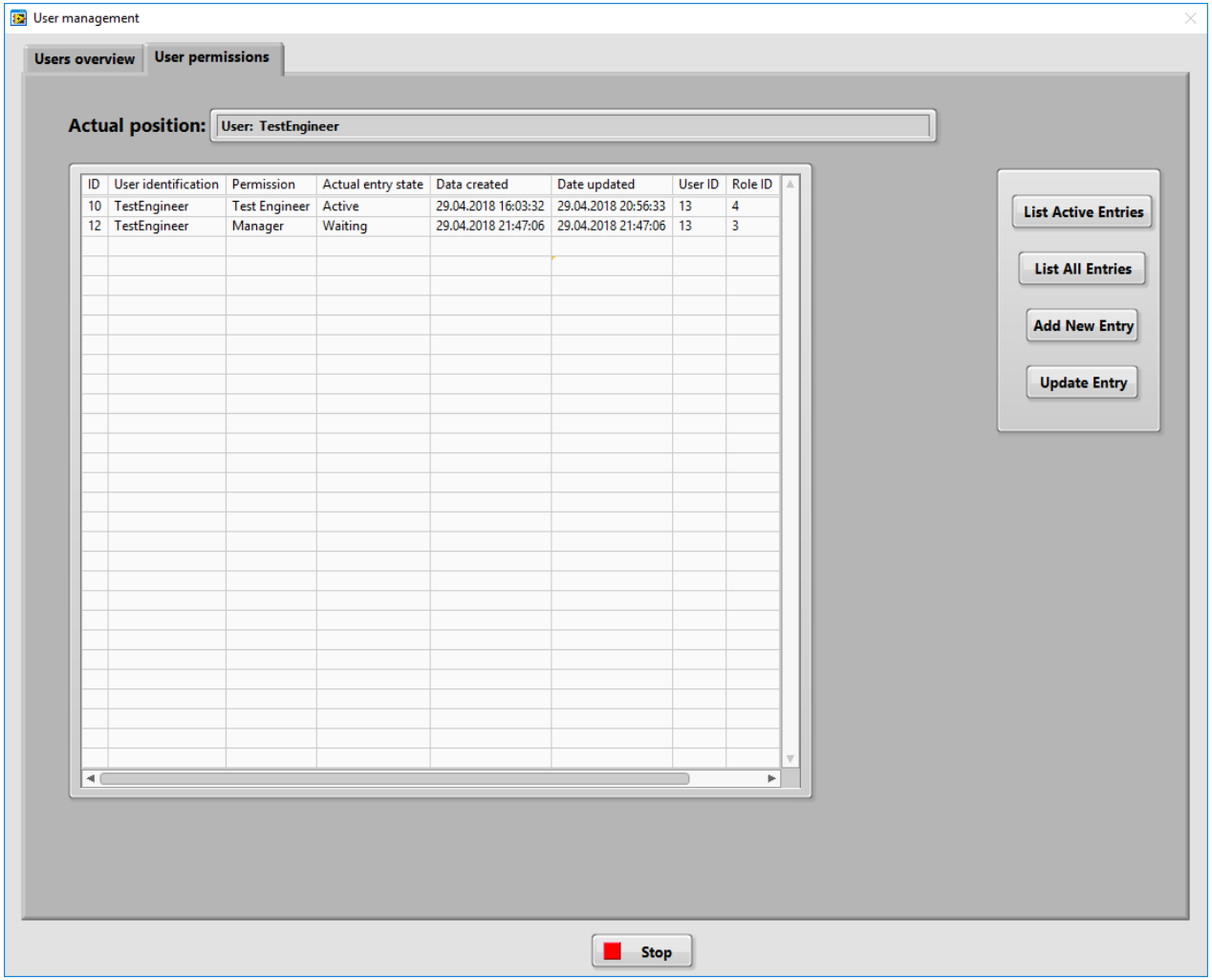
Obr. 5.40: Pohled na modul „User Setting“ po otevření uživatelem s právem Analytic.



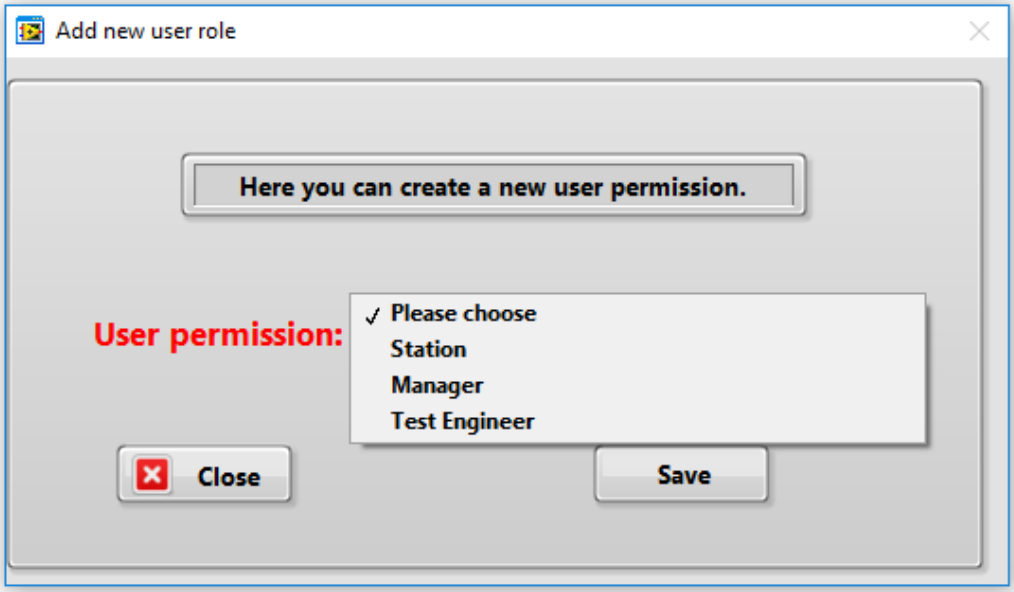
Obr. 5.41: Okno pro přidání nového uživatele. Jelikož přidávající má oprávnění Developer, jsou zobrazena všechny uživatelská oprávnění.



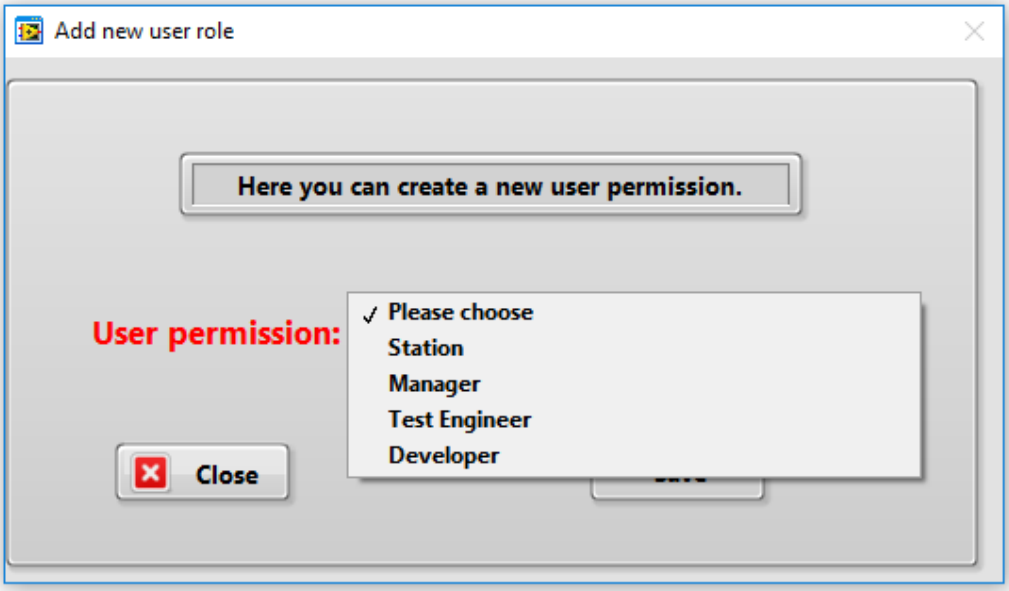
Obr. 5.42: Okno pro přidání nového uživatele. Jelikož přidávající má oprávnění Test Engineer, jsou zobrazena pouze příslušná uživatelská oprávnění.



Obr. 5.43: Pohled na záložku „User Permissions“ po otevření.



Obr. 5.44: Okno pro přidání nového oprávnění pro uživatele. Je přihlášen uživatel s právy Test Engineer.



Obr. 5.45: Okno pro přidání nového oprávnění pro uživatele. Je přihlášen uživatel s právy Developer.

5.6 Modul Results

Posledním zde popsaným modulem je modul pro zobrazení výsledků. Plný přístup k tomuto modulu mají všichni uživatelské kromě těch s oprávněním Operátor/Test station. Pohled na modul po otevření je na obrázku 5.46.

Aktuálně otevřená záložka je „Result Overview“. Zde je možno nastavit parametry a nechat si zobrazit statistiku testovaných jednotek, které prošli testovacími stanicemi.

V levé části obrazovky od tlusté svislé čáry je možno nastavit filtry pro statistiku. Je možné vybrat testovací stanici podle názvu, případě použít volbu „All Stations“ 5.47. Dále je potřeba vybrat verzi testovací stanice a nebo je opět možno použít volbu All versions. Dále je možno pomocí tlačítka „Include also test mode“ zahrnout do statistiky i záznamy nalogované v testovacím módu. Testovací mód se nastavuje pokud se aktuálně na testovací stanici odlaďuje program, případně provádí servisní zásah.

Další tlačítko umožňuje vybrat pouze záznamy v testovacím módu. Toto se bude využívat většinou pouze při servisním zásahu na programovací stanici. V rámci servisních zásahů je třeba vyzkoušet celý testovací cyklus včetně zápisu do databáze a zároveň je třeba dbát na oddělení testovacích dat a dat z ostré výroby. V případě, že jsou použita tlačítka obě, počítají se pouze záznamy v testovacím módu, jak vyplývá z pojmenování tlačítek.

Dále je možné nastavit časový interval, v němž byly hledané záznamy zapsány. Je možné použít pouze horní možnost „List DUT since“, pouze spodní možnost a nebo obě dvě najednou. Poté při stisku tlačítka „Show“ se vpravo od svislé tenké čáry zobrazí v boxu počet DUT, které skončili ve stavu „Passed“, těch, které skončili jako „Failed“ a nebo jako „Error“.

„Error“ stav znamená, že došlo k chybě v průběhu testovací sekvence a nebyli provedeny všechny její kroky. Počet nalogovaných kroků poté neodpovídá počtu kroků uvedených v receptu. Při výpočtu počtu zařízení s daným statusem se zařízení, pokud prošlo vícekrát testovací stanicí započítává pouze jednou. Pokud zařízení prošlo stanicí třikrát a pokaždé byl výsledek jiný, bude započítáno jako jedna jednotka do všech tří skupin. Pokud zařízení prošlo třikrát s výsledkem „Passed“, je započítáno pouze jednou. Toto řešení výpočtu bylo zvoleno s ohledem na nejčastější přání zákazníků. Ti chtějí vědět nejen, jestli jednotka úspěšně prošla testovací stanicí, ale také to, jestli nějaký předchozí pokus neskončil neúspěchem a nebo chybou.

Nastavené parametry v záložce „Result Overview“ jsou poté použity při přepnutí do záložky „Station Detail“. Pohled na tuto záložku ukazuje obrázek 5.49. Zde je nahoře opět lišta umožňující orientaci ve struktuře dat a dále tabulka, která obsahuje všechny záznamy o zařízeních, které prošli stanicí v časovém intervalu spe-

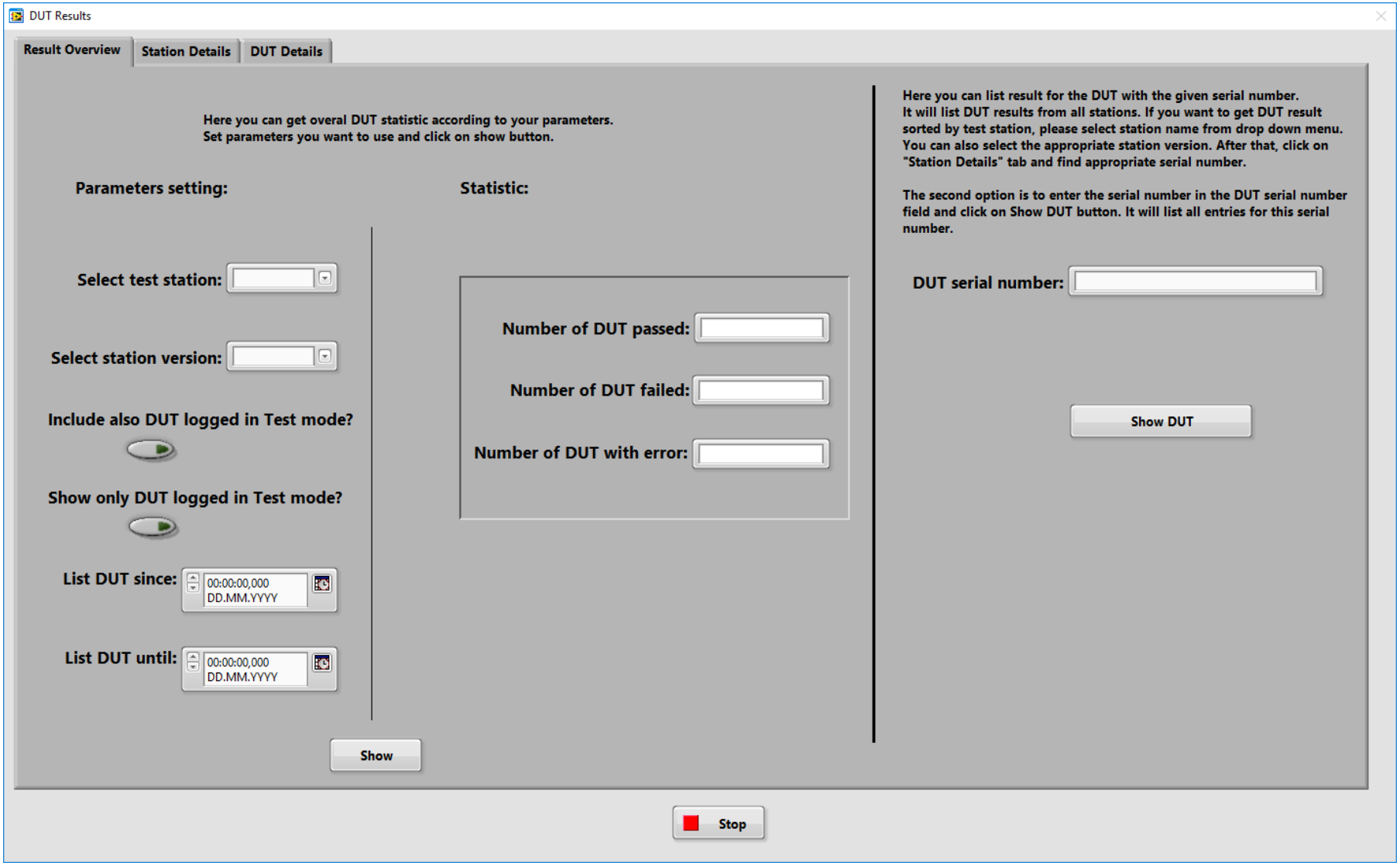
cifikovaném s parametrech v předchozí záložce. Tlačítka vpravo umožňují záznamy filtrovat a zobrazit podle výsledku.

Záznamy v tabulce jsou seřazeny podle sériového čísla. Tabulka neumožňuje další filtrování. Důvod k tomu je následující: Při ostré výrobě se záznamy s výsledkem „Error“ téměř nevyskytují. V případě, že se testovací stanici dostane do stavu „Error“ ať už z důvodu softwarové chyby a nebo vadného testovaného kusu, je stanice automaticky odstavena a čeká se na údržbu linky. O toto se stará řídící logika stanice, která je nezávislá na zapisování do databáze.

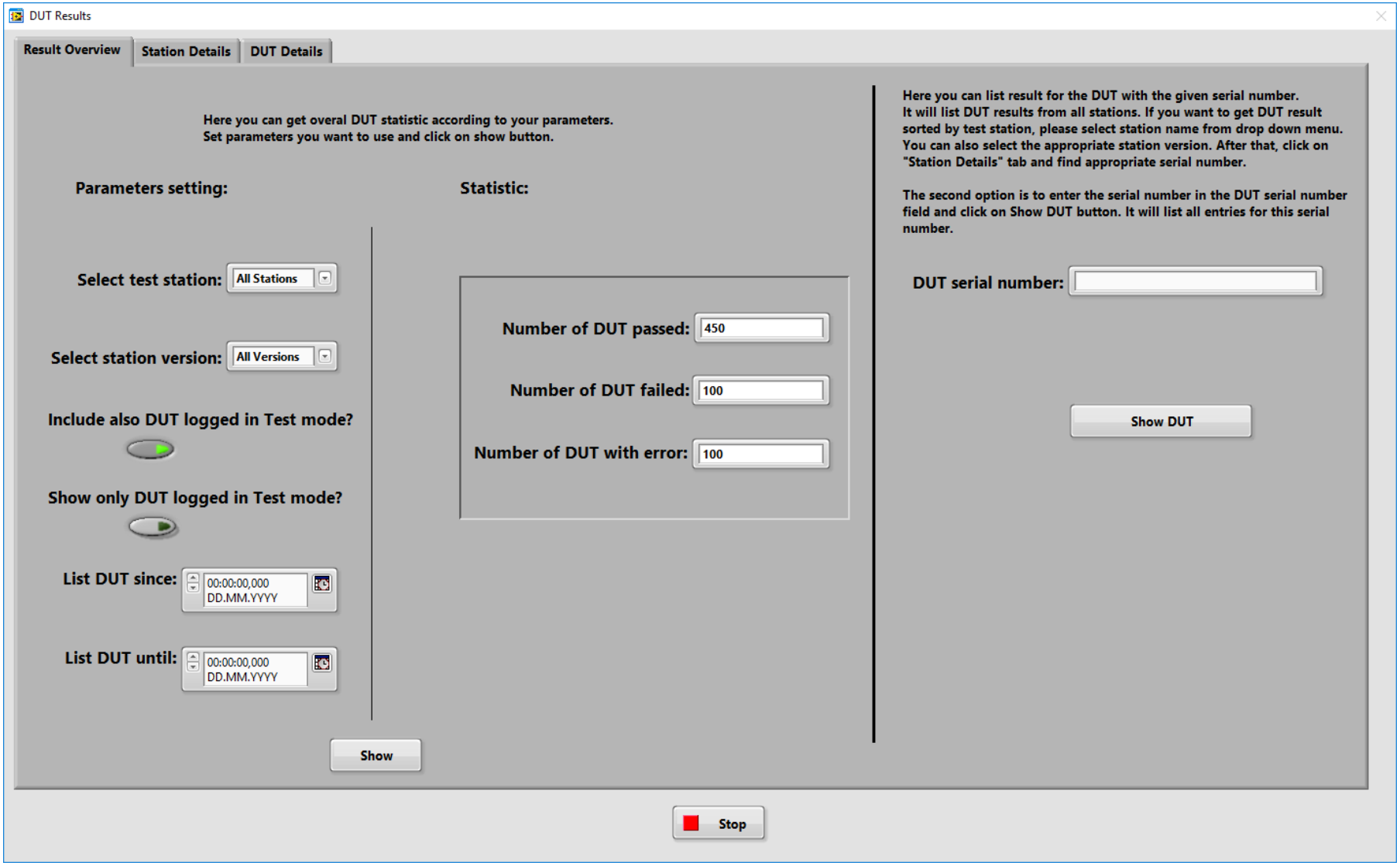
V případě, že testování jednotky skončí s výsledkem „Failed“, je testovaná jednotka zařazena mezi vadné kusy. Ve výrobě se nepočítá s větším množstvím kusů, které skončí s výsledkem „Failed“. V případě, že toto nastane v několika více kusů, je testování zastaveno a dojde ke kontrole celé várky, případně ke kontrole limitů. Test Engineer si v tom okamžiku potřebuje zobrazit pouze záznamy, které skončili jako „Failed“. Poslední tlačítko umožňuje zobrazení pouze zařízení, které prošly bez problému. Z pohledu testovací stanice a testovací linky o tyto záznamy většinou nikdo nemá zájem.

Uživatel má možnost si po dvojkliku myší na příslušný řádek tabulky zobrazit výsledky jednotlivých kroků. Po kliknutí se záložka přepne automaticky do záložky „DUT Details“. Její obsah vidíme na obrázku 5.50. Zde je opět možnost zobrazení pouze kroků, které skončili jako „Passed“ a nebo jako „Failed“.

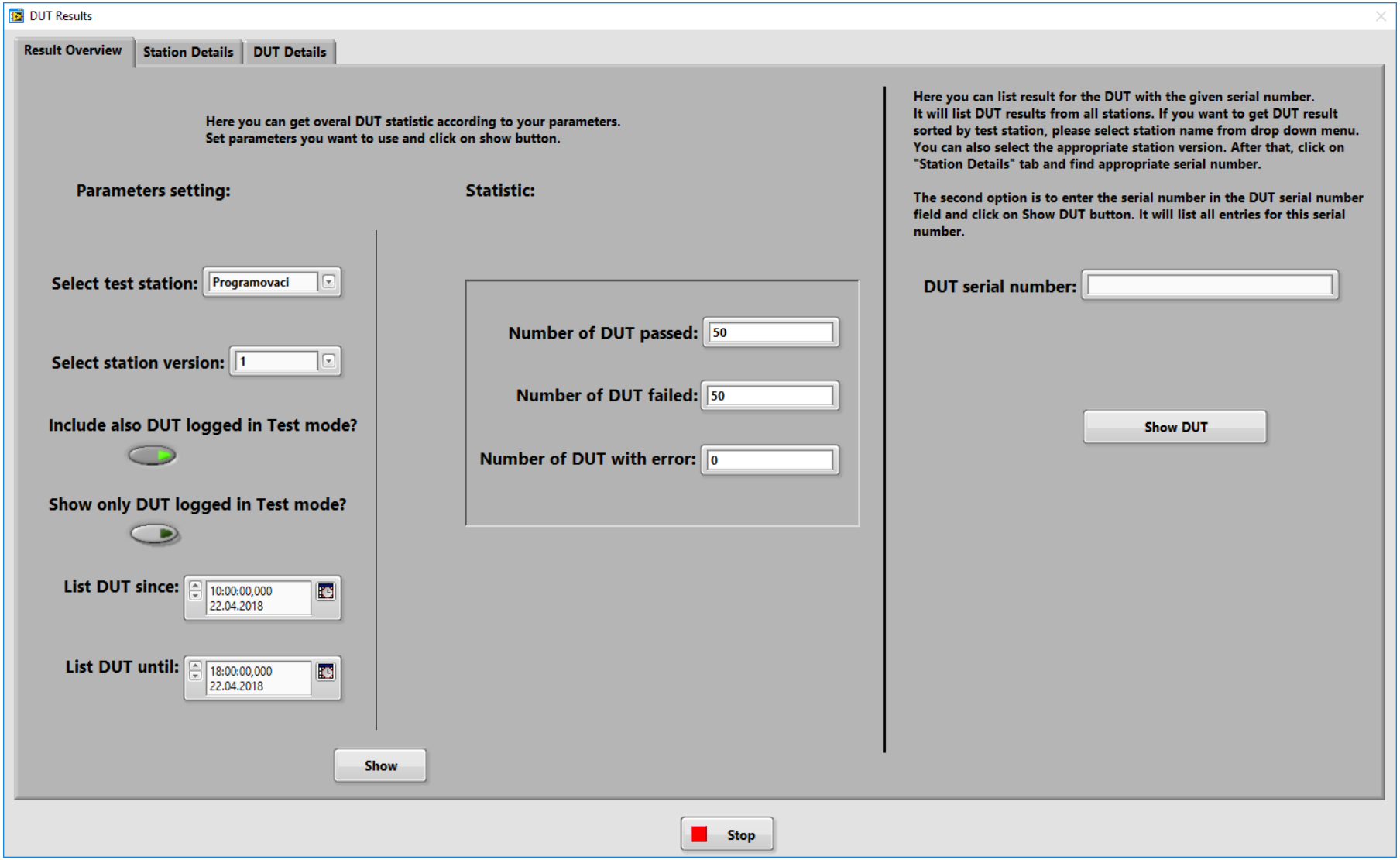
Uživatel má také možnost, jak si zobrazit průběh celého testování pro konkrétní jednotku podle jejího sériového čísla. V záložce „Results Overview“ je možno do pole „DUT serial number“ zadat sériové číslo výrobku, jehož průchod všemi stanicemi chceme zobrazit a stisknout tlačítko „Show DUT“. Poté je uživatel automaticky přepnut do záložky „DUT Details“. Jak je možné si všimnout na obrázku 5.51, navigační panel nahoře neobsahuje informaci o jakou se jedná stanici a ani její verzi, protože se zobrazují všechny testovací kroky, které byly provedeny na testované jednotce v chronologickém pořadí.



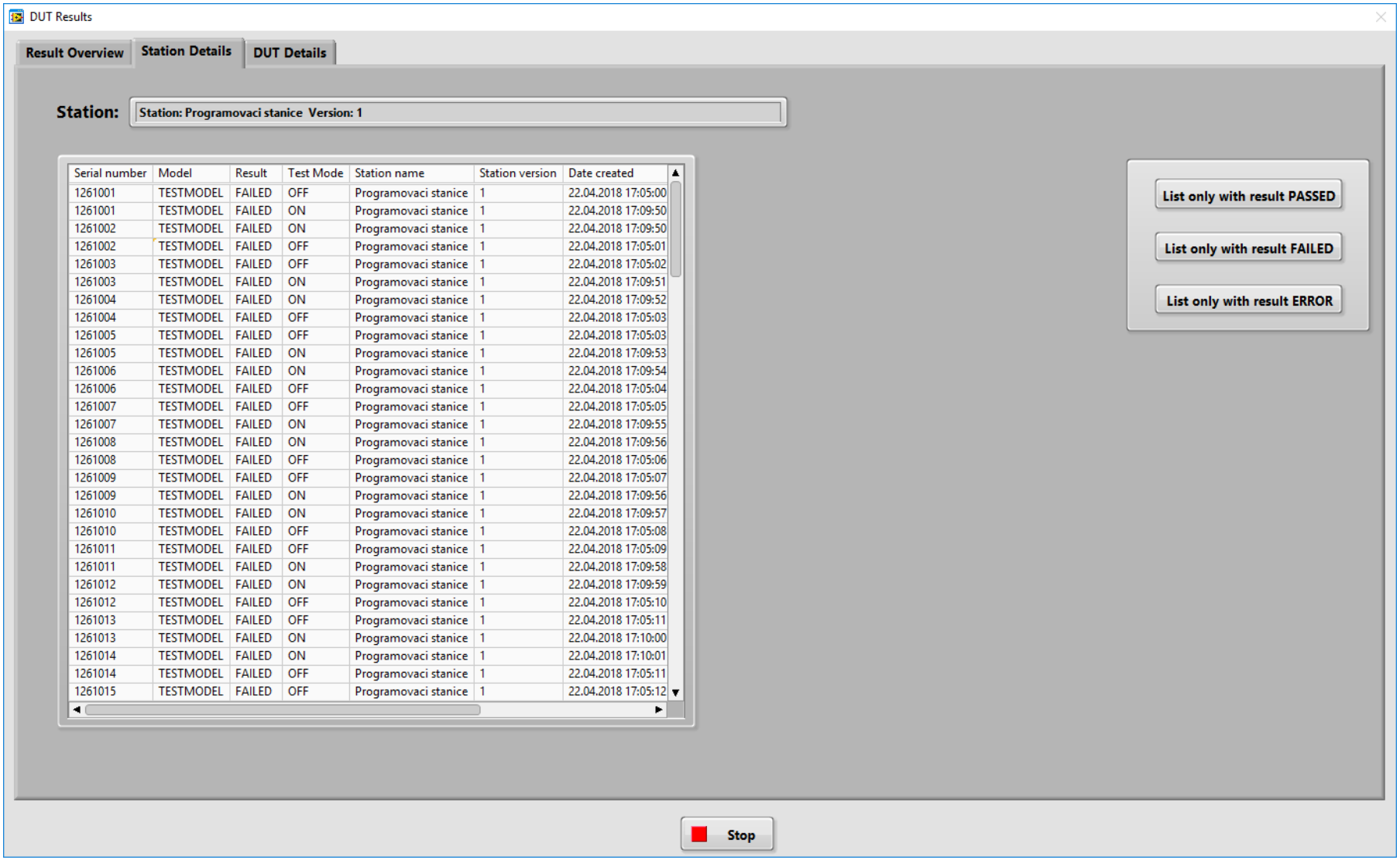
Obr. 5.46: Modul „Results“ po otevření.



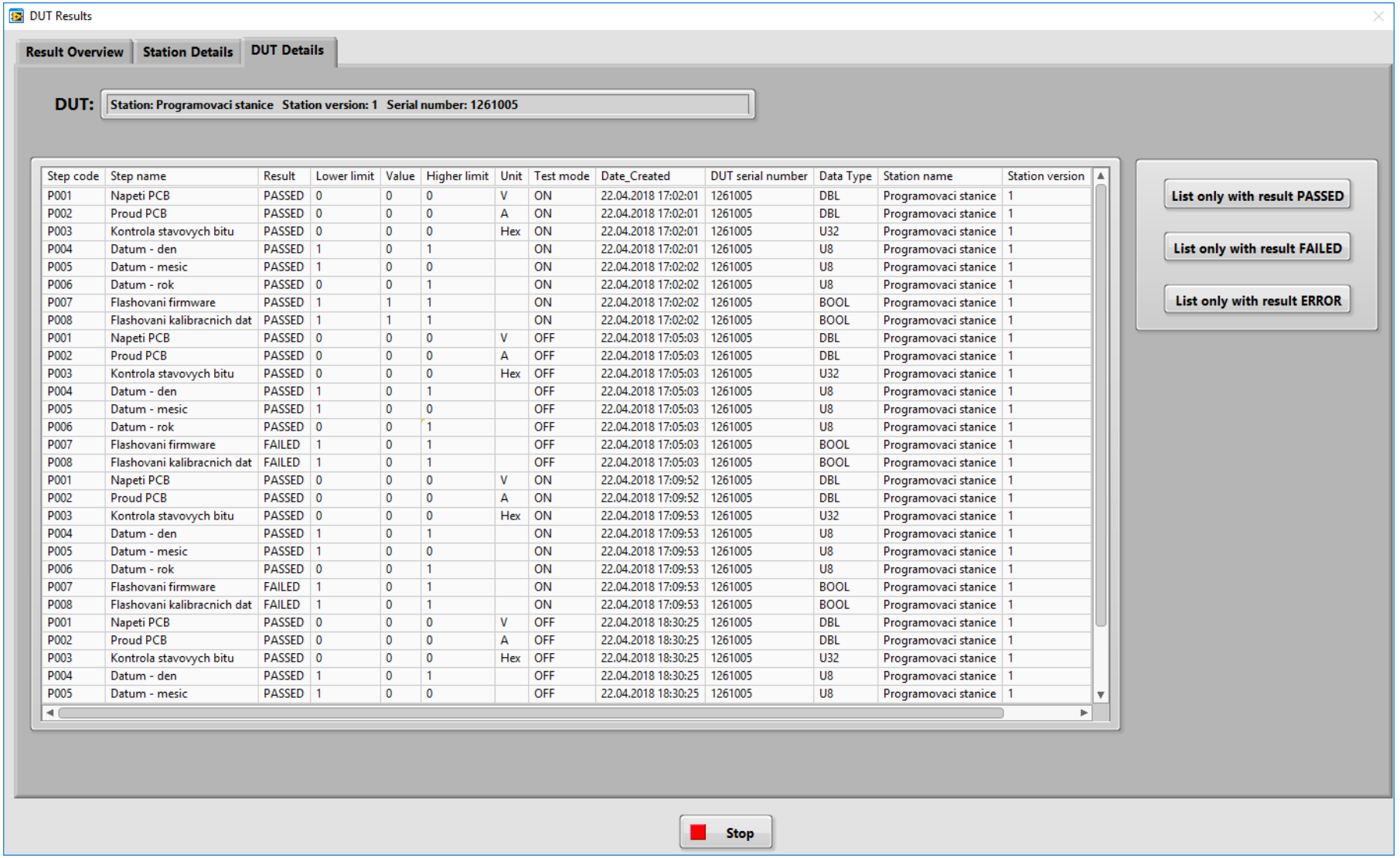
Obr. 5.47: Statistika všech výsledků ze všech stanic.



Obr. 5.48: Pohled na „Result View“ s nastaveným filtrem.

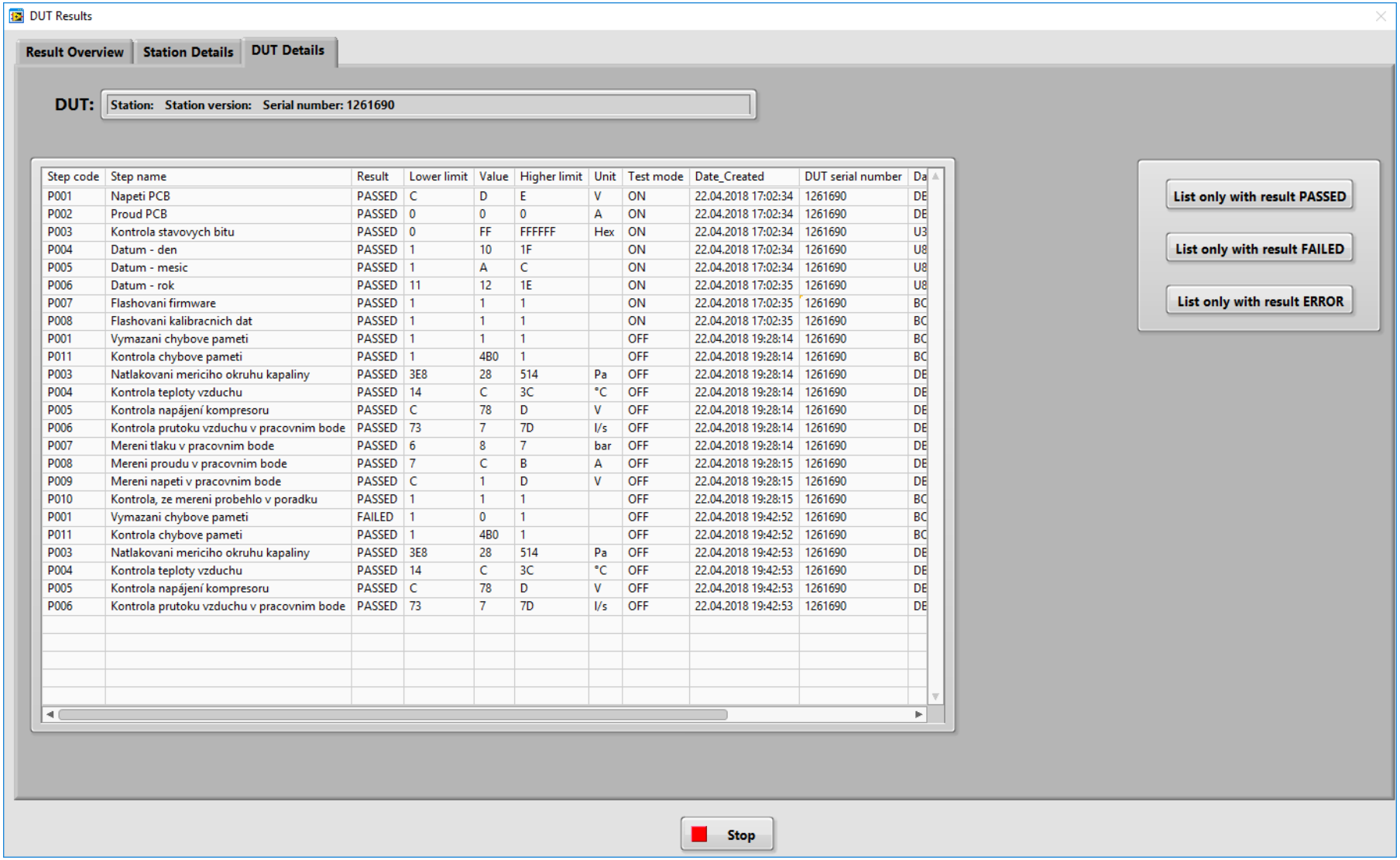


Obr. 5.49: Pohled na záložku „Station Details“.



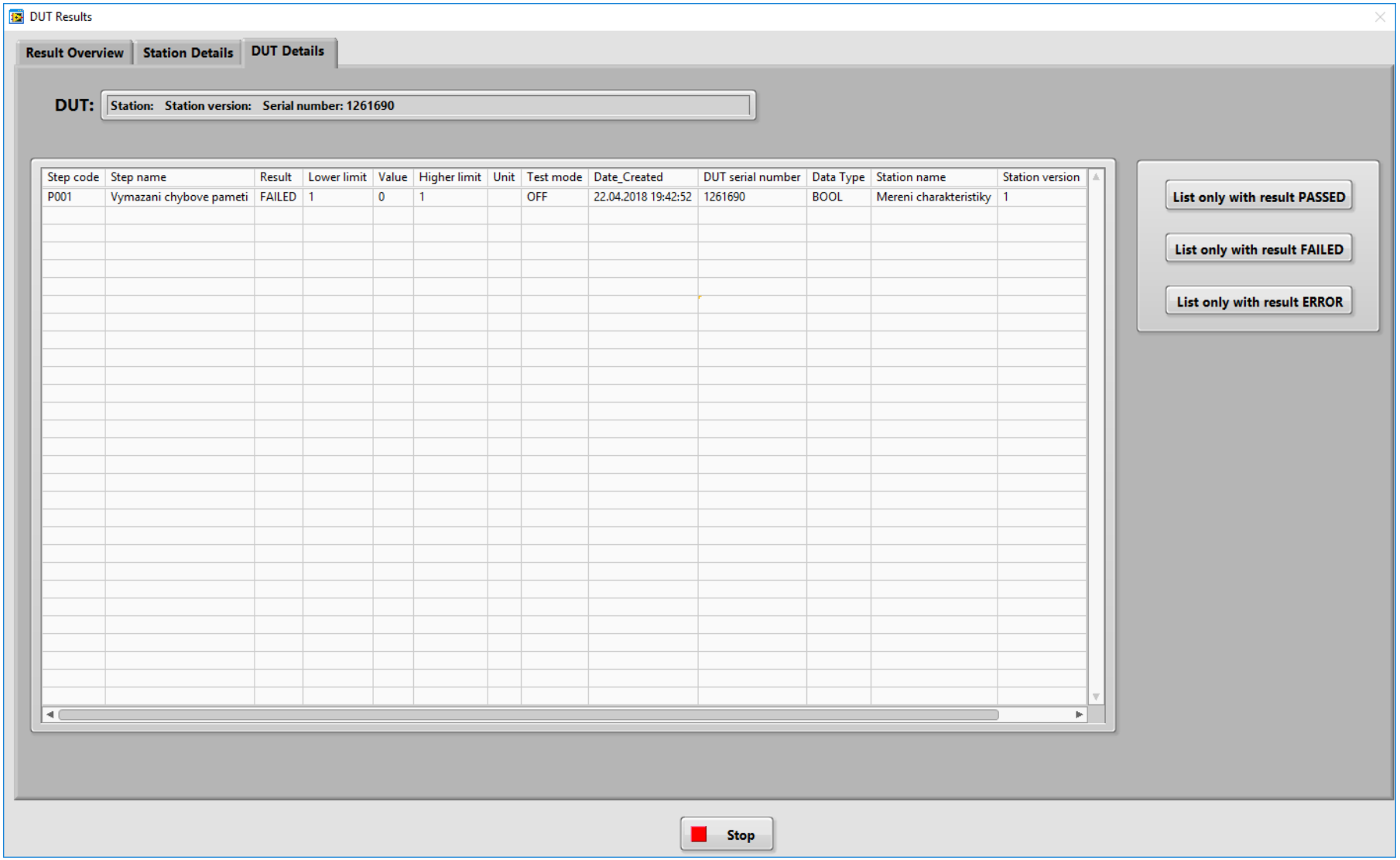
Step code	Step name	Result	Lower limit	Value	Higher limit	Unit	Test mode	Date_Created	DUT serial number	Data Type	Station name	Station version
P001	Napeti PCB	PASSED	0	0	0	V	ON	22.04.2018 17:02:01	1261005	DBL	Programovaci stanice	1
P002	Proud PCB	PASSED	0	0	0	A	ON	22.04.2018 17:02:01	1261005	DBL	Programovaci stanice	1
P003	Kontrola stavovych bitu	PASSED	0	0	0	Hex	ON	22.04.2018 17:02:01	1261005	U32	Programovaci stanice	1
P004	Datum - den	PASSED	1	0	1		ON	22.04.2018 17:02:01	1261005	U8	Programovaci stanice	1
P005	Datum - mesic	PASSED	1	0	0		ON	22.04.2018 17:02:02	1261005	U8	Programovaci stanice	1
P006	Datum - rok	PASSED	0	0	1		ON	22.04.2018 17:02:02	1261005	U8	Programovaci stanice	1
P007	Flashovani firmware	PASSED	1	1	1		ON	22.04.2018 17:02:02	1261005	BOOL	Programovaci stanice	1
P008	Flashovani kalibracnich dat	PASSED	1	1	1		ON	22.04.2018 17:02:02	1261005	BOOL	Programovaci stanice	1
P001	Napeti PCB	PASSED	0	0	0	V	OFF	22.04.2018 17:05:03	1261005	DBL	Programovaci stanice	1
P002	Proud PCB	PASSED	0	0	0	A	OFF	22.04.2018 17:05:03	1261005	DBL	Programovaci stanice	1
P003	Kontrola stavovych bitu	PASSED	0	0	0	Hex	OFF	22.04.2018 17:05:03	1261005	U32	Programovaci stanice	1
P004	Datum - den	PASSED	1	0	1		OFF	22.04.2018 17:05:03	1261005	U8	Programovaci stanice	1
P005	Datum - mesic	PASSED	1	0	0		OFF	22.04.2018 17:05:03	1261005	U8	Programovaci stanice	1
P006	Datum - rok	PASSED	0	0	1		OFF	22.04.2018 17:05:03	1261005	U8	Programovaci stanice	1
P007	Flashovani firmware	FAILED	1	0	1		OFF	22.04.2018 17:05:03	1261005	BOOL	Programovaci stanice	1
P008	Flashovani kalibracnich dat	FAILED	1	0	1		OFF	22.04.2018 17:05:03	1261005	BOOL	Programovaci stanice	1
P001	Napeti PCB	PASSED	0	0	0	V	ON	22.04.2018 17:09:52	1261005	DBL	Programovaci stanice	1
P002	Proud PCB	PASSED	0	0	0	A	ON	22.04.2018 17:09:52	1261005	DBL	Programovaci stanice	1
P003	Kontrola stavovych bitu	PASSED	0	0	0	Hex	ON	22.04.2018 17:09:53	1261005	U32	Programovaci stanice	1
P004	Datum - den	PASSED	1	0	1		ON	22.04.2018 17:09:53	1261005	U8	Programovaci stanice	1
P005	Datum - mesic	PASSED	1	0	0		ON	22.04.2018 17:09:53	1261005	U8	Programovaci stanice	1
P006	Datum - rok	PASSED	0	0	1		ON	22.04.2018 17:09:53	1261005	U8	Programovaci stanice	1
P007	Flashovani firmware	FAILED	1	0	1		ON	22.04.2018 17:09:53	1261005	BOOL	Programovaci stanice	1
P008	Flashovani kalibracnich dat	FAILED	1	0	1		ON	22.04.2018 17:09:53	1261005	BOOL	Programovaci stanice	1
P001	Napeti PCB	PASSED	0	0	0	V	OFF	22.04.2018 18:30:25	1261005	DBL	Programovaci stanice	1
P002	Proud PCB	PASSED	0	0	0	A	OFF	22.04.2018 18:30:25	1261005	DBL	Programovaci stanice	1
P003	Kontrola stavovych bitu	PASSED	0	0	0	Hex	OFF	22.04.2018 18:30:25	1261005	U32	Programovaci stanice	1
P004	Datum - den	PASSED	1	0	1		OFF	22.04.2018 18:30:25	1261005	U8	Programovaci stanice	1
P005	Datum - mesic	PASSED	1	0	0		OFF	22.04.2018 18:30:25	1261005	U8	Programovaci stanice	1

Obr. 5.50: Záložka ukazující průběh testování jednotky s vybraným sériovým číslem na stanici, jejíž název a verzi lze přečíst v panelu nahoře.



Step code	Step name	Result	Lower limit	Value	Higher limit	Unit	Test mode	Date_Created	DUT serial number	Da
P001	Napeti PCB	PASSED	C	D	E	V	ON	22.04.2018 17:02:34	1261690	DE
P002	Proud PCB	PASSED	0	0	0	A	ON	22.04.2018 17:02:34	1261690	DE
P003	Kontrola stavovych bitu	PASSED	0	FF	FFFFFF	Hex	ON	22.04.2018 17:02:34	1261690	U3
P004	Datum - den	PASSED	1	10	1F		ON	22.04.2018 17:02:34	1261690	U8
P005	Datum - mesic	PASSED	1	A	C		ON	22.04.2018 17:02:34	1261690	U8
P006	Datum - rok	PASSED	11	12	1E		ON	22.04.2018 17:02:35	1261690	U8
P007	Flashovani firmware	PASSED	1	1	1		ON	22.04.2018 17:02:35	1261690	BC
P008	Flashovani kalibracnich dat	PASSED	1	1	1		ON	22.04.2018 17:02:35	1261690	BC
P001	Vymazani chybove pameti	PASSED	1	1	1		OFF	22.04.2018 19:28:14	1261690	BC
P011	Kontrola chybove pameti	PASSED	1	4B0	1		OFF	22.04.2018 19:28:14	1261690	BC
P003	Natlakovani mericiho okruhu kapaliny	PASSED	3E8	28	514	Pa	OFF	22.04.2018 19:28:14	1261690	DE
P004	Kontrola teploty vzduchu	PASSED	14	C	3C	°C	OFF	22.04.2018 19:28:14	1261690	DE
P005	Kontrola napájení kompresoru	PASSED	C	78	D	V	OFF	22.04.2018 19:28:14	1261690	DE
P006	Kontrola prutoku vzduchu v pracovnim bode	PASSED	73	7	7D	l/s	OFF	22.04.2018 19:28:14	1261690	DE
P007	Mereni tlaku v pracovnim bode	PASSED	6	8	7	bar	OFF	22.04.2018 19:28:14	1261690	DE
P008	Mereni proudu v pracovnim bode	PASSED	7	C	B	A	OFF	22.04.2018 19:28:15	1261690	DE
P009	Mereni napeti v pracovnim bode	PASSED	C	1	D	V	OFF	22.04.2018 19:28:15	1261690	DE
P010	Kontrola, ze mereni probehlo v poradku	PASSED	1	1	1		OFF	22.04.2018 19:28:15	1261690	BC
P001	Vymazani chybove pameti	FAILED	1	0	1		OFF	22.04.2018 19:42:52	1261690	BC
P011	Kontrola chybove pameti	PASSED	1	4B0	1		OFF	22.04.2018 19:42:52	1261690	BC
P003	Natlakovani mericiho okruhu kapaliny	PASSED	3E8	28	514	Pa	OFF	22.04.2018 19:42:53	1261690	DE
P004	Kontrola teploty vzduchu	PASSED	14	C	3C	°C	OFF	22.04.2018 19:42:53	1261690	DE
P005	Kontrola napájení kompresoru	PASSED	C	78	D	V	OFF	22.04.2018 19:42:53	1261690	DE
P006	Kontrola prutoku vzduchu v pracovnim bode	PASSED	73	7	7D	l/s	OFF	22.04.2018 19:42:53	1261690	DE

Obr. 5.51: Detail průběhu jednotky s daným sériovým číslem všemi stanicemi. Toto je zobrazení po použití tlačítka „Show DUT details“ v „Result Overview“.



Obr. 5.52: Zobrazení pouze kroků, které skončili stavem „Failed“ pro DUT na všech stanicích.

6 OVĚŘENÍ FUNKČNOSTI DATABÁZE

Pro ověřené funkčnosti databáze byla vytvořena aplikace pro zápis dat z testovací stanice. Moduly této aplikace odpovídají specifikaci v předchozí kapitole a budou později použity jako jeden modul, které bude paralelně připojen k LabVIEW architektuře, která řídí vlastní testovací stanici. Kvůli zachování know-how firmy tato architektura není použita v diplomové práci. Místo toho bylo vytvořeno VI, které simuluje odesílání naměřených dat. Jelikož se nepočítá s využitím celého aplikace nadále, ale pouze jejich částí, je tato aplikace pouze jako LabVIEW project, kde je potřeba spustit hlavní VI, které poté volá své podřízené VI. Nebyl vytvořen spustitelný soubor „EXE“ a ani uživatelské prostředí.

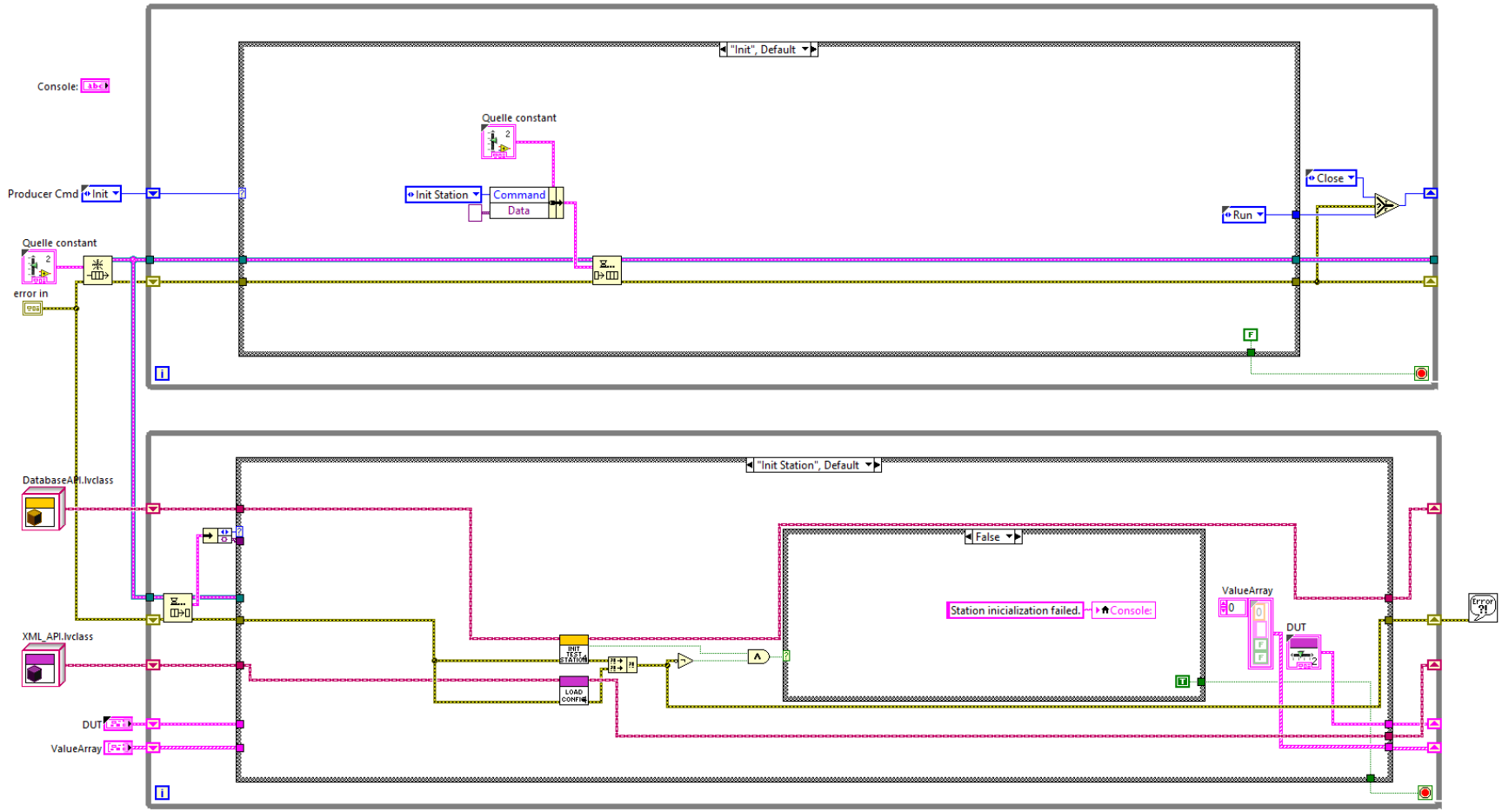
6.1 Aplikace pro zápis dat z testovací stanice

Jako architektura pro testovací aplikaci byla zvolena architektura „Producer - consumer“. Původní návrh byl s využitím Actor Frameworku, ale jak bylo zmíněno v předchozí kapitole, kvůli nefunkčnosti některých knihovních funkcí poskytovaných firmou National Instruments použit nebyl. Jedná se o dvě paralelně běžící smyčky, kdy první smyčka - „Producer“ produkuje data a odesílá je druhé smyčce. Tato architektura byla zvolena kvůli její podobnosti s cílovým využitím aplikace. „Producerem“ dat a také řídící aplikací bude poté LV architektura a „consumerem“ bude aplikace pro zápis do databáze vytvořená v rámci této diplomové práce. Pohled na celkový návrh je na obrázku 6.1.

6.1.1 Generátor dat

Obě dvě smyčky obsahují stavový automat. „Producer“ řídí druhou smyčku pomocí generování uživatelských událostí, které se vkládají do fronty a poté se obsluhují ve druhé smyčce. Událost obsahuje název události, který je shodný s názvem stavu ve stavovém automatu „consumerské“ smyčky a volitelně data, které „consumerská“ smyčka očekává. Stavový automat producentské smyčky má celkem 3 stavy:

- Stav „Init“, ve kterém smyčka začíná a je poslána událost „Init“ do „consumerské“ smyčky, která se postará o inicializaci stanice. Po odeslání události se automaticky přejde do stavu „Run“.
- Ve stavu „Run“ se opakovaně volá VI, které slouží jako generátor dat. Pohled na jeho zdrojový diagram je na obrázku 6.2.Toto VI simuluje data z reálné měřící stanice. Zde je třeba naplnit daty 3 tabulky: První obsahuje kód kroku, druhá obsahuje naměřené hodnoty a třetí obsahuje úspěch/neúspěch měřeného kroku. Poté se z poskytnutých tabulek naplní cluster(struktura pro ukládání



Obr. 6.1: Pohled na architekturu „Producer - consumer“ realizovanou v LabVIEW. Horní smyčka slouží jako producent dat pro smyčku spodní.

dat) a je vytvořena událost „New DUT“ a přidána do fronty. Poté následuje podle počtu kroků vytvoření odpovídajícího počtu událostí „DUT Step“ a nakonec se přidá událost „DUT End“, která znamená ukončení testovacího cyklu pro dané DUT. V případě ostrého provozu by tyto události generovala testovací stanice v průběhu měření. Po vykonání požadovaného počtu cyklů generování dat přejde stavový automat do posledního stavu.

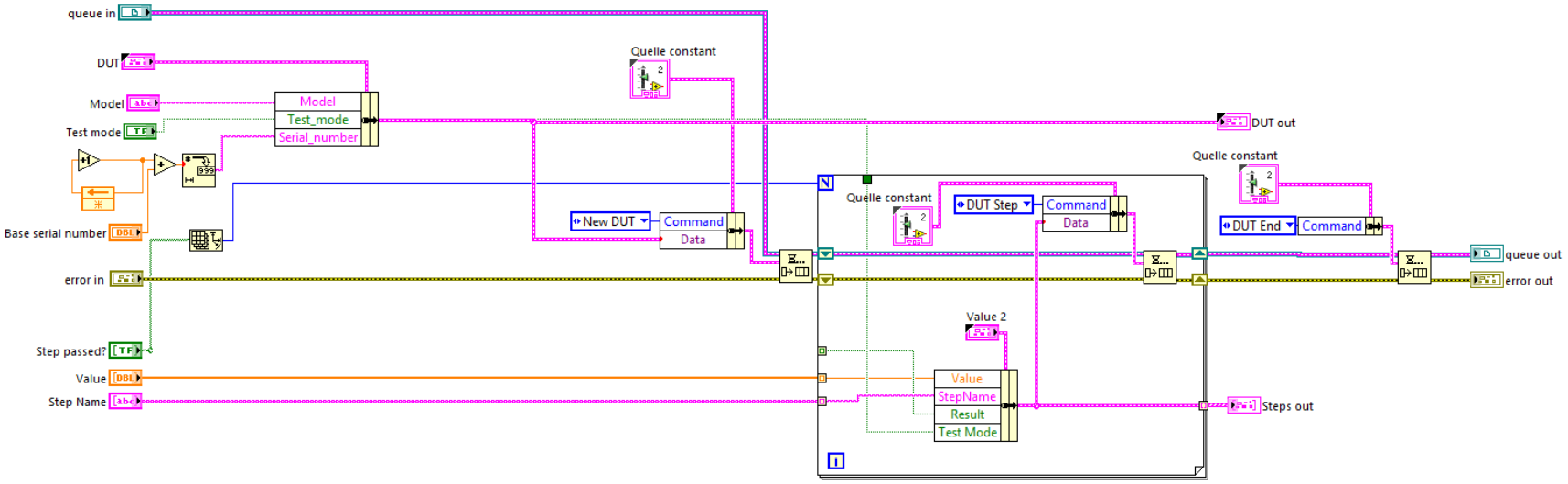
- Poslední stav je „Close“. Zde se odešle „consumerovi“ událost „Close“ a horní „while“ smyčka, která obsahuje stavový automat se ukončí. „Consumer“ se poté postará o korektní zavření aplikace a uložení dat pro stanici.

6.1.2 Zápis dat na server

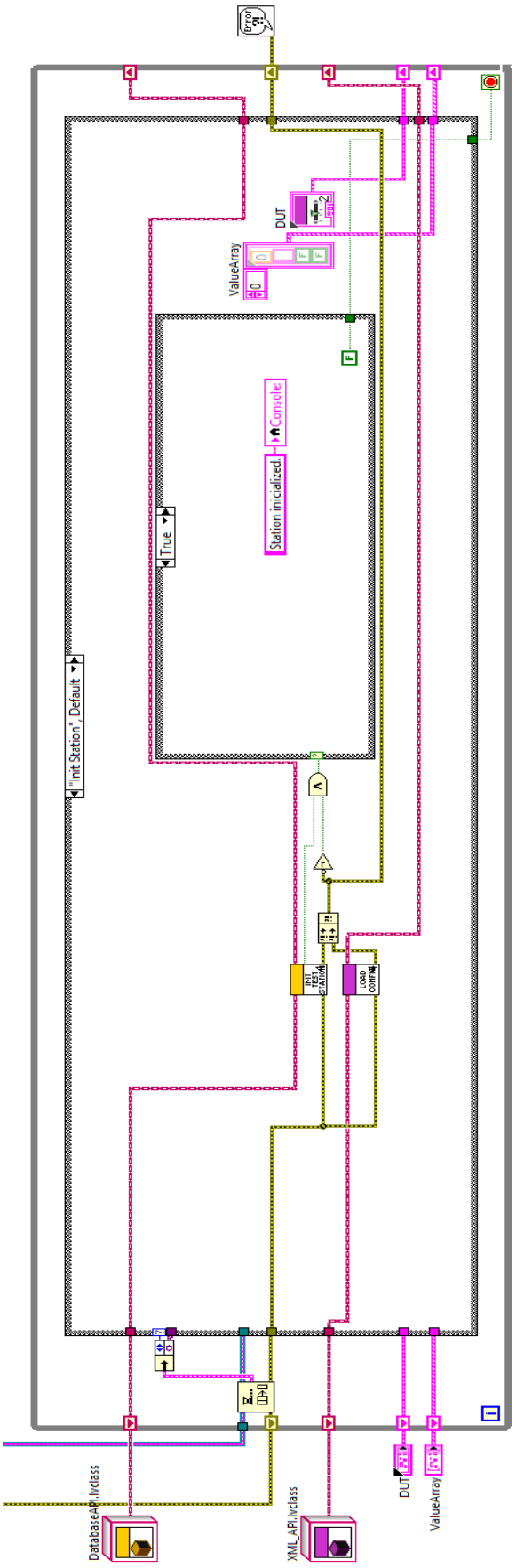
Stavový automat v „consumerské“ smyčce obsahuje 5 stavů, které odpovídají událostem generovaným „producerskou“ smyčkou. „Consumerská“ smyčka při každé nové iteraci „while“ smyčky čeká na událost, poté událost odebere z FIFO fronty a přepne se do příslušného stavu. Obsluha událostí je popsána v následujících subkapitolách.

6.1.3 Inicializace testovací stanice

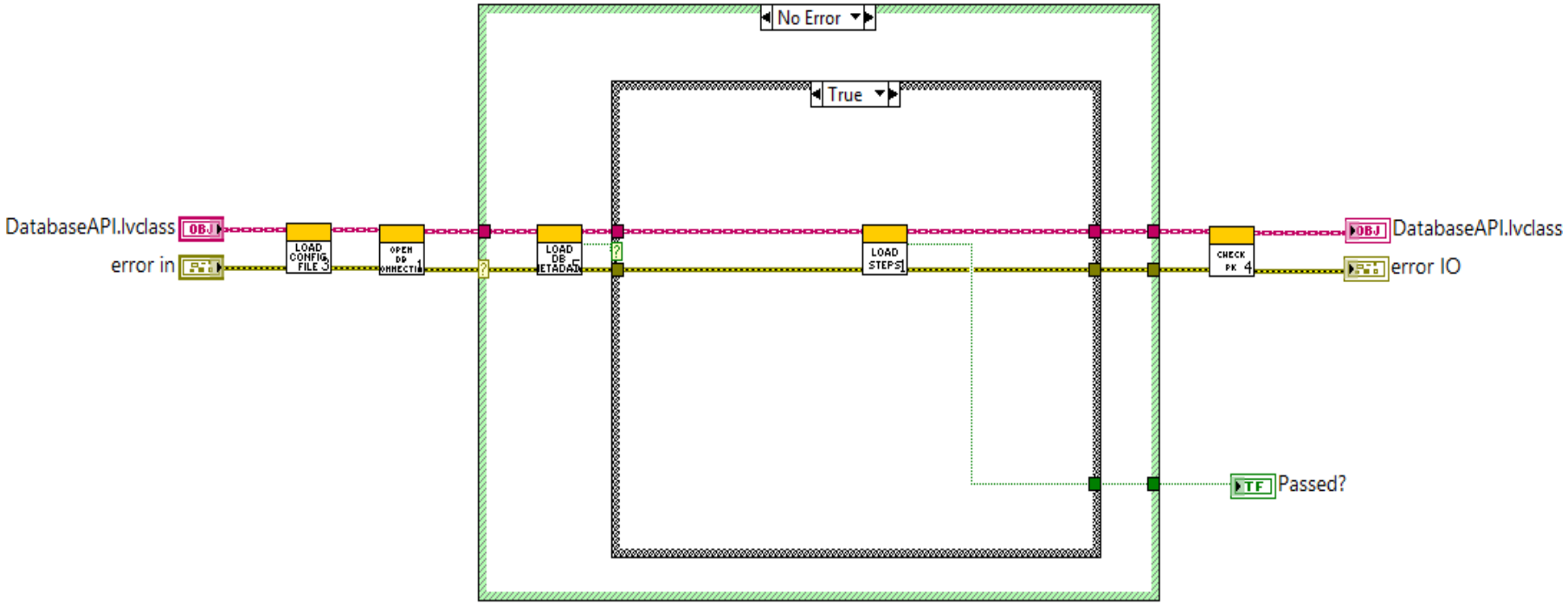
Při obsluze této události dojde k inicializaci testovací stanice. Parametry pro testovací stanici jsou uloženy v souboru „config.ini“. Tento soubor obsahuje název testovací stanice a uložený poslední stav počítadla vytvořených záznamů do tabulky DUT a Value. Do privátní proměnné třídy DatabaseAPI jsou načteny hodnoty počítadel a název stanice. Zároveň jsou ze souboru „config.ini“ nainicializovány i privátní proměnné třídy pro zápis výsledků do XML. Blokový diagram pro obsluhu této události je na obrázku zde: 6.3. Dále je navázáno spojení s databází. SubVI zodpovědné za navázání spojení s databází je na obrázku 6.4. Pokud není možné navázat spojení s databází, je celá testovací aplikace ukončena, protože není možno zjistit aktivní testovací předpis. Pokud je spojení navázáno, je vyhledána aktivní stanice s daným jménem a její ID číslo z databáze je uloženo. Poté je vyhledán aktivní recept s cizím klíčem ID testovací stanice. Pokud neexistuje žádný odpovídající záznam ve stavu „Active“ a nebo pokud je záznamů více než jeden, je na tuto skutečnost uživatel upozorněn a testovací aplikace se ukončí. Pokud byl úspěšně vyhledán recept, jsou z databáze načteny ID všech jeho kroků. Pokud počet záznamů v tabulce Step, které mají jako cizí klíč ID receptu a jsou ve stavu „Active“, neodpovídá zadanému požadovanému počtu kroků z receptu, je testovací aplikace ukončena. Pokud je nalezen odpovídající počet záznamů v tabulce Step, je dalším krokem načtení limitů pro dané kroky. Pokud jsou nalezeny všechny limity pro příslušné kroky, je stanice nainicializována.



Obr. 6.2: Pohled na block diagram VI pro generování dat. Zde je ukázáno generování událostí „New DUT“, „DUT Step“ a „DUT End“



Obr. 6.3: Blokový diagram pro obsluhu události „Init“, kde dochází i inicializaci testovací stanice.



Obr. 6.4: SubVI, které naváže kontakt s databází a načte aktivní testovací předpisy.

6.1.4 Události „New DUT“ a „New Step“

Při obsluze těchto událostí se vyjme z fronty kromě pokynu o jakou událost se jedná i data, které událost přenáší. Tyto data se nastaví do shift registru - proměnné, kde jsou uloženy dokud nepřijde událost „DUT End“.

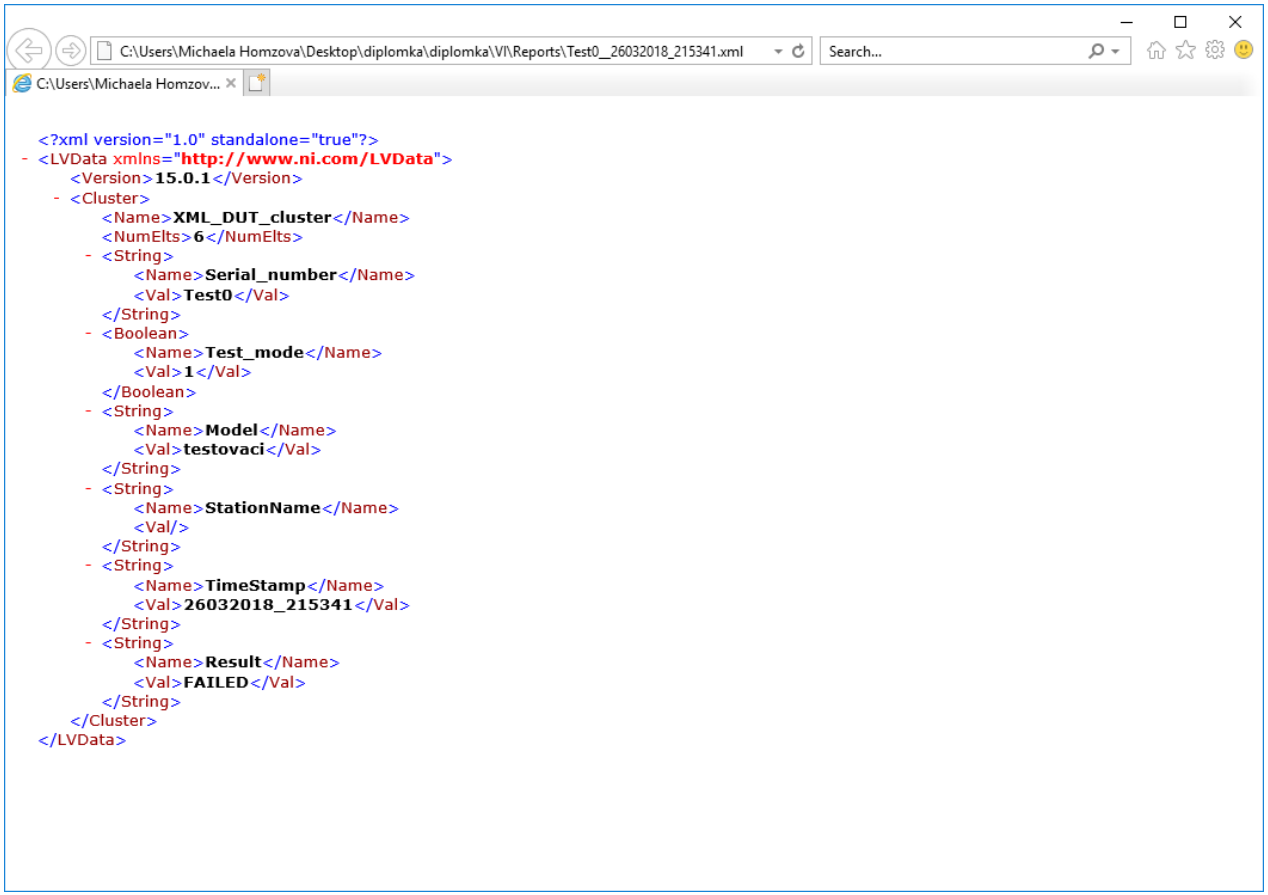
6.1.5 Událost „DUT End“

Událost „DUT End“ nenese žádné data, ale je signálem, že testování dané jednotky již bylo ukončeno a je možno zadat data na server. Ve VI, které zprostředkovává zápis se jako první projde celé pole výsledků kroků a zjistí se celkový výsledek pro testovaný kus. Pokud je alespoň jeden krok ve stavu „Failed“ a nebo „Error“, je výsledek celého testování na dané jednotce označen tímto výsledkem. V případě, že počet výsledků testování pro danou jednotku neodpovídá počtu kroků v receptu, je celé měření vyhodnoceno jako „Error“. Poté proběhne pokus o zapsání nového záznamu pro DUT na server.

Pokud proběhne úspěšně, jsou do databáze odeslány záznamy pro jeho kroky. Pokud se nepodaří zapsat na server nový záznam do tabulky DUT, znamená to, že bylo ztraceno spojení s databází a je vytvořen XML soubor, který obsahuje výsledky měření a je uložen ve složce „Reports“. VI obsluhující zápis na server má výstup parametr, který indikuje, jestli se proběhl zápis na server. Pokud výsledek indikuje, že data byla zapsána na server, je prohledána složka „Reports“ a pokud v ní existují záznamy, jsou tyto XML načteny, jejich obsah zapsán na server a pokud byl zapsán úspěšně, tak je XML smazáno. Toto zajišťuje, že je možno testovat zařízení i po ztrátě spojení s databází a při jeho obnovení jsou data nahrány a nedochází k jejich ztrátě.

6.1.6 Událost „Close“

Při ukončení aplikace z jakéhokoliv důvodu uvedeného výše, včetně zapsání všech vygenerovaných dat, se přechází k obsluhy události „Close“. Zde dojde k uzavření spojení s databází a uložení aktuálního stavu počítadel do konfiguračního souboru stanice. Poté se testovací aplikace ukončí.



Obr. 6.5: Pohled na strukturu XML pro uložení výsledků měření.

7 ZÁVĚR

Cílem této diplomové práce bylo vytvořit systém pro uchovávání výsledků testování výrobků na testovacích stanicích, které jsou součástí výrobní linky. V průběhu této práce byl proveden průzkum software používaného na testovacích stanicích jak můžeme vidět v kapitole 1. Ve druhé kapitole je popsána struktura testovacích předpisů a jejich způsob ukládání v současnosti. Na základě těchto předpisů byl vytvořen datový a procesní model a vybrán vhodný databázový stroj vzhledem k cílovým zákazníkům a použité technologii. Toto popisuje kapitola třetí. Ve čtvrté kapitole je popsán použitý software a stanoveny požadavky na aplikace. Na základě těchto požadavků bylo vytvořeno uživatelské rozhraní pro nastavení metadat a testovacích předpisů. Toto je popsáno v páté kapitole. V šesté kapitole je popsáno vytvoření testovací aplikace a ověření funkčnosti celého řetězce.

Ověření funkčnosti proběhlo následovně: Pomocí uživatelského rozhraní byly zadány kompletní předpisy pro dvě testovací stanice. Tyto předpisy jsou v tabulkách 2.1 a 2.2. Poté úspěšně proběhla inicializace testovací stanice pomocí testovací aplikace a následovně bylo nahráno přes 400 testovacích záznamů ve stavu „Passed“, „Failed“ a „Error“. Tyto záznamy jsou přístupné přes modul „Results“. Díky použití architektury „Producer - consumer“ a fronty bylo otestováno odesílání naměřených hodnot se zpožděním 50 ms, 20 ms a také bez uměle vytvořeného zpoždění a nedošlo ke žádné ztrátě dat. Dále je ošetřen případ, kdy dojde během výroby z náhlému výpadku spojení s databází. Naměřená data jsou uložena ve formátů XML a při obnovení spojení s databází nahrána zpět. Testovací aplikace pro zápis do databáze i aplikace uživatelského prostředí jsou vyhodnoceny jako úspěšné.

Tato diplomová práce bude dále sloužit jako základní kostra pro informační systém v zadavatelské firmě. Dalšími kroky bude přidání nové prezenční vrstvy, kdy se na stávající LabVIEW uživatelské moduly přidá rozhraní s moderním designem, zabezpečení komunikace s databází anebo přiřazení testovací stanice pouze konkrétním uživatelům.

LITERATURA

[1] *OPPEL, Andy. Databáze bez předchozích znalostí. Brno: Computer Press, 2006. ISBN 80- 251- 1199- 7.*

[2] *SQL vs. NoSQL Databases: What's the Difference? Upwork [online]. [cit. 2017-12-10]. Dostupné z: <https://www.upwork.com/hiring/data/sql-vs-nosql-databases-whats-the-difference/>.*

[3] *JSON. In: MongoDB [online]. MongoDB, 2017 [cit. 2017-12-10]. Dostupné z: <https://www.mongodb.com/json-and-bson>.*

[4] *TestExec SL Software. Keysight Technologies [online]. 2017 [cit. 2017-12-10]. Dostupné z: <https://www.keysight.com/en/pd-359651-pn-E2011GC/testexec-sl-software?cc=US&lc=eng>.*

[5] *Automation Desk - dSpace. DSpace [online]. 2017 [cit. 2017-12-25]. Dostupné z: <https://www.dspace.com/en/pub/home/products/sw/test_automation_software/automdesk.cfm>.*

[6] *Test Automation Software [online]. In: . dSPACE, 2017, s. 36 [cit. 2017-12-25]. Dostupné z: <https://www.dspace.com>.*

[7] *Use Measurement Studio With TestStand Test Management Software. National Instruments [online]. 2017 [cit. 2017-12-25]. Dostupné z: <http://www.ni.com/mstudio/whatis/useteststand/>.*

[8] *Tip #7 – How to include comments in SQL script. Vertabelo - design your database [online]. [cit. 2017-12-27]. Dostupné z: <http://www.vertabelo.com/blogdocumentation/database-modeling-tip-7-how-to-include-comments-in-sql-script>.*

[9] *What is LabVIEW? Electronics Notes [online]. [cit. 2018-01-01]. Dostupné z: <https://www.electronics-notes.com/articles/test-methods/labview/what-is-labview.php>.*

[10] *LabVIEW Core 1 Participant Guide. Druhé vydání. 2014. ISBN Part Number 326292A-01.*

[11] *Lesson 4: Connecting to Databases. National Instruments.*

[12] *What is NI TestStand? National Instruments [online]. [cit. 2018-01-01]. Dostupné z: <http://www.ni.com/teststand/whatis/>.*

[13] *LabVIEW: Database Connectivity Toolkit User Manual [online]. 371525A-01. National Instruments Corporation, 2008 [cit. 2018-04-22].*

[14] *LabVIEWTM Core 3 Course Manual: Part Number 375510D-01 [online]. National Instruments, 2014, , 143 [cit. 2018-04-28].*

[15] *SMITH, Allen C. a Stephen R. MERCER. Using the Actor Framework in LabVIEW [online]. s. 9 [cit. 2018-04-28]. Dostupné z: <https://forums.ni.com/t5/Actor-Framework-Documents/READ-THIS-FIRST-to-get-started-with-Actor-Framework/ta-p/3493762>.*

[16] *HALVORSEN, Hans-Petter. Introduction to LabVIEW [online]. University College of Southeast Norway, 2016, 2016-09-07, , 116 [cit. 2018-04-28]. Dostupné z: <http://home.hit.no/~hansha/documents/labview/training/Introduction%20to%20LabVIEW/Introduction%20to%20LabVIEW.pdf>.*

[17] *Float and real (Transact-SQL). Technical documentation, API, and code examples [online]. 07-22-2017 [cit. 2018-04-28]. Dostupné z: <https://docs.microsoft.com/en-us/sql/t-sql/data-types/float-and-real-transact-sql?view=sql-server-2017>.*

[18] *JKI Right-Click Framework for LabVIEW. Autosize Column Width of Table, MultiColumnListbox, and Tree - Plugin Right-Click Framework for LabVIEW - Discussion Forums - National Instruments [online]. [cit. 2018-04-28]. Dostupné z: <https://forums.ni.com/t5/JKI-Right-Click-Framework-for/Autosize-Column-Width-of-Table-MultiColumnListbox-and-Tree/gpm-p/3502814>.*

[19] *Calculating the MD5 Message-Digest of a String or File [online]. 2006 [cit. 2018-05-07]. Dostupné z: <http://www.ni.com/example/25424/en/>.*

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

CSV	Comma - separated values - hodnoty oddělené čárkami
DUT	Device under test - testovaný kus
FIFO	Fronta typu první dovnitř, první ven
fixtura	Fixtura je mechanické zařízení, sloužící k nakontaktování osazených desek plošných spojů v požadovaných bodech, aby bylo možné desku rychle a přesně otestovat.
HEX File	Soubor obsahující data zapsaná ve formátu oktetů pomocí čísel v hexadekadické soustavě
HTML	HyperText Markup Language - hypertextový jazyk
LV	LabVIEW
MS	Microsoft
NI	National Instruments
PCB	Deska plošných spojů
SQL	Structured query language - structurovaný databázový jazyk
TXT	Prostý textový soubor
UML	Unified Modeling Language - univerzální modelovací jazyk
XML	Extensible Markup Language - rozšiřitelný značkovací jazyk

SEZNAM PŘÍLOH

A DATOVÝ MODEL

Dvojlist formátu A3 obsahující vytvořený datový model systému.

B OBSAH PŘILOŽENÉHO CD

- build – Složka obsahující build aplikace a jeho závislé soubory. Aplikaci je třeba spouštět z této složky.
- Homzova_2018.pfd – Elektronická verze práce.
- VI – Složka obsahující zdrojové soubory. Ty je možno spustit ve vývojovém prostředí LabVIEW verze 2015 a novější.
- LVRTE2015_f3Patchstd.exe – Run – time engine, který je potřeba pro spuštění buildu. Verze LabVIEW 2015, 32 – bitů.
- procesnimodel.png – Procesní model systému.
- datovymodel.png – Datový model systému.