**ЗМІСТ**

[ВСТУП 7](#_Toc484796265)

[1 ОГЛЯД ТА АНАЛІЗ ОБЛАСТІ ДОСЛІДЖЕННЯ 9](#_Toc484796266)

[1.1 Постановка проблеми 9](#_Toc484796267)

[1.2 Існуючі рішення 10](#_Toc484796268)

[1.3 Мета роботи та постановка задачі 15](#_Toc484796269)

[2 ОПИС ВЛАСНОГО ВИРІШЕННЯ ПРОБЛЕМИ 16](#_Toc484796270)

[2.1 Функціональне призначення 16](#_Toc484796271)

[2.2 Зберігання данних 17](#_Toc484796272)

[2.3 Обґрунтування технології програмування 18](#_Toc484796273)

[2.3.1 Інструментальні засоби розробки програмного продукту 19](#_Toc484796274)

[2.3.2 Мова програмування 20](#_Toc484796275)

[2.4 Додаткові технології 21](#_Toc484796276)

[2.4.1 WPF 21](#_Toc484796277)

[2.4.2 XAML 23](#_Toc484796278)

[2.4.3 .NetCore 24](#_Toc484796279)

[2.4.4 JSON 26](#_Toc484796280)

[2.5 Системні вимоги 26](#_Toc484796281)

[3 ОПИС РОЗРОБЛЕННОГО ПРОГРАМНОГО ПРОДУКТУ 28](#_Toc484796282)

[3.1 Функціональні можиловсті 28](#_Toc484796283)

[3.3.1 Конструктор 28](#_Toc484796284)

[3.3.2 Сервер 29](#_Toc484796285)

[3.3.3 Переглядач 29](#_Toc484796286)

[3.2Структура данних 30](#_Toc484796287)

[3.3Опис інтерфейсу 32](#_Toc484796288)

[3.3.1 Конструктор 32](#_Toc484796289)

[3.3.2Сервер 37](#_Toc484796290)

[3.3.3 Переглядач 38](#_Toc484796291)

[3.4 Приклад використання 40](#_Toc484796292)

[3.4.1Створення розкаду 41](#_Toc484796293)

[3.4.2 Налаштування серверу 45](#_Toc484796294)

[3.4.3 Використання студентом 46](#_Toc484796295)

[ВИСНОВКИ 48](#_Toc484796296)

[СПИСОК ВИКОРИСТАННИХ ДЖЕРЕЛ 50](#_Toc484796297)

[ДОДАТКИ 52](#_Toc484796298)

[ДОДАТОК А. Настанова користувача 53](#_Toc484796299)

[ДОДАТОК Б. Код програми 62](#_Toc484796300)

# ВСТУП

Багато транспортні, навчальні та виробничі події організовані періодичним способом, повторюючись в один і той же час через добу, тиждень, інше ціле число діб. Подібна організація дозволяє зменшити витрати на планування.

Розклад - вид календаря (тобто, впорядкованість за часом), для якого вказана інформація про майбутні (планованих або потім відбулися) події. Оформляється зазвичай у вигляді таблиці. Оптимізація розкладу відноситься до дослідження операцій. Розклади служать для вказівки того, де і що (хто) має перебувати в зазначені моменти часу[1].

За наявної в сучасних умовах можливості інноваційної діяльності, створення авторських програм, зміни набору предметів і введення нових дисциплін, що включаються в навчальний план освітньої установи, а також зміни кількості годин на вивчення традиційних курсів за рахунок годин регіонального та шкільного компонентів зразкового базисного навчального плану виникає небезпека перевантаження учнів. У зв'язку з цим виникає проблема пошуку науково обґрунтованих шляхів і засобів підвищення ефективності навчально-виховного процесу, проблема оптимізації навчальних занять. Практичне здійснення цього принципу залежить від цілого ряду дидактичних, методичних, психолого-педагогічних, санітарно-гігієнічних, медичних та інших умов.

Важливим елементом організації роботи навчального закладу є науковий підхід до складання розкладу занять. Раціонально складений розклад сприяє ефективності навчально-виховного процесу, зниження і ліквідації перевантажень учнів, підвищенню працездатності учнів і педагогів, оптимальному використанню навчальних приміщень освітнього закладу. Таким чином, розклад занять в освітній установі є одним з найважливіших і діючих видів планування навчально-виховної роботи, основним організаційним документом, учнівського та педагогічного колективу, всієї установи в цілому. Воно встановлює розпорядок занять протягом дня, тижня, навчального року. На його основі організовується позакласна і позашкільна робота, чергування адміністрації та вчителів, робота бібліотеки і проведення класних і загальношкільних батьківських зборів, діяльність учнівських організацій та зв'язок з громадськістю, робота гуртків і секцій додаткової освіти. Для створення оптимальних умов діяльності педагогічного та учнівського колективів і ефективності роботи освітнього закладу необхідно враховувати вимоги до санітарно-гігієнічному режиму освітнього процесу і складання розкладу[2].

Проблема складання розкладу полягає у встановленні послідовності зустрічей викладачів і [студентів](https://uk.wikipedia.org/wiki/%D0%A1%D1%82%D1%83%D0%B4%D0%B5%D0%BD%D1%82) у заздалегідь заданий проміжок часу (як правило, протягом тижня), з урахуванням задоволення низки обмежень різного характеру.

Складання розкладу вручну, зазвичай, вимагає дуже багато часу. При цьому, в зв'язку з надзвичайною складністю урахування всіх обмежень, остаточне рішення може виявитися незадовільним. Тому велику увагу приділяють [автоматизації](https://uk.wikipedia.org/wiki/%D0%90%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82%D0%B8%D0%B7%D0%B0%D1%86%D1%96%D1%8F) складання розкладу. В літературі пропонується значна кількість різних варіантів вирішення проблеми, що відрізняються один від одного типами навчальних закладів і видами обмежень. Основною задачею при складанні розкладу занять є планування і забезпечення методично правильного процесу вивчення усіх навчальних дисциплін навчального плану: їх взаємозв'язки, правильна послідовність і чергування усіх форм навчальної роботи з дисциплін на базі врахування можливостей студентів зі сприйняття і переробки навчальної інформації [3].

Тому метою даної роботи є розробити рішення для навчальних закладів, що до поліпшення створення та поширення розкладу серед студентів.

# 1 ОГЛЯД ТА АНАЛІЗ ОБЛАСТІ ДОСЛІДЖЕННЯ

1.1 Постановка проблеми

Завдання планування розкладу навчальних занять - це завдання на складання розкладу комбінаторного типу, характерною особливістю якої є величезна розмірність і наявність великого числа обмежень складної форми. Фактично, в даний час, не існує універсальних методів вирішення таких завдань. Математична (класична) теорія розкладів охоплює лише вузьке коло добре формалізуються проблем, які зазвичай зводяться до задач комівояжера, транспортної і.т.д. Пряме застосування даних методів до задачі складання розкладу навчальних занять не представляється можливим. Проте є ряд евристичних і переборних методів, які цілком піддаються програмуванню.

Є думка, що досвідчений диспетчер зможе скласти розклад так, що воно буде відповідати інтересам навчального процесу та суспільного життя освітньої установи. Однак з цим не можна погодитися. Ручне рішення задачі складання розкладу занять вимагає великих витрат часу, кваліфікованих фахівців, в той же час результат такого рішення часто виходить далеко не оптимальним. Після введення вихідної інформації потрібно її узгодження, в той час як неможливість отримання необхідного розкладу може бути визначена ще на етапі аналізу.

Під час складання розкладу можливе виникнення тупикових ситуацій. Все це вимагає зміни вихідних даних і ослаблення обмежень, і тут без людини не обійтися. Без внесення даних змін розклад не матиме практичної цінності. Також слід врахувати той момент, що розклад може змінюватися і під час його використання, тобто після складання, і тут дуже важливим є людський фактор. В цьому плані важлива підтримка даного процесу автоматизованими методами і процедурами. Основна перевага полягає в тому, що автоматизоване складання усуває масу рутинної роботи, такий як: пошук можливих варіантів внесення чергових елементів в розклад, перевірку виконання вимог, пошук випадкових помилок в готовому розкладі, оформлення розкладу на папері у вигляді різних таблиць (для викладачів, груп , покабінетного), залишаючи людині більше часу на більш інтелектуальні дії. Комп'ютер в даному випадку також є інструментом, істотно підсилює здібності людини, тому що людина не в змозі перебрати і проаналізувати таку ж кількість варіантів розкладів, як комп'ютер[4].

1.2 Існуючі рішення

На сьогоднішній день існує ряд програм реалізують технологію "Розклад". Пропонується розглянути кілька, на прикладах відомих програм.

Система складання розкладів і обліку навантаження викладачів у ВНЗ "1С: Хронограф Розклад".

Програма "1С: Хронограф Розклад" призначена для автоматизації навчального планування і складання розкладу в окремих підрозділах професійних і вищих навчальних закладів, на різних комерційних і некомерційних навчальних курсах (комп'ютерних, вивчення іноземних

"1С: Хронограф Розклад" надає можливість:

1)підготувати дані про період навчання, з урахуванням специфіки організації навчальної діяльності конкретного освітнього закладу на основі:

* завдання навчального року;
* розбиття навчального року на періоди навчального планування;
* завдання навчальних періодів в рамках одного періоду навчального планування з метою деталізації термінів викладання
* навчальних курсів, навчання навчальних груп та роботи викладачів;
* автоматичного заповнення списку свят;
* завдання неробочих тижнів і канікул;
* автоматичного формування списку тижнів, що припадають на обраний період навчального планування із зазначенням ознаки парності / непарності і кількості робочих днів.

2)планувати навчальну діяльність всього навчального закладу або окремих підрозділів (факультетів, відділень, кафедр і т.п.), використовуючи:

* організацію і завдання структури рівнів навчання з можливістю диференціації їх на підрівні;
* формування переліку навчальних курсів (предметів, дисциплін, тим і т.п.) із зазначенням термінів їх викладання і можливістю
* перегляду інформації про навчальні тижнях, що входять в заданий відрізок часу;
* створення переліку навчальних груп із зазначенням чисельності, спеціалізації і рівня навчання;
* поділ навчальних груп на необхідну кількість підгруп в рамках обраного навчального курсу;
* завдання ідентичних підгруп в рамках однієї групи з кількох навчальних курсів;
* створення потокових об'єднань груп / підгруп за обраним навчальним курсом;
* завдання навантаження навчальним групам / підгрупах на весь період навчального планування;
* формування списку викладачів;
* розподіл годин навчального курсу в рамках навчальної групи / підгрупи по декільком викладачам на різні відрізки часу в межах заданого періоду навчального планування;
* автоматичне планування навантаження викладача на задану послідовність тижнів з можливістю ручного редагування.

3)поставити графіки роботи викладачів, учнів і кабінетів на основі:

* призначення конкретних неробочих годин і днів для викладачів, навчальних груп та кабінетів;
* завдання максимально можливої ​​кількості робочих днів викладача для кожного тижня поточного періоду навчального планування;
* автоматичного копіювання графіка роботи обраного викладача / групи / кабінету з одного тижня на задану послідовність тижнів.

4)створити методично витримане розклад навчальних занять на основі:

* потижневого планування занять конкретного викладача безпосередньо при складанні розкладу;
* призначення занять викладачам в режимі обраної групи на поточний тиждень;
* можливості копіювання розкладу обраного викладача або навчальної групи з поточного тижня на задану послідовність тижнів в рамках навчального періоду;
* використання ефективного алгоритму "Попереднього розрахунку" при складанні "чорнових" варіантів розкладу;
* застосування інтерактивного "Автоматичного розрахунку" для складання "остаточного варіанту" розкладу на поточний тиждень для будь-якої кількості навчальних груп, з можливостями подальшого ручного редагування;

Програма "Avtor" (АВТОРаспісаніе).

Система "АВТОРаспісаніе" призначена для швидкого, зручного і якісного складання розкладів занять і супроводу їх протягом всього навчального року.

Є вісім основних модифікацій програми для різних навчальних закладів для середніх загальноосвітніх шкіл, ліцеїв і гімназій, коледжів, технікумів та професійних училищ, училищ мистецтва і культури, для вузів.

AVTOR допомагає максимально полегшити і автоматизувати складна праця укладачів розкладу.

Система допомагає легко будувати, корегувати і роздруковувати у вигляді зручних і наочних документів:

* розкладу занять класів (навчальних груп);
* розкладу викладачів;
* розклад зайнятості аудиторій (кабінетів);
* навчальні навантаження.

Час роботи програми залежить від розмірності навчального закладу і потужності комп'ютера. Повний розрахунок і оптимізація розкладу школи середнього розміру зі складними вихідними даними (40 класів, 80 викладачів, з них більше 10 сумісників; дві зміни; дефіцит аудиторій) йде близько 2-3 хвилин на комп'ютері типу Celeron-2000.

AVTOR дозволяє:

* будувати розклад без "вікон" у класів (навчальних груп);
* оптимізувати в розкладі "вікна" викладачів;
* враховувати необхідний діапазон днів / годин для класів, для викладачів і для аудиторій;
* враховувати характер роботи і побажання, як штатних співробітників, так і сумісників-погодинників;
* оптимально розміщувати заняття по кабінетах (аудиторіям) з урахуванням особливостей класів, предметів, пріоритетів викладачів і місткості кабінетів;
* вводити розклад дзвінків;
* встановлювати час переходу (переїзду) між навчальними корпусами;
* оптимізувати кількість переходів з кабінету в кабінет, і з корпусу в корпус;
* легко з'єднувати будь-які класи (навчальні груп) в потоки при проведенні будь-яких занять;
* розділяти класи (навчальні групи) при проведенні занять з іноземної мови, фізичної культури, праці, інформатики (і будь-яким іншим предметам) на будь-яку кількість підгруп (до десяти!);
* вводити комбіновані заняття для підгруп (типу "іноземний / інформатика") по будь-яких предметів;
* вводити (крім основних предметів) спецкурси і факультативи;
* оптимізувати рівномірність і трудомісткість розкладу;
* легко і швидко вводити і коректувати вихідні дані;
* мати будь-яку кількість варіантів розкладів;
* автоматично перетворювати розкладу при зміні бази даних;
* легко зберігати в архівах, копіювати і пересилати по E-mail повні бази даних і варіанти розкладів (обсяг архіву повної бази розкладу середньої школи - 10-30 K, великого ВНЗ - 50-70K);
* швидко вносити будь-які необхідні корективи в розклад;
* знаходити заміни тимчасово відсутніх викладачів;
* автоматично контролювати розклад, виключаючи будь-які "накладки" і протиріччя;
* виводити розкладу у вигляді зручних і наочних документів: текстових, Word, HTML, а також файлів dBaseі книг Excel;
* виставляти готові розкладу в локальній мережі і на Інтернет-сторінках для загального доступу.

Такі програми дуже дорогі, зроблені для великих підприємств або складні в зверненні. Так само для ведення таких програм потрібна підтримка, яка вимагає великих матеріальних витрат, висновок спеціальних договорів про супроводі і необхідність утримувати цілий штат співробітників, які будуть займатися її підтримкою. Було прийнято рішення створити оригінальну програму легшу в експлуатації з більш простим функціоналом.

1.3 Мета роботи та постановка задачі

Метою роботи є поліпшення використання розкладу у навчальних закладах, використанням інформаційних обчислювальних технологій та Інтернету.

Після проведення аналізу існуючих програмних продуктів щодо розробки та поширення розкладку, принятно рішення про створення власного програмного засобу. Тому метою даної роботи розробити програмний засіб для електронного створення та поширення розкладу. Доосновних можливостей програмного засобу має входити наступний функціонал:

* створення розкладу для груп;
* можливість поширення створених розкладів;

# 2 ОПИС ВЛАСНОГО ВИРІШЕННЯ ПРОБЛЕМИ

2.1 Функціональне призначення

Розглядаеться розробка системи із програм, яка буде надавати саме ті функції які необхідні певному користивачу.На рисунку1 зображен концепт схеми роботи інформаційної системи «Розклад для навчальних закладів».

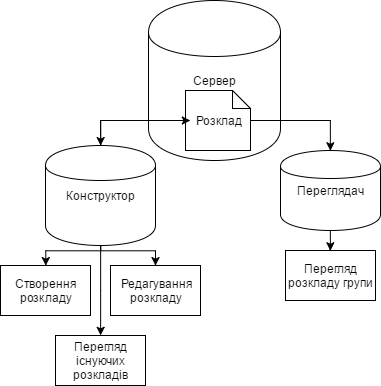


Рисунок 1 - Концепт схеми роботи інформаційної системи «Розклад для навчальних закладів».

Для реалізації постановленного завдання, пропоную розробити програмне забезпечення «Конструктор розкладів». Головною метою якого буде створення,редагування розкладів для груп. Цей додаток необхідно буде поширити серед деканату, або іншого органу, що відповідає за розклад.

Для поширення розкладу серед студентів необхідно реалізувати додаток-сервер, який буде поширювати розклад для студентів. Цей додаток має бути встановленний лише на один ПК учбового закладу.

«Переглядач розкладу» має передбачати із себе схожий за функціоналом «Конструктор розкладів», але з відстунім функціоналом редагування. Цей додаток також повинен працювати лише з однією групой, отримувати інформацію у фоновому режимі та повідомлювати користувача про зміни. Цей додаток слід пошири серед студентів.

2.2 Зберігання данних

Для зберігання данних данних розглядаеться використовувати технолгію JSON. Ця технологія дозволяє швидко створити текстову строку із єкзампляру обєкту. Наприклад:

Product product = new Product();

product.Name = "Apple";

product.Expiry = new DateTime(2008, 12, 28);

product.Sizes = new string[] { "Small" };

У цьому прикладі ми створили обєкт Продукт. Імя продукту «Apple» , термін придатності 28.12.2008, розмір продукту : «small». На виході отримаємо наступну jsonстроку.

// {

//   "Name": "Apple",

//   "Expiry": "2008-12-28T00:00:00",

//   "Sizes": [

//     "Small"

//   ]

// }

Маючи інформацію про обєкт у текстовоми виді ми можемо зберігти у файловій системі, підготовити для відправки на сервер. Розібрати строку знову до обєкту також просто та не потребує велик зусиль, або обчислювальної потужності.

У інформаційної системи «Розклад для навчальних закладів» планується зберігання остання використаного розкладу у файловій системі. Ця функція забезпечить скорочення кількості запитів с сервера, або відкривання данних с файлової системи.

Для захисту данних вся jsonстрока повинна буде зашифрована. Тим самим це передбачить самостійну зміну даних користувачем у файловій системі. Тому що внесені змінни можуть привести до аварійної роботи програми. Наприклад користувач міг би змінити інтервал час більш ніж 24 години в добі, це привело б до аварійного завершення. А якщо користувач буду намагатися змінити зашифрований файл, конвертер JSON побачить помилку в архітектурі, і інформаційна системи «Розклад для навчальних закладів» не завантажить вибранний файл.

## 2.3 Обґрунтування технології програмування

Для вирішення поставленої задачі було обрано використовувати парадигму програмування – Об'є́ктно-орієнто́ване програмува́ння.

Об'є́ктно-орієнто́ване програмува́ння (ООП) — одна з парадигм програмування, яка розглядає програму як множину «об'єктів», що взаємодіють між собою. Основу ООП складають три основні концепції: інкапсуляція, успадкування та поліморфізм. Одною з переваг ООП є краща модульність програмного забезпечення (тисячу функцій процедурної мови, в ООП можна замінити кількома десятками класів із своїми методами).

На відміну від традиційних поглядів, коли програму розглядали як набір підпрограм, або як перелік інструкцій комп'ютеру, ООП програми можна вважати сукупністю об'єктів. Відповідно до парадигми об'єктно-орієнтованого програмування, кожний об'єкт здатний отримувати повідомлення, обробляти дані, та надсилати повідомлення іншим об'єктам. Кожен об'єкт — своєрідний незалежний автомат з окремим призначенням та відповідальністю

Поліморфізм дозволяє керувати багатьма об’єктами які можуть відрізнятися один від одного, але які мають одну роль. Тим самим зберігаючи час на розробку та роблячи програмний код більш стабільним до помилок, та легких для внесення корегувань.

### 2.3.1 Інструментальні засоби розробки програмного продукту

Для досягнення поставленої мети найкращим вибором було використовувати візуальну середу Microsoft Visual Studio 2017.

Microsoft Visual Studio — серія продуктів фірми Майкрософт, які включають інтегроване середовище розробки програмного забезпечення та ряд інших інструментальних засобів. Ці продукти дозволяють розробляти як консольні програми, так і програми з графічним інтерфейсом, в тому числі з підтримкою технології Windows Forms, а також веб-сайти, веб-застосунки, веб-служби як в рідному, так і в керованому кодах для всіх платформ, що підтримуються Microsoft Windows, Windows Mobile, Windows Phone, Windows CE, .NET Framework, .NET Compact Framework та Microsoft Silverlight. [5]

Додатки Windows, створені за допомогою класів .NET Framework, мають і інші переваги. Можна отримати доступ до служб операційної системи і скористатися іншими перевагами, наданими користувальницької обчислювальним середовищем. Доступ до даних можна отримати за допомогою ADO.NET. GDI + надає додаткові можливості для малювання та розмальовки в додатках Windows Forms. Додатки Windows можуть викликати методи, які звичайно пропонуються службами XML, даючи можливість скористатися відомостями і комп'ютерними ресурсами з багатьох джерел і партнерів.

Як і у випадку з іншими додатками .NET Framework, можна створювати додатки Windows в текстовому редакторі, викликати методи і класи .NET Framework, компілювати додаток в командному рядку і поширювати отриману в результаті виконувану програму. Маючим досвід роботи з Visual Studio ці функції покажуться знайомими ; вони аналогічні функціям, доступним в більш ранніх версіях Visual Basic і Visual C ++. У Visual Studio 2005 ці можливості розширені і утворюють широкий набір засобів для розробки додатків Windows[6].

2.3.2 Мова програмування

C# (вимовляється Сі-шарп, також відомий як ЦЭ-решітка) — об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET. Розроблена Андерсом Гейлсбергом, Скотом Вілтамутом та Пітером Гольде під егідою Microsoft Research (при фірмі Microsoft).

Синтаксис C# близький до С++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML. Перейнявши багато що від своїх попередників — мов С++, Delphi, Модула і Smalltalk — С#, спираючись на практику їхнього використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад множинне спадкування класів (на відміну від C++)[7].

C# розроблялась як мова програмування прикладного рівня для [CLR](https://uk.wikipedia.org/wiki/CLR) і тому вона залежить, перш за все, від можливостей самої CLR. Це стосується, перш за все, системи типів C#. Присутність або відсутність тих або інших виразних особливостей мови диктується тим, чи може конкретна мовна особливість бути трансльована у відповідні конструкції CLR. Так, з розвитком CLR версії 1.1 до 2.0 значно збагатився і сам C#; подібної взаємодії слід чекати і надалі. (Проте ця закономірність буде порушена з виходом C# 3.0, що є розширеннями мови, що не спираються на розширення платформи .NET.) CLR надає C#, як і всім іншим .NET-орієнтованим мовам, багато можливостей, яких позбавлені «класичні» мови програмування. Наприклад, збірка сміття не реалізована в самому C#, а проводиться CLR для програм, написаних на C# точно так, як і це робиться для програм на VB.NET, J# тощо.

2.4 Додаткові технології

2.4.1 WPF

Windows Presentation Foundation (WPF, кодова назва — Avalon) — графічна (презентаційна) підсистема в складі .NET Framework 3.0, що має пряме відношення до XAML. WPF разом з .NET Framework 3.0 вбудована в Windows Vista, а також доступна для установки в Windows XP Service Pack 2 і Windows Server 2003.

WPFвідкриває великі можливості для проектування власних компонентів, вирівнювання динамічного дизайну тощо, тому саме ця платформа була обрана для проекту.

WPF - перше реальне оновлення технологічного середовища призначеного для користувача інтерфейсу з часу випуску Windows 95. Воно включає нове ядро, яке повинне замінити GDI і GDI+, використовувані на нинішній Windows-платформі. WPF є високорівневим об'єктно-орієнтованим функціональним шаром (англ. framework), що дозволяє створювати двовимірні та тривимірні інтерфейси.

Вікна (window) є основними елементами в будь-якому настільному застосунку — настільки «основними», що в їхню честь навіть була названа операційна система Windows. І хоча в WPF є модель для створення навігаційних застосунків, що розподіляють задачі по окремих сторінках, вікна залишаються переважною технологією для створення застосунків.

Тим, кому доводилось раніше програмувати із застосуванням набору інструментальних засобів Windows Forms, більша частина пропонованого матеріалу здасться знайомою, тому що клас Window, фактично, є вільнішою моделлю класу [Form](http://msdn.microsoft.com/en-us/library/system.windows.forms.form.aspx).

Клас Window успадкований від класу ContentControl. Це означає, що він може містити тільки одного нащадка (яким звичайно є контейнер макета на зразок елемента керування Grid) і що його тло можна зафарбовувати за допомогою кисті шляхом установки властивості Background. Можна ще також використовувати й властивості BorderBrush і BorderThickness для додавання навколо вікна границі, але ця границя додається усередині віконної рамки (тобто по краю клієнтської області). Віконну рамку можна взагалі видаляти і встановлювати на власний розсуд.

Щоб відобразити вікно, необхідно створити екземпляр класу Window і викликати метод Show() або ShowDialog(). Метод ShowDialog() відображає модальне вікно. Модальні вікна не дозволяють користувачеві одержувати доступ до батьківського вікна, блокуючи можливість використання в ньому миші й можливість уведення в ньому яких-небудь даних до тих пір. доки модальне вікно не буде закрито. Також метод ShowDialog() ще й не здійснює повернення до тих пір. доки модальне вікно не буде закрито, так що виконання будь-якого коду, що знаходить після нього, на час відкладається. (Це, однак, не означає, що в цей час не може виконуватися й ніякий інший код — наприклад, при наявності запущеного таймера оброблювач його подій однаково буде працювати). Найбільше часто застосовувана в коді схема виглядає так: відображення модального вікна, очікування його закриття й наступне виконання над його даними якої-небудь операції.

Метод Show() відображає немодальне вікно, що не блокує доступ користувача ні до яких інших вікон. Більше того, метод Show() здійснює повернення відразу ж після відображення вікна, так що наступні після нього в коді оператори виконуються негайно. Можна створювати й показувати відразу кілька немодальних вікон, і користувач може взаємодіяти з усіма ними одночасно. У випадку застосування немодальних вікон іноді потрібен код синхронізації, що гарантує відновлення інформації в другому вікні при внесенні якихось зміні в першому й тим самим виключати імовірність роботи користувача з недійсними даними. [8]

2.4.2 XAML

XAML ([англ.](https://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D1%96%D0%B9%D1%81%D1%8C%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0) eXtensible Application Markup Language) - декларативна [мова розмітки](https://uk.wikipedia.org/wiki/HTML). З точки зору моделі програмування [.NET Framework](https://uk.wikipedia.org/wiki/.NET_Framework) мова XAML спрощує створення [користувацького інтерфейсу](https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BA%D0%BE%D1%80%D0%B8%D1%81%D1%82%D1%83%D0%B2%D0%B0%D1%87%D0%B0) для програми .NET Framework.

Дизайн у WPF реалізується за допомогою розмітки XAML. Можна створити видимі елементи інтерфейсу користувача в декларативній розмітці XAML, а потім відокремити визначення користувача інтерфейсу від логіки часу виконання, використовуючи файли коду програмної частини, приєднані до розмітки за допомогою визначень поділюваних [класів](https://uk.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)). Мова XAML безпосередньо представляє створення екземплярів об'єктів в конкретному наборі резервних типів, визначених у збірках. У цьому полягає її відмінність від більшості інших мов розмітки, які, як правило, є інтерпретованими мовами без прямого зв'язку з системою резервних типів. Мова XAML забезпечує робочий процес, що дозволяє декільком учасникам розробляти користувацький інтерфейс і логіку програми, використовуючи потенційно різні засоби.

При поданні у вигляді тексту файли XAML є XML-файлами, які зазвичай мають розширення .xaml. Файли можна зберігати в будь-якому кодуванні, що підтримує XML, але зазвичай використовується кодування [UTF-8](https://uk.wikipedia.org/wiki/UTF-8). [9]

2.4.3 .NetCore

.NET Core - це практично повне перезавантаження стека .NET Framework. З нової платформи з різних причин був виключений ряд технологій. Слід розуміти, що платформа .NET Core розрахована в першу чергу на розробку для серверних і хмарних рішень. Для десктопних додатків краще підходять класичний .NET для Windows (з підтримкою WPF і Windows Forms) і Mono для Linux і Mac OS X (з підтримкою Windows Forms). Мобільні проекти можна створювати, використовуючи Xamarin. На рисунку 2 показано, як технології розподілені всередині різних реалізацій .NET.

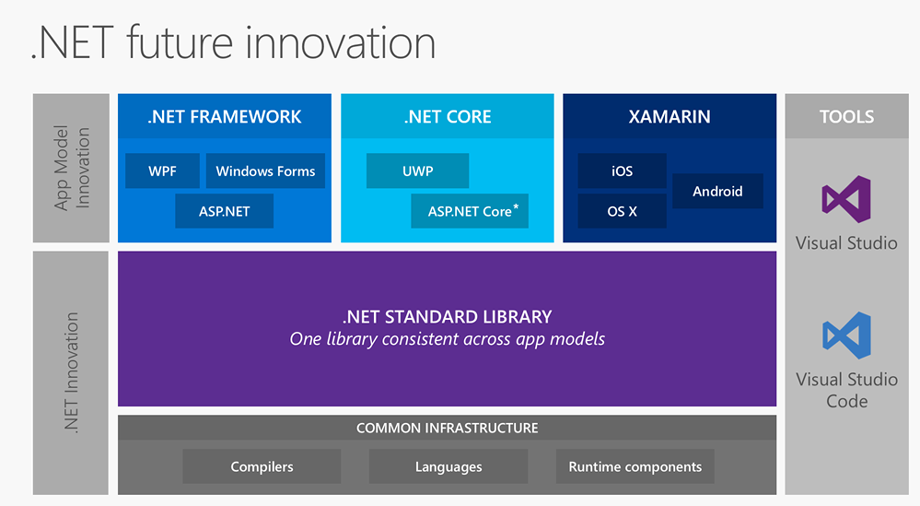


Рисунок 2 - .NETcoreу сімействі .NET

Розглядається варіант розроблення Серверної частини рішення на платформі .NETCore, що дозволяє працювати на будь-якій платформі де є підтримка .NETCore. Здебільшого на платформах Linux, адже на сьогоднішній день більшість серверів працюють саме на цій платформі.

Таким чином, в .NET Core були виключені:

* + ASP.NET WebForms;
  + WCF;
  + WPF;
  + Windows Forms.

Зате інструменти для розробки консольних і веб-додатки отримали новий етап розвитку. При розробці більшість необхідних компонент додатки можуть завантажуватися як окремі модулі через пакетний менеджер NuGet. Це дозволяє зменшити кількість надлишкових залежностей і загальний розмір готового продукту.

Open Source. На відміну від класичного .NET Framework, код якого здебільшого є закритим, код .NET Core повністю відкритий і розповсюджується під ліцензіями MIT і Apache2.

Варто відзначити, що після відкриття коду .NET Core команда, що займається розробкою проекту Mono, оголосила про намір об'єднання кодової бази тих компонентів Mono, які реалізовані в .NET Core.

Робота в хмарі. Як і проект, написаний на класичному .NET, проект на базі .NET Core досить легко перенести в хмару. Microsoft Azure вже підтримує розміщення .NET Core проектів як в службах Application Services, так і на віртуальних машинах.

Бібліотеки для роботи з сервісами Microsoft Azure також вже починають перенести на .NET Core. Наприклад, Windows Azure Storage вже доступна для роботи. Відповідно Azure Storage Services можуть використовуватися в core-проектах.

Також, для проектів .NET Core з'являється можливість розміщення на майданчиках тих хмарних провайдерів, які не забезпечили підтримку Windows оточення, але при цьому мають інші привабливими можливостями. Наприклад, віртуальні сервери Digital Ocean за рахунок використання SSD дисків дуже швидкі, але на даний момент дозволяють розгортати тільки ОС сімейства Linux. Розгорнути проект на .NET Core на подібно сервері не складе труднощів [10].

2.4.4 JSON

У розроблоному проекті використовуеться форматування JSON. Завдяки цій технології вдається типізувати запрограмовані обєкти у текстовий формат для збереження та передачі між додатками.

JSON ([англ.](https://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D1%96%D0%B9%D1%81%D1%8C%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0) JavaScript Object Notation, [укр.](https://uk.wikipedia.org/wiki/%D0%A3%D0%BA%D1%80%D0%B0%D1%97%D0%BD%D1%81%D1%8C%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0) об'єктний запис [JavaScript](https://uk.wikipedia.org/wiki/JavaScript), вимовляється джейсон)—це текстовий[формат](https://uk.wikipedia.org/wiki/%D0%A4%D0%BE%D1%80%D0%BC%D0%B0%D1%82)обміну[даними](https://uk.wikipedia.org/wiki/%D0%94%D0%B0%D0%BD%D1%96)між[комп'ютерами](https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%27%D1%8E%D1%82%D0%B5%D1%80). JSON базується на тексті, може бути прочитаним людиною. Формат дозволяє описувати [об'єкти](https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BA%D1%82_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)) та інші [структури даних](https://uk.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85). Цей формат головним чином використовується для передачі структурованої інформації через [мережу](https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%27%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0_%D0%BC%D0%B5%D1%80%D0%B5%D0%B6%D0%B0) (завдяки процесу, що називають [серіалізацією](https://uk.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D1%96%D0%B0%D0%BB%D1%96%D0%B7%D0%B0%D1%86%D1%96%D1%8F)).

Розробив і популяризував формат [Дуглас Крокфорд](https://uk.wikipedia.org/wiki/%D0%94%D1%83%D0%B3%D0%BB%D0%B0%D1%81_%D0%9A%D1%80%D0%BE%D0%BA%D1%84%D0%BE%D1%80%D0%B4).

JSON знайшов своє головне призначення у написанні веб-програм, а саме при використанні технології [AJAX](https://uk.wikipedia.org/wiki/AJAX). JSON, що використовується в AJAX, виступає як заміна [XML](https://uk.wikipedia.org/wiki/XML) (використовується в [AJAX](https://uk.wikipedia.org/wiki/AJAX)) під час асинхронної передачі структурованої інформації між [клієнтом](https://uk.wikipedia.org/wiki/%D0%9A%D0%BB%D1%96%D1%94%D0%BD%D1%82_(%D1%96%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) та [сервером](https://uk.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%B5%D1%80). При цьому перевагою JSON перед XML є те, що він дозволяє складні структури в атрибутах, займає менше місця і прямо інтерпретується за допомогою [JavaScript](https://uk.wikipedia.org/wiki/JavaScript) в об'єкти [11].

2.5 Системні вимоги

Для додатків «Конструктор» та «Переглядач» необхідні такі вимоги до системи та обладення на ПК де плануеться використання програмного продукту:

* *Процесор:* Процесор або система на чипі з мінімальною тактовою частотою 1 ГГц.
* *Оперативна пам’ять:* 1 ГБ (для 32-розрядної версії) або 2 ГБ (для 64-розрядної версії ОС).
* *Дисковий простір*: Потрібні 200 МБ
* *Відеокарта*: З підтримкою DirectX 9 або новішої версії з драйвером WDDM 1.0.
* *Роздільна здатність*: не менша за 800x600.
* *Операційна Система*: WindowsXP (починаючи з редакції Home), Windows 7, Windows 8, Windows 10
* *Програмне забезпечення*: .NetFramework 4 або новіше.

Серверна частина працює на .NETCoreта може працювати усюди де є можливість встановити .NETCore.

Наступні вимоги до систем ОС Windows:

* Windows 7 SP1;
* Windows 8.1;
* Windows 10;
* Windows Server 2008 R2 SP1 (Full Server або Server Core);
* Windows Server 2012 SP1 (Full Server або Server Core);
* Windows Server 2012 R2 SP1 (Full Server або Server Core);
* Windows Server 2016 (Full Server, Server Core або Nano Server);

Вимоги дo macOS:

* macOS 10.12 "Sierra";
* macOS 10.11 "El Capitan";

Вимоги до систем Linux:

* [Red Hat Enterprise Linux 7 Server](https://www.microsoft.com/net/core#linuxredhat);
* [Ubuntu 14.04, 16.04 и Linux Mint 17](https://www.microsoft.com/net/core#linuxubuntu);
* [Debian 8.2](https://www.microsoft.com/net/core#linuxdebian);
* [Fedora 23](https://www.microsoft.com/net/core#linuxfedora);
* [CentOS 7.1 и Oracle Linux 7.1](https://www.microsoft.com/net/core#linuxcentos);
* [OpenSUSE 13.2](https://www.microsoft.com/net/core#linuxopensuse).

# 3 ОПИС РОЗРОБЛЕННОГО ПРОГРАМНОГО ПРОДУКТУ

3.1 Функціональні можиловсті

Програмний поділяеться на три частини:

* «Сервер»;
* «Конструктор»;
* «Переглядач».

Розглянемо роль кожного у наступних пунктах.

3.3.1 Конструктор

Основною задачею програми «Конструктор» є створення розкладу для груп. Оператор ПК створює групу, заповню список вчителів, аудиторії, часові інтервали дзвінків. І тоді лиш, він має можливість назначити лекцію на конкретний час у певну неділю. Неділі в свою чергу можут також поділятися на чисельник та знаменик.

Кожна група має свій незалежний розклад, але має спільний доступ до списку вчителів, аудиторіій тощо.

Саме можливість вибору даних що повторються можут дійсно зменшити обсяг часу на розробку розкладу. Для прикладу Фізику завжди проводять в 404 аудиторії, і лектор в ВУЗі лише один. Оператору не потрібно буде кожен раз набирати ПІБ лектора та номер аудиторії.

Коли розклад готовий, оператор зберігає його у файлову систему, або відправляє на сервер.

Вже існуючий розклад можливо завантажити із файлової системи, або скачати з сервера.

3.3.2 Сервер

Сервер представляє із себе центром для поширення розкладу. Основна задача серверу отримувати нові версії розкладу від конструктору, та відправляти переглядачам та конструкторам по їх запитам.

Для оптимальної роботи сервер має бути ввімкний постійно, адже він обслуговує деяку кількість переглядачів і конструкторів, які можуть ствоюрвати запит у будь-яку годину доби.

В сервері передбаченна роботи с чорним і білим списком. IP-адресси, що знаходяться в чорному списку не мають можливості взаємодіяти с сервером, а їх запити використовують мінімальні обчислювальні ресурси.

IP-адреси с білого списко мають привілегію на зміну розкладу на сервере. Це міра захисту зробленна для того, щоб злоумисник, навіть якщо отримає доступ до конструктора, або вгадає його шифр передачі не мав змогу зіпсувати діючий розклад.

Сервер зберігає останій діючий розклад, записує усі дії у логи. Реалізован у консольному варіанті, що дозволяє бути запущенним навіть у середі без графічного інтерфейсу.

3.3.3 Переглядач

Переглядач представляє із себе видозмінений конструктор з деякими різницями. До цих різниць наступні вилучення:

* можливість редагування данних (лектори, аудиторії, інтервали,пари);
* можливість вибору груп;
* можливість завантаження на сервер розкладу;
* можливість завантаження розкладу с файлової системи.

До доданих функцій можно зазначити «відстежування групи». Користувач після вибору своєї групи «підписуеться» на її оновлення, та буде отримувати повідомлення о змінах до розкладу. Переглядач також має змогу роботи у режимі «трея». При мінімузуванні програми, вона продовжує роботу у треї, не знаходячись на панелі задач. Переглядач будут використовувати студенти, але для роботи переглядача необхідно мати активний та налаштований сервер.

3.2Структура данних

Кожен програмний продукт у системі розклад поділяє між собою одну і туж саму структуру данних. Головною одинцею данних є класс «Розклад», який уміщює у собі всю інформацію яку необхідно передати на інший продукт для незалежної та автономної роботи.

Розглянемо кожно структуру ближче:

* Аудиторія – числовий тип данних. Зберігає номер аудиторії
* Викладач – тип данних в якому, знаходяться такі текстові поля, як : Прізвище, Ім’я, По батькові
* Предмет – текстовий тип данних, визначає лише назву предмету.
* Інтервал – складний тип данних, який повертає часовий інтервал (початок та кінець пари).
* Група – складний тип данних, який містить у собі усю інформацію про групу. До неї входить текстовий комментар, та 2 типи данних «Неділя», що відображають чиселньик та знаменник відповідно.
* Неділя – тип данних, який містить у собі інший тип данних «День». Усього 6 днів неділі (з Понеділка до Суботи).
* День – тип данних, зберігає у собі список лекцій для цього дня.
* Лекція – основна одиниця. Представляє собою набір із наступних данних: аудиторія, викладач, предмет, номер пари та список замін.
* Заміна – тип данних який зберігає інформацію о заміні. Включає у себе дату та назву предмета.

На рисунку 3 візуалізована схема структури даних.

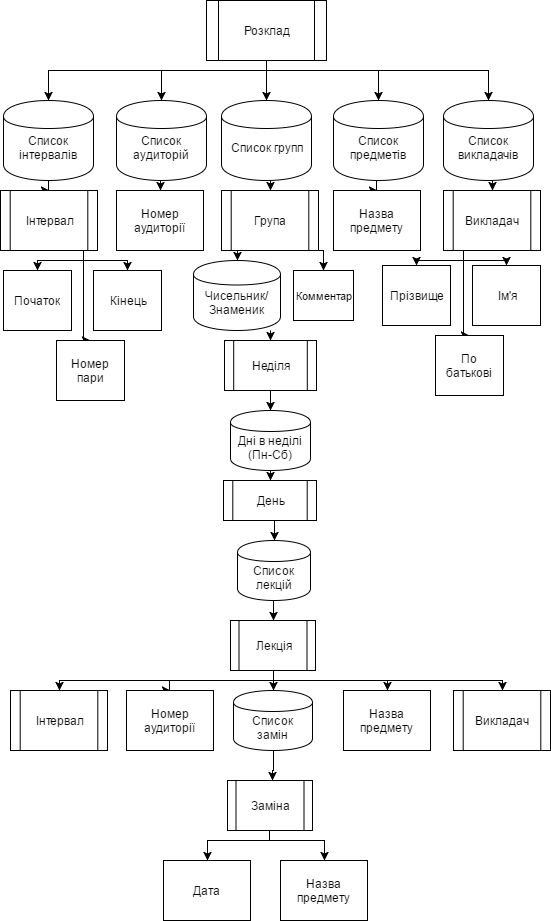


Рисунок 3 - Блок-схема структури даних

3.3Опис інтерфейсу

3.3.1 Конструктор

На рисунку 4 зображен інтерфейс «Конструктора»

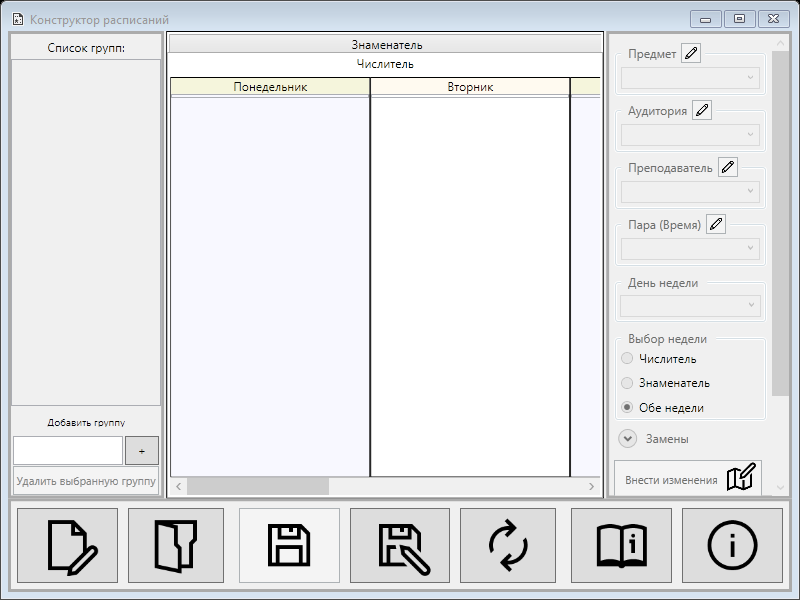


Рисунок 4 - Інтерфейс "Конструктора"

На рисунку 5 зображена панель груп. За допомгою цієї панелі користувач може створювати, видаляти та вибарати групу.

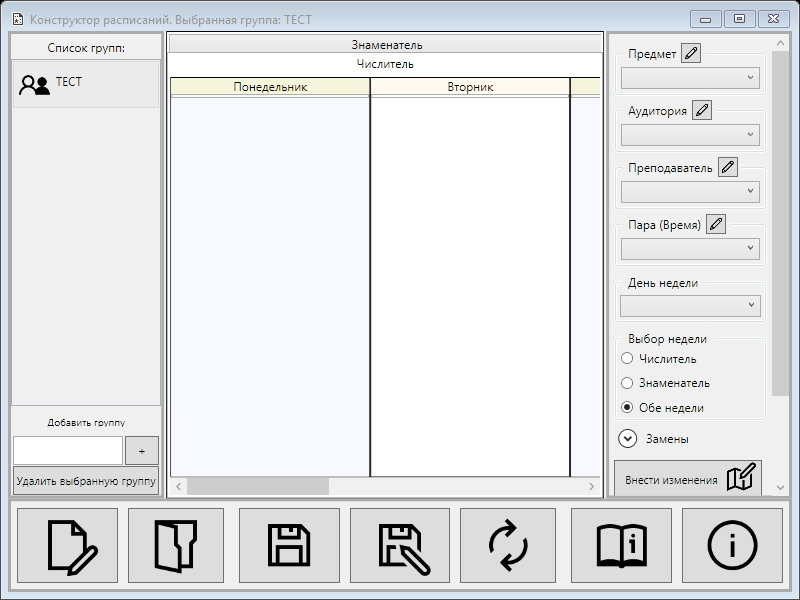


Рисунок 5 – Панель груп

На рисунку 6 зображена функціональна панель. На ній знаходяться велика кількість функціоналу по роботі с розкладом (відкриття розкладу, зберігання, відправка тощо).

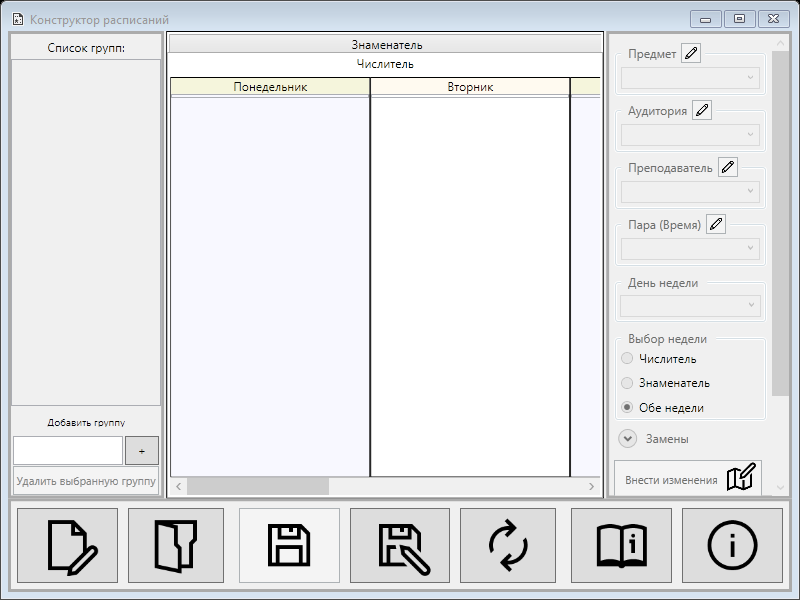


Рисунок6 - Функціональна панель

На рисунку 7 знаходиться панель редагування пари. За допомогою цієї панелі моделюеться пара. Тобто користувач може змінити вже існуючу пару, видалити її, або створити її. Для створення і редагування використовуються саме ті парамтри, що були вказані на цій панелі.

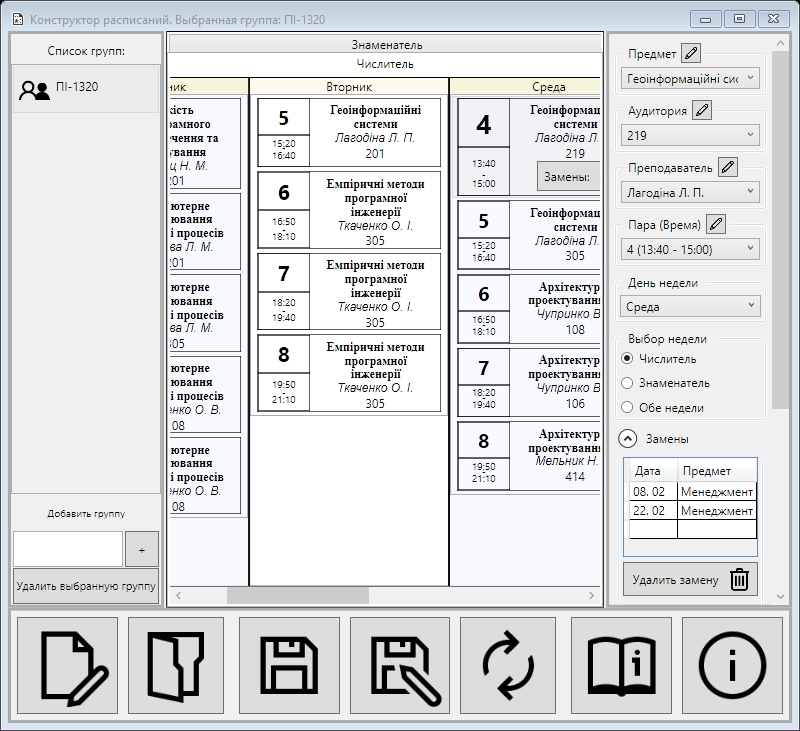


Рисунок 7 - Панель редагування пари

На рисунку 8 розміщенні 4 вікна для редагування додаткових данних. Додаткові дані – це дані які використовуються для створення пар. Наявність цих данних необхідно, для створення пар і тим самим скорочують час на введеня даних накшталк ПІБ лектора для деклільких лекцій впродовж дня.

3 вікна мають схожий функціонал (Додавання, видалення) , але вікно інтервалів дозволяє виконати зберігання лишу у випадку правильного вводу часового інтервалу дзвінків.

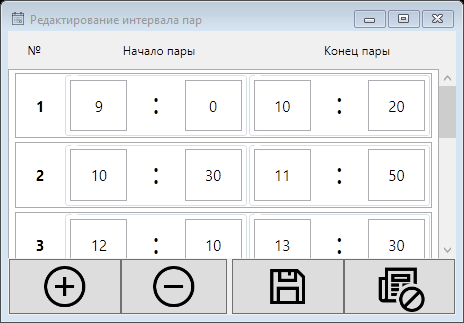
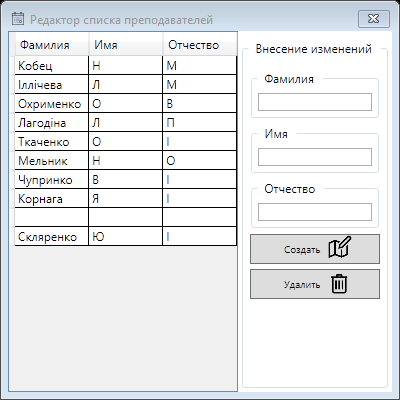
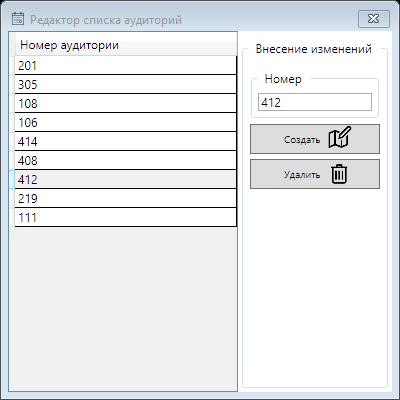
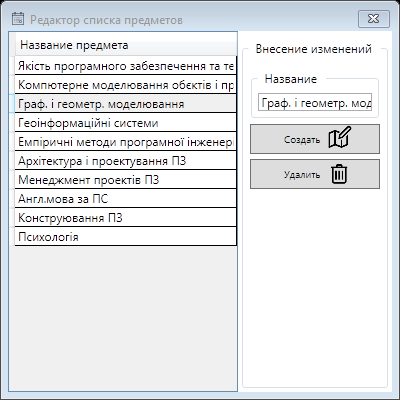
абв г

Рисунок 8 - Панелі редагування додаткових данних: а- редактор інтервалу пар; б- редактор списку лекторів; в- редактор списку лекторів; г – редактор списку предметів

На рисунку 9 відображена робоча панель, у якій ми можемо спостерігати розклад на кожень день, та перемикачі на горі між неділями (знаменик та чисельник). У режимі конструктора натискання на пару копіює її значення до панелі редагування пари, а наведення миші до кнопки «Замены» показує замінидля вибраної пари (якщо такі є).

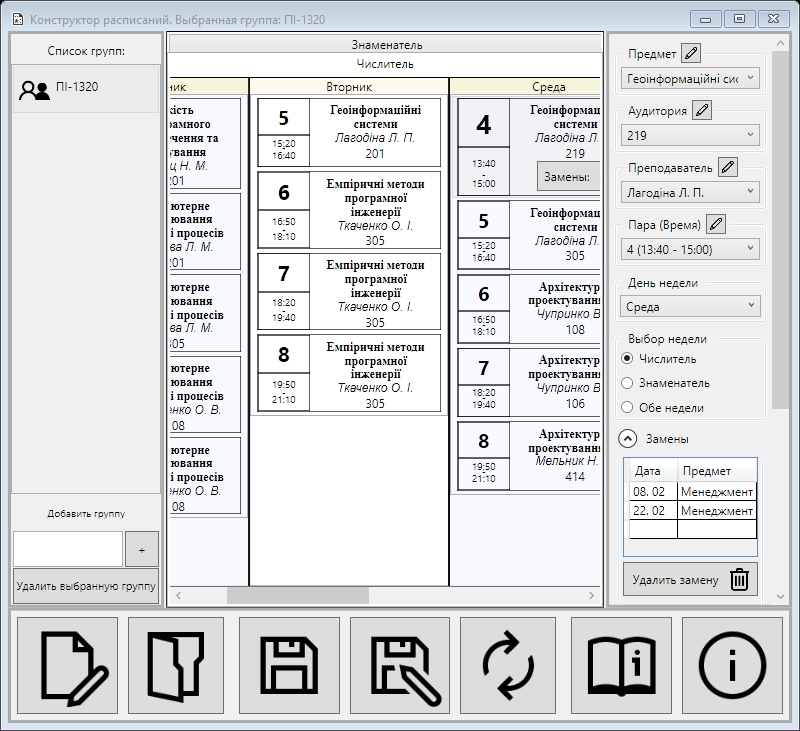


Рисунок 9 – Робоча панель

Віно синхронизації, що відображено на рисунку 10 необхідне для вводу данних для зєднання з сервером. Це в свою чергу дозволить оновоити/перевірити/завантажити нову версію розкладу

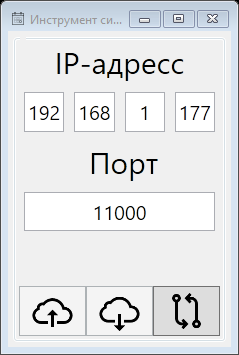


Рисунок 10 - Вікно синхронизації

3.3.2Сервер

Робочий інтерфейс серверного додатку представляє із себе консольне вікно, що зображене на рисунку 11 та рисунку 12 . Уся інформація відображаеться в ньому, деякі важливі елементи підкреслються різним кольором.

Уся інформація водиться користувачем у останю строку.

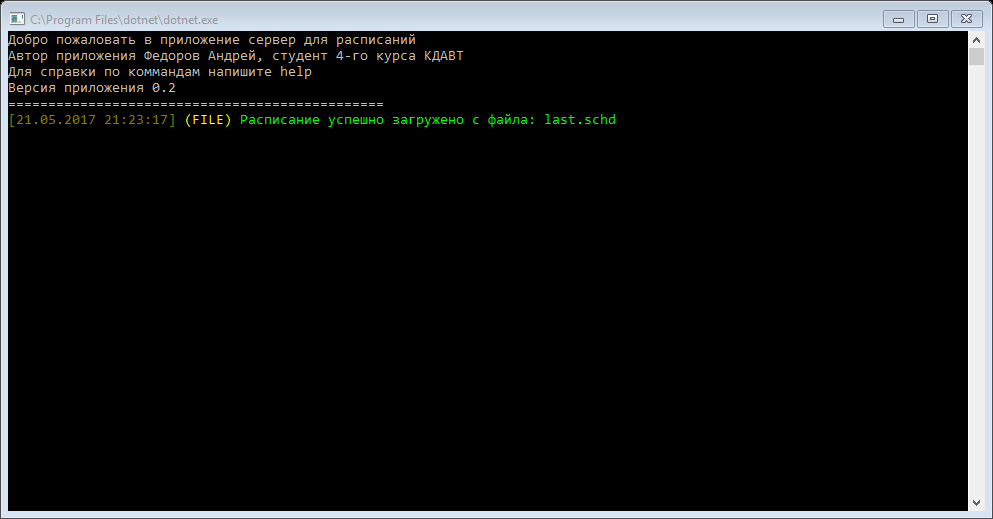


Рисунок 11 - Інтерфейс серверу

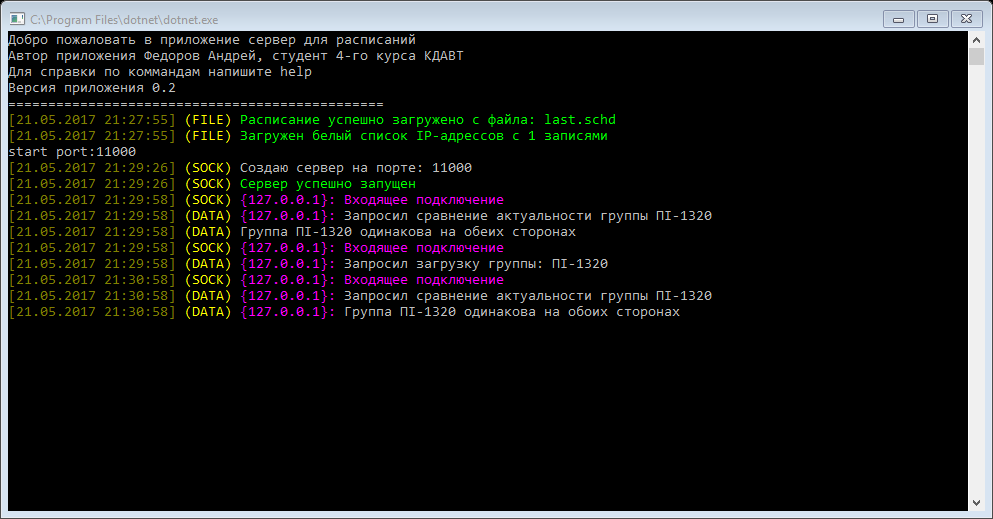


Рисунок 12 - Інтерфейс серверу в робочому прикладі

3.3.3 Переглядач

За основу інтерфейса переглядача був взят інтерфейс конструктора, з невеликими змінами. На рисунку 13 зображен інтерфейс «Переглядача».

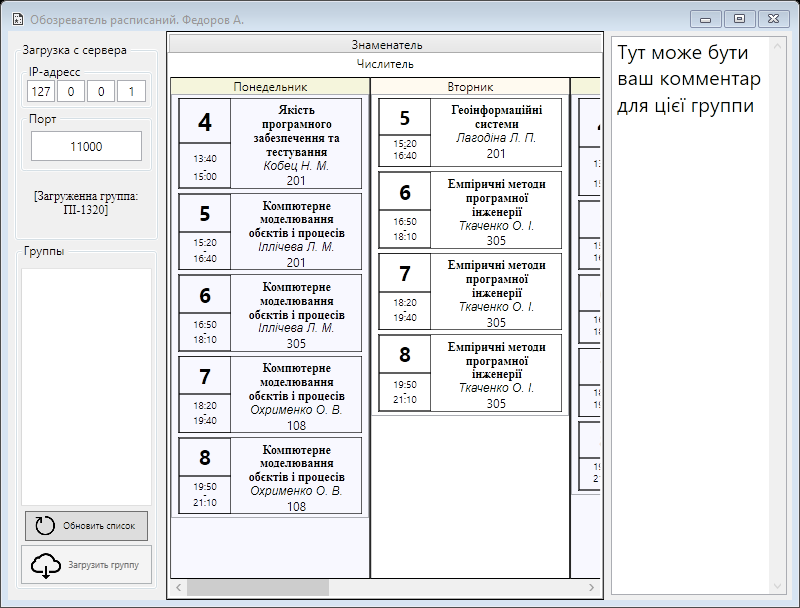


Рисунок 13 - Інтерфейс «Переглядача»

На лівій боковій панелі (див. рисунок 14) росположена панель синхронизація, вона має схожу функції з тим що є у конструктором, але працює лише з однією групою. Також кожен певний період «Переглядач» "намагаеться автоматично перевірити актуальність розкладу групу, і якщо завантажує новий розклад, тоді на екран користувачу виводиться нове повідомлення, що зображено на рисунку 15.

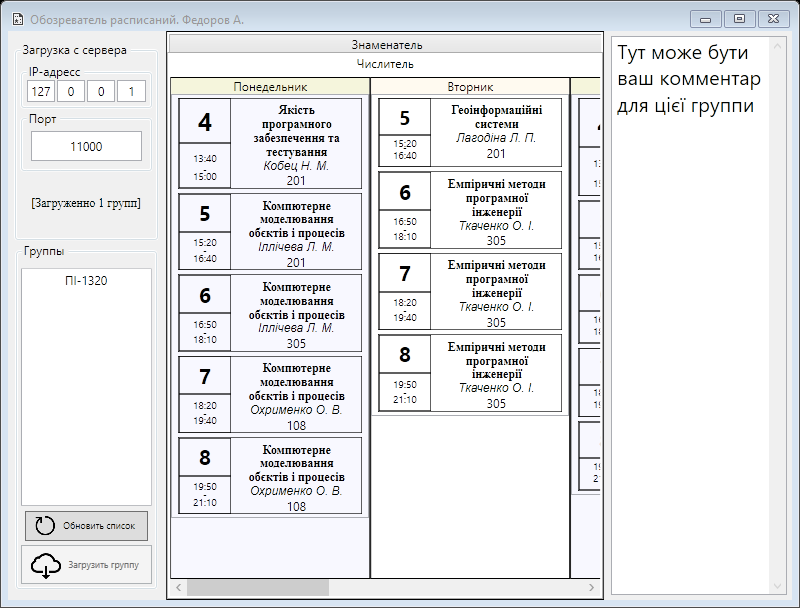


Рисунок 14 - Панель синхронизації Переглядача

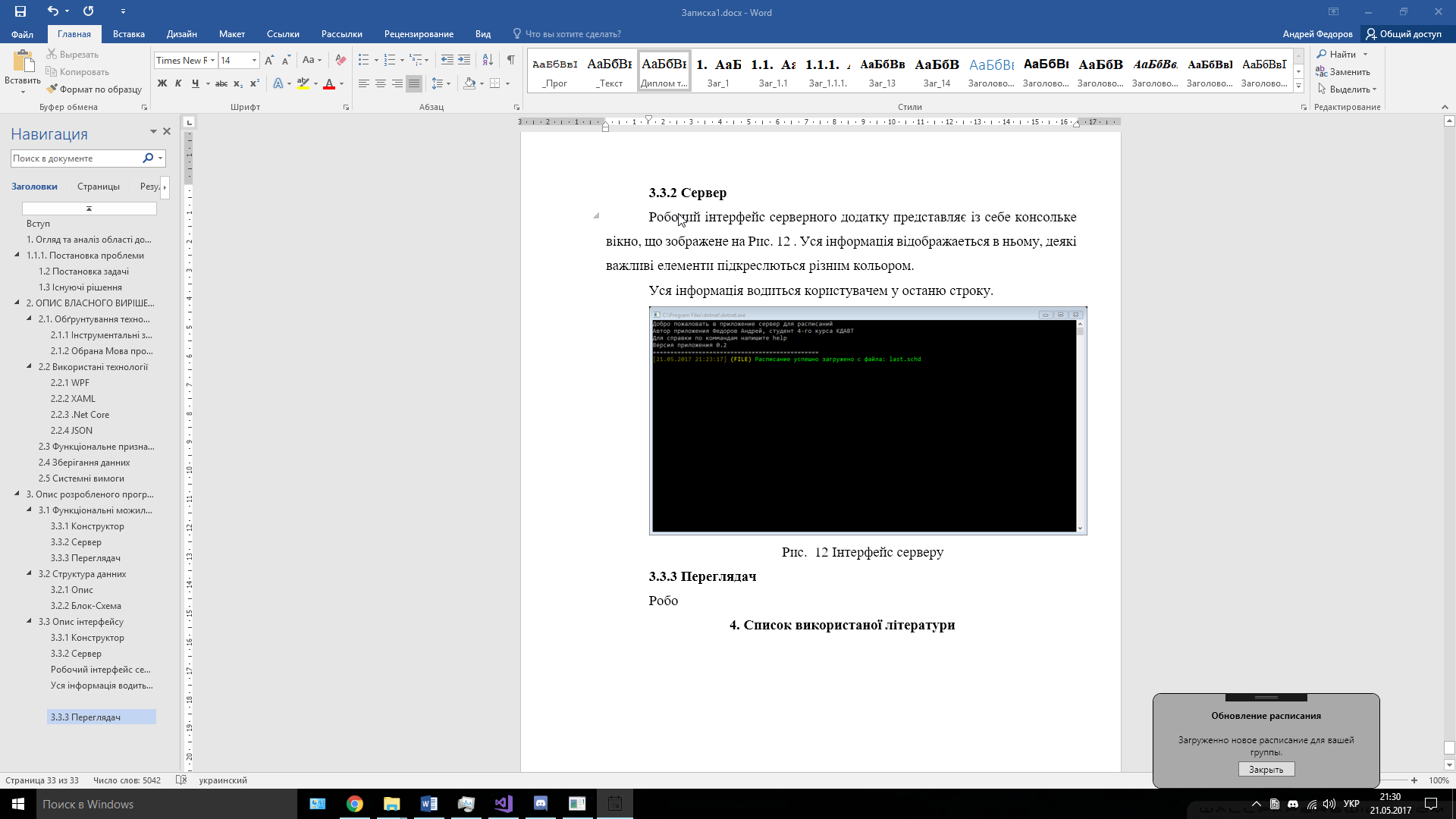


Рисунок 15 - Інформуючий баннер

Додані комментарі щодо вибраної групи будуть знаходитись в правій панелі (Рисунок 16).

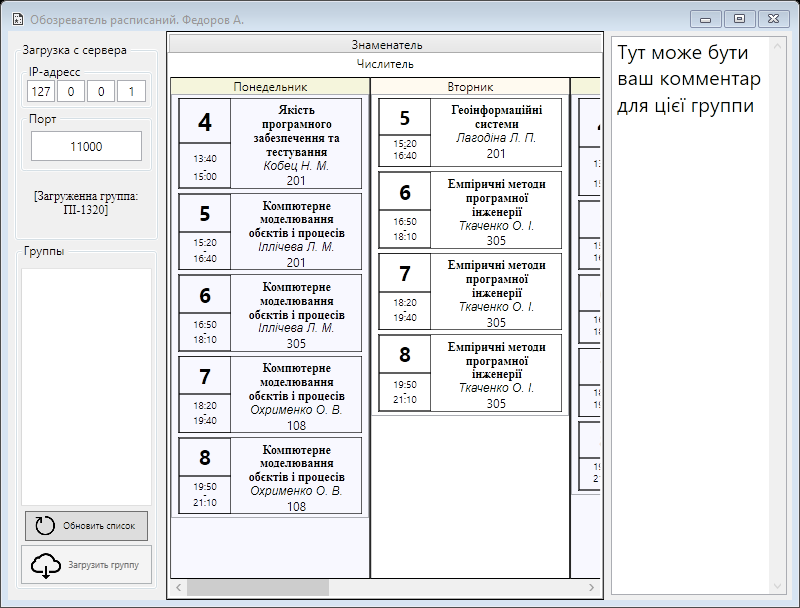


Рисунок 16 -Коментар для групи

3.4 Приклад використання

Приведемо тестову ситуація. Вищий навчальний заклад планує використовувати інформаційну систему «Розклад» для поширення розкладу серед студентів.

Поперше, оператор ПК має отримати розклади для кожної групи. Для прикладу візьмемо лиш одну групу ПІ-1320.

3.4.1Створення розкаду

Для початку оператор вмикае програму «Конструктор» та починає створювати групи на лівій панелі (рисунок 17).

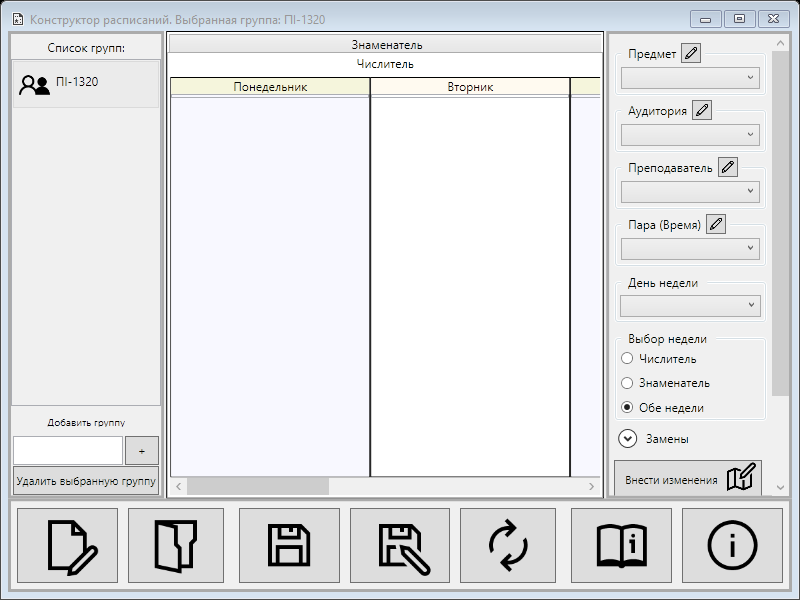
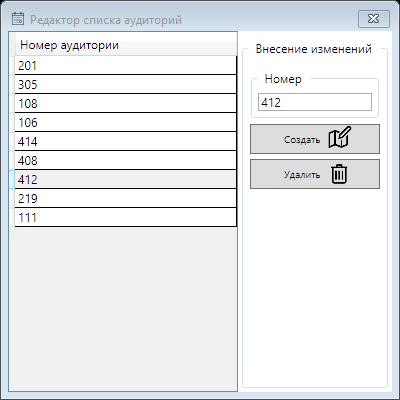
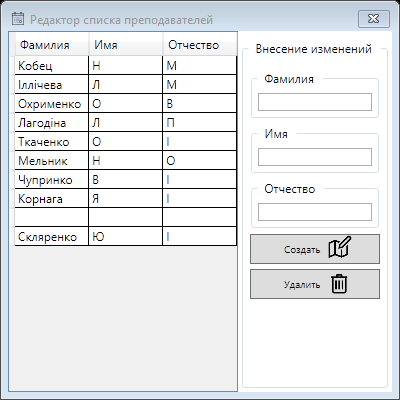


Рисунок 17 - Створенна одна група

Коли група вдало створенно, слід приступи до заповнення «додаткової інформації». Тобто заповнити лекторів, назви предметів, аудиторії. Для цього на правій панель напроти кожного поля (Предмет, Аудитория, Лектор) оператор нажимае на кнопку з олівцем, щоб відкрити вікно редагування. Ці вікна зображені на рисунку 18.

а  б

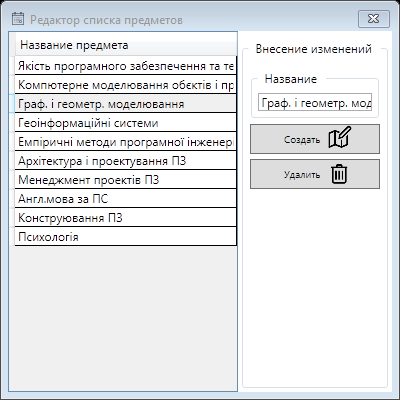
в

Рисунок 18 - Вікна заповнення інформації: а- редактор списку аудиторій; б – редактор списку лекторів; в – редактор списку предметів

Далі встановити інтервал дзвінків, натиснувши на кнопку з олівцем біля інтервалу відкриеться вікно редактору інтервалу (рисунок 19).

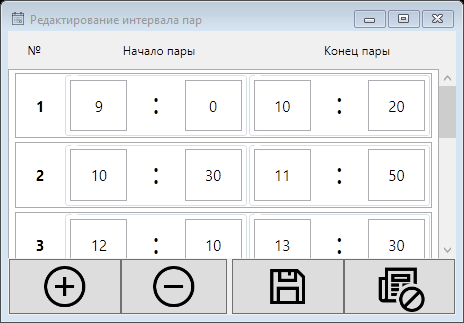


Рисунок 19 - Вікно встановлення інтервалів

Теперь у кожному полі оператор заповнює інформацію, встанволює заміни і додає пару до вибранного дня та неділі. Пара зявляеться на екрані (рисунок 20).

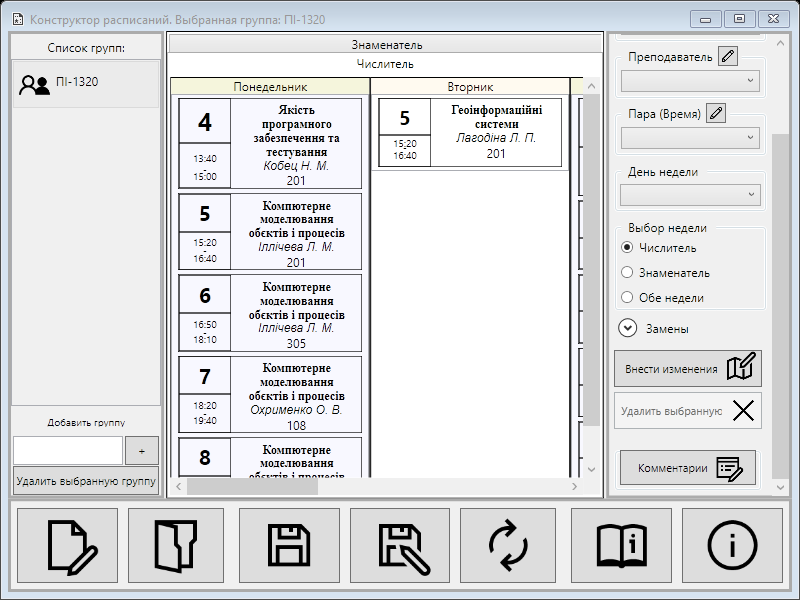


Рисунок 20 – Вигляд пари на робочій області

Оператор продовжуе заповненя, отримуемо готовий розклад що зображен на рисунку 21.

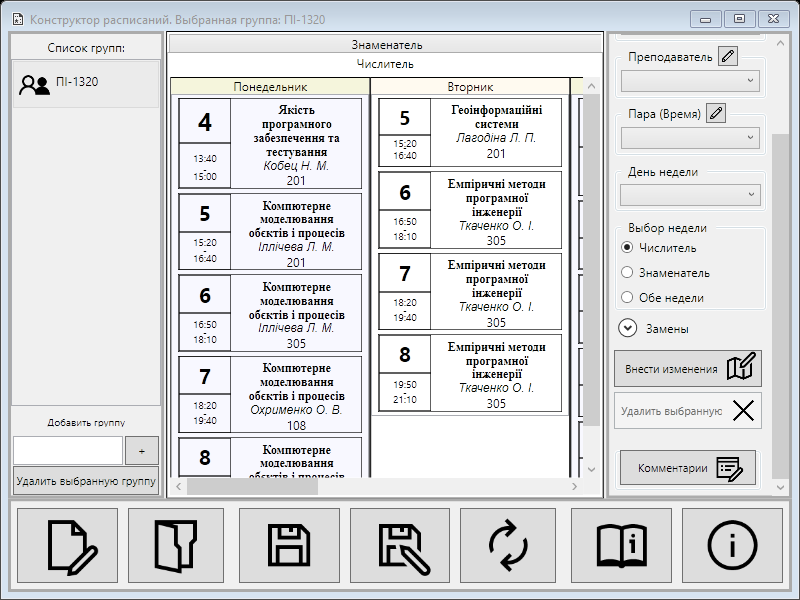


Рисунок21 - Готовий розклад

Наступним этапом буде збереження розкладу за допомогою кнопки з логотипом дискети, або поширення через меню синхронизації на сервер. Оператор відкирває меню синхронизації (рисунок 22) вводить для підключення до серверу, натискае кнопку «порівняти розклади», та відправляє свій розклад.

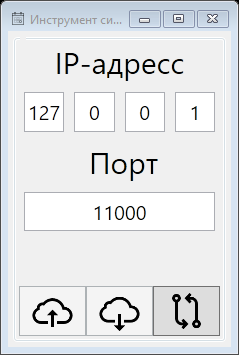


Рисунок 22 - Вікно синхронизації

Тепер розклад встановленно, студенти використовуючи «Переглядач» можуть завантажити свою групу.

3.4.2 Налаштування серверу

Для налаштування серверу необхідно запустити додаток «Сервер». Після запуску на екрані являеться вікно, що зображене на рисунку 23.

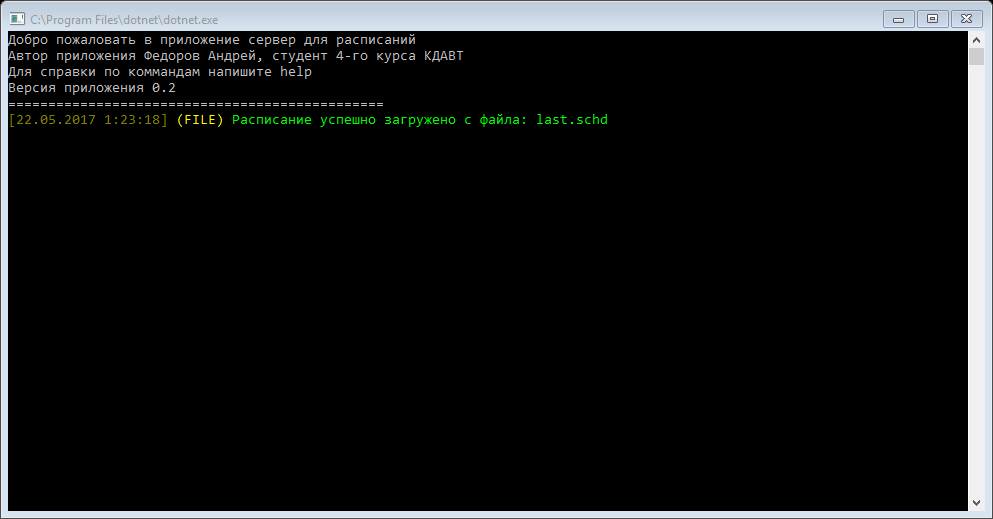


Рисунок 23 - Вид сервера після запуску

Далі оператор вводить команду start:port#### . Де # номер порта, рекомендуеться до використання порт 11000. На рисунку 24 зображен успішно запущений «Сервер».

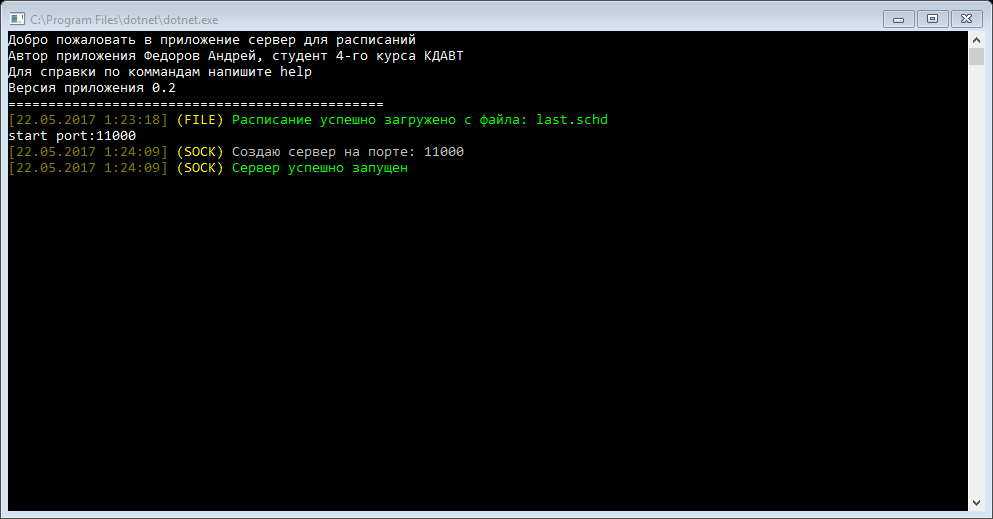


Рисунок 24- Успішний запуск сервера

Теперь для того, щоб розклад міг бути завантажений необхідно додати IP-адрессу компютера, що завантажуе до білого списку, для цього оператор пише команду whitelistadd:###.###.###.### - де # IPv4 адреса

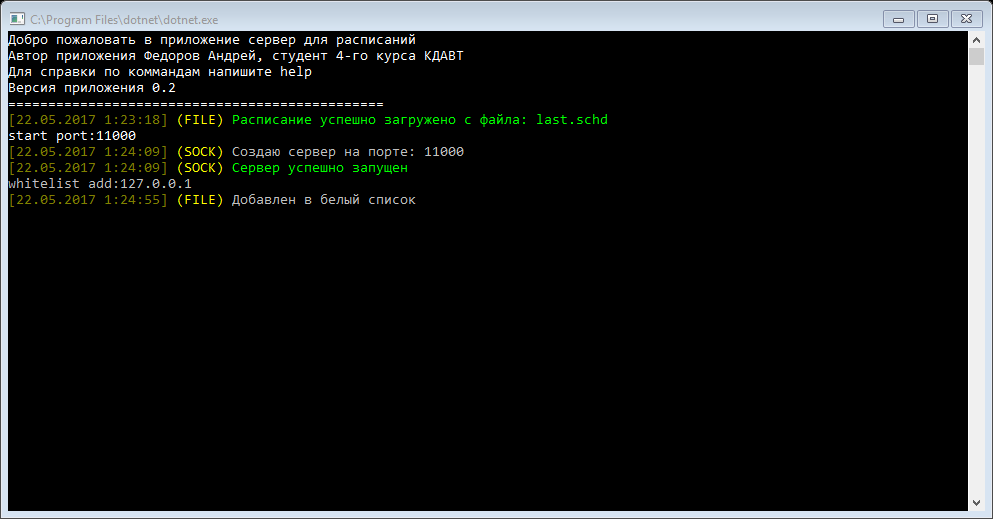


Рисунок -25 Доданний запис до білого списку

«Сервер» готов до роботи.

3.4.3 Використання студентом

Після встановлення студент запускает додаток «Переглядач» вперше. Перед ним вікдриваеться вікно, що зображено рисунку 26.

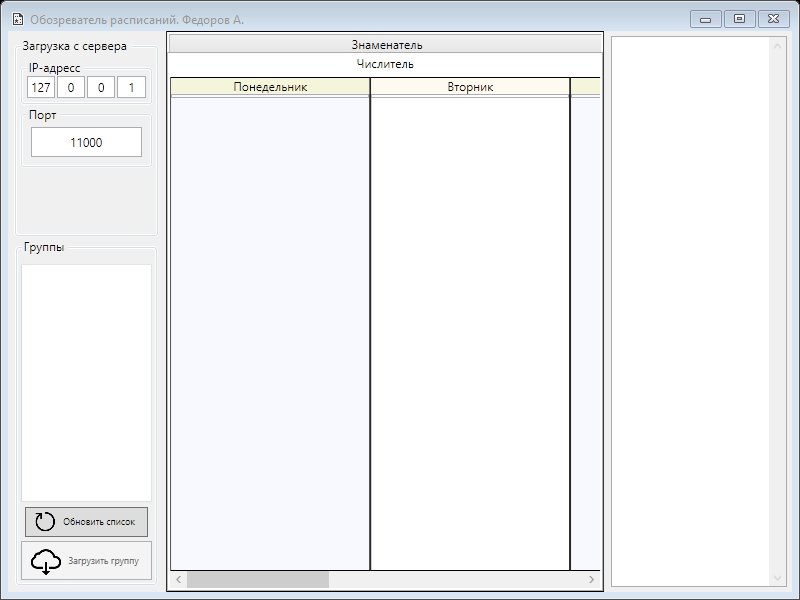


Рисунок26 - Вікно Переглядача

Перше, що необхідно для початку роботи з програмую – це ввести данні сервера (IP-адресса та порт), який студент буде отримувати від керівництва ВУЗа. Ці данні вводяться в ліве поле. Далі студент натискае «оновити список груп», вибирає свою і натискає завантажити. На рисунку 27 зображена панель синхронизації переглядача.

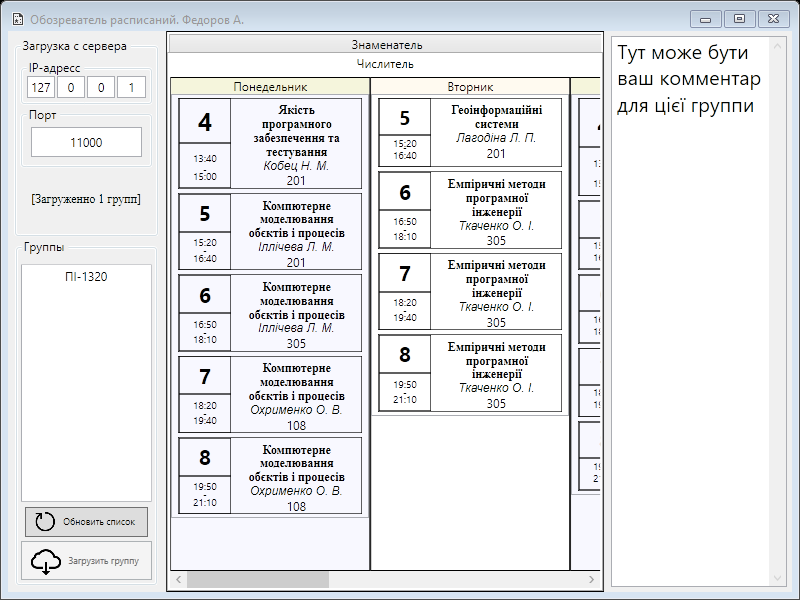


Рисунок27 - Панель синхронизації переглядача

Відтепер програма запамятала останній завантажений розклад та параметри зєднання. І вона періодачно буде перевіряти розклад, і в разі оновлення розкладу вибранної групи виводити повідомлення, як зображено на рисунку 28.

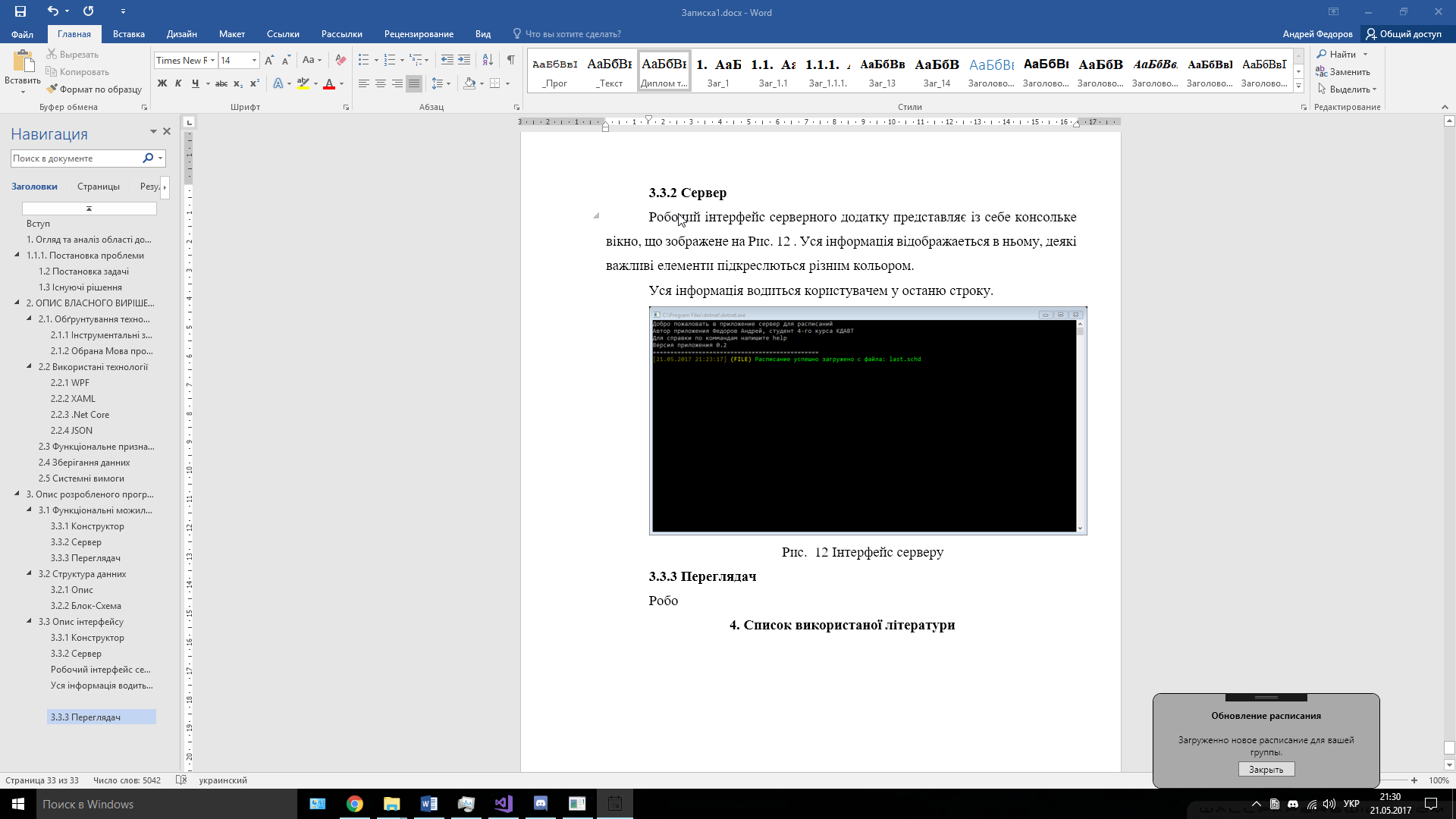


Рисунок 28 - Повідомлення про зміну розклада

ВИСНОВКИ

В результаті виконаня атестатиційної роботи бакалавра була створеннаінформаційна система «Розклад для навчальних закладів». Основною метою цього рішення – спростити створення, поширення та перегляд навчального розкладу в навчальних закладах. До складу рішення входить 3 програмних продукту: «Конструктор», «Сервер», «Переглядач».

Був проведен аналіз вже існуючих аналогів. В результаті дослідженя вияснилось, що на ринку програмних продуктів існують рішення для роботи з роскладами, але їх головний функціонал націлен на автоматизоване створення розкладу без поширення серед студентів. Тому було вирішенно створити власний програмний продукт.

Розробленне програмне рішення допомогає в створенні розкладу, та забезпечує просту можливість поширення розкладу для студентів. Налаштування та встановлення як сереврної так і студентьскої частини не потребує значних навичок .

Програмне забезпечення використовує сучасні технології для функціонування так як: JSON, .NETCore, WPF, XAML. Ці технології забезпечують просту та стабільну роботу з рішенням.

Серверне програмне рішення працює на будь-якій платформі, де існує підтримка .NETCore, навіть у системах де відсутня графічна оболчка. Це дозволяє встановити «Сервер» вже на існуюче обладення у навчальному закладу, та поставить вимогу облаштовувати нове обладення для фунціонування рішення.

Інтерфейс, що був створенний з використанням технологій WPF та XAML дуже гнучкий. Розміри елементів змінються в залежності від розміру екрану. Подальша підтримка передбачає дуже гнучку та легку зміну інтерфейсу у разі необхідності.

Під час проведення тестових прикладів було замічено скоротання часу створення розкладу, більш своєчасне та просте поширення розкладу, та більш зрозумілий вид розклад, ніж при поширенні звичайним «паперовим» методом.

В перспективі розробка «Переглядача» для мобільних пристроїв на базі Android, iOS, WindowsPhone, аналіз створеного розклад на предмет занятості аудиторії або лектора.

Отже використання інформаційної системи «Розклад», як заміну або аналог до стаднартного «паперового» рішення розкладу, може вирішити проблеми з вчасним поширенням, недорозумінням, повільним створенням розкладу.

# СПИСОК ВИКОРИСТАННИХ ДЖЕРЕЛ

1. Расписание — Википедия [Електронний ресурс]. – Режим Доступу:

<https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D1%81%D0%BF%D0%B8%D1%81%D0%B0%D0%BD%D0%B8%D0%B5>

2. Расписание занятий как организационный документ, определяющий режим работы ОУ [Електронний ресурс]. – Режим Доступу:

http://festival.1september.ru/articles/519295/

3. Розклад занять — Вікіпедія [Електронний ресурс]. – Режим Доступу:

https://uk.wikipedia.org/wiki/%D0%A0%D0%BE%D0%B7%D0%BA%D0%BB%D0%B0%D0%B4\_%D0%B7%D0%B0%D0%BD%D1%8F%D1%82%D1%8C

4. Бартенев А.С. Обзор основных вопросов автоматизированного составления расписания занятий в высшем учебном заведении [Електронний ресурс]. – Режим Доступу:

http://web.snauka.ru/issues/2011/09/2576

5. Microsoft Visual Studio — Вікіпедія [Електронний ресурс]. – Режим Доступу:

https://uk.wikipedia.org/wiki/Microsoft\_Visual\_Studio

6. Общие сведения о приложениях для Windows [Електронний ресурс]. – Режим Доступу:

https://msdn.microsoft.com/ru-ru/library/5b13a7k4(v=vs.100).aspx

7. C Sharp — Вікіпедія [Електронний ресурс]. – Режим Доступу:

https://uk.wikipedia.org/wiki/C\_Sharp

8. Windows Presentation Foundation — Вікіпедія [Електронний ресурс]. – Режим Доступу:

https://uk.wikipedia.org/wiki/Windows\_Presentation\_Foundation

9. XAML — Вікіпедія [Електронний ресурс]. – Режим Доступу:

https://uk.wikipedia.org/wiki/XAML

10. .NET Core: возможности и перспективы | DOU [Електронний ресурс]. – Режим Доступу:

https://dou.ua/lenta/articles/net-core/

11. JSON — Вікіпедія [Електронний ресурс]. – Режим Доступу:

<https://uk.wikipedia.org/wiki/JSON>

ДОДАТКИ

ДОДАТОК А. Настанова користувача

Створення розкладу

Після запуску програми отримуемо наступне вікно (рисунок А.1).

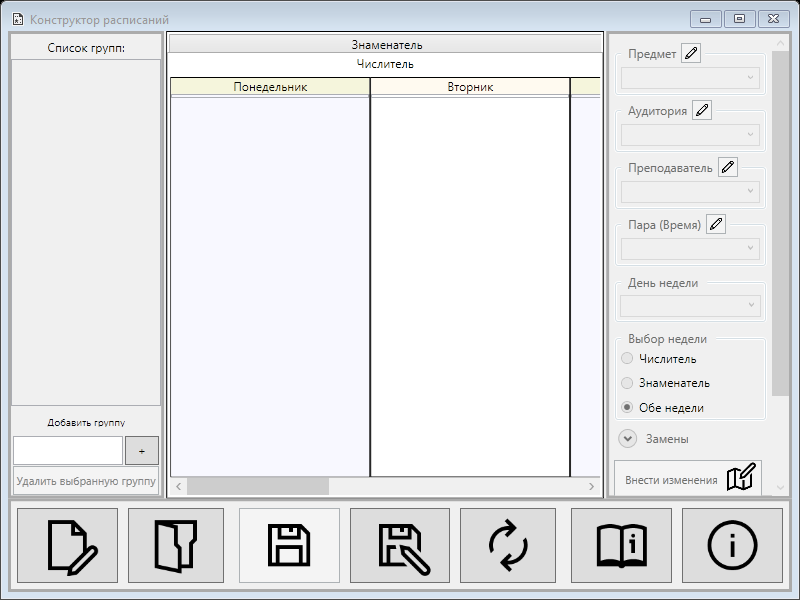


Рисунок А.1 –Інтерфейс конструктора

Для створення нового розкладу, слід натисниту на кнопку «Новий розклад» на нижній панелі, вона має наступний вигляд (рисунок А.2).

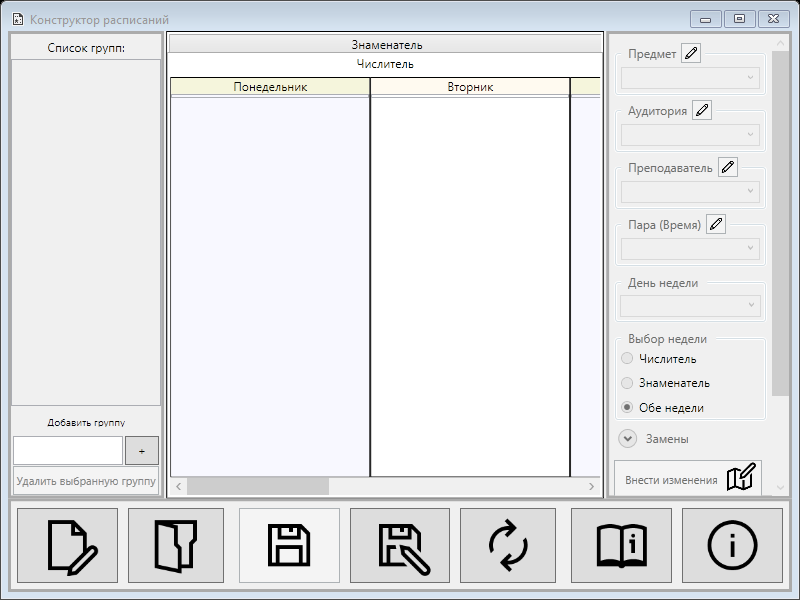


Рисунок А.2 – Кнопка створення нового розкладу

Для того, щоб додати нову групу необхідно в лівій боковій панели (рисунок А.3) ввести назву групу та натинснути +.

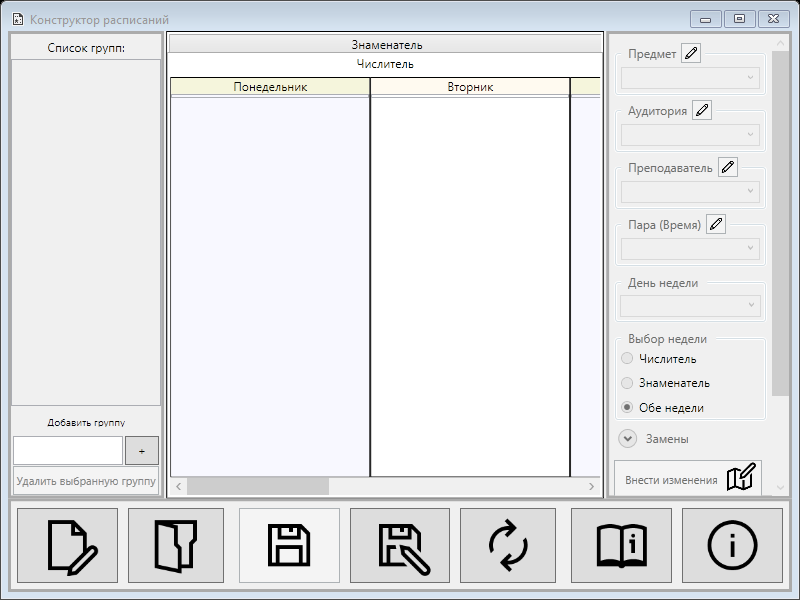


Рисунок А.3 –Створення групи.

Щойно створенна група буде автоматично вибрана. Щоб видалити групу: виберіть гуппу на лівій панелі та натисніть «Удалить выбранную групу».

Для додавання лекції до розкладу, необхідно заповнити «додактові дані». До цих даних відноситься:Аудиторія,лектор, назва предмету, інтервал. Для внесення цих даних, напротив кожного елементу вибору на правій панелі є кнопка з олівцем. На рисунку А.4 зображена ця панель.

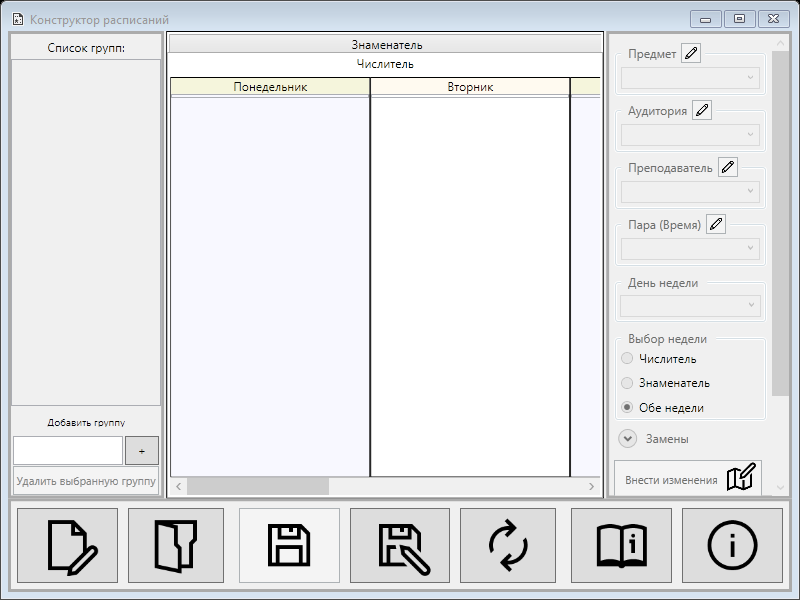


Рисунок А.4 – Панель редагування пар

Натиснувши на кнопку з олівцем, відкриваеться відповідне вікно редагування даних. Всі воні мають схожий інтерфейс, та використання. В цих вікнах на правій панелі можно створити запис заповнивши текстове поле і натисниту «Создать», або видалити вибравши необхідний запис та натиснувши «Удалить». На рисунку А.5 зображені вікна редагування додаткової інофрмації.

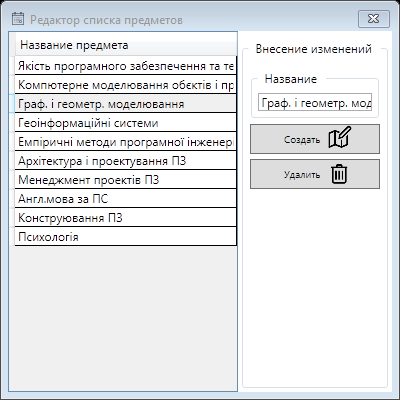
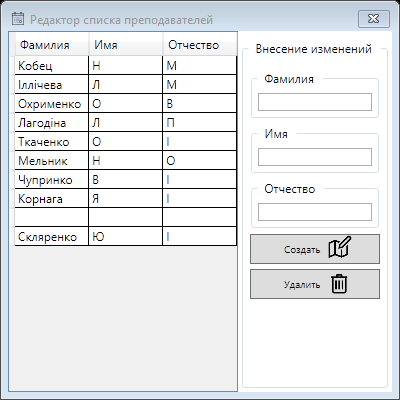
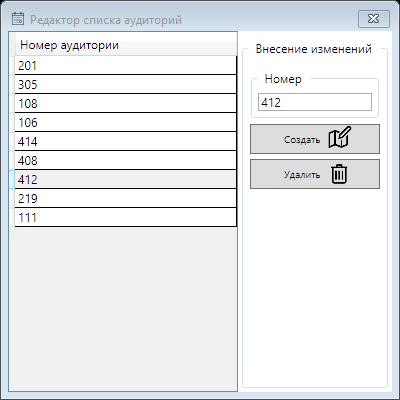
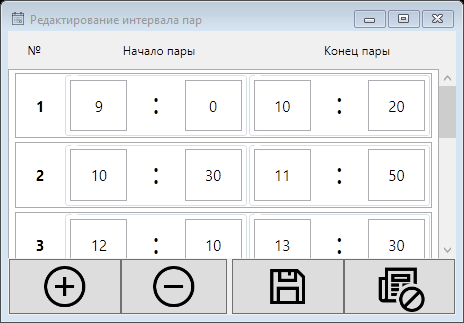
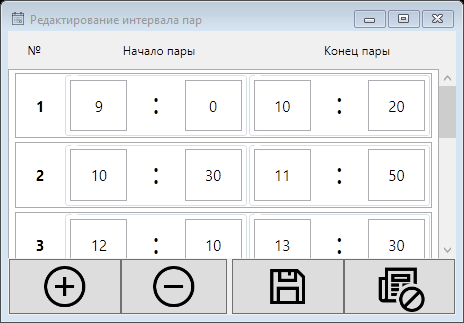
 аб  в

Рисунок А.5 - Вікна заповнення інформації: а- редактор списку предметів; б – редактор списку лекторів; в – редактор списку аудиторії

Вікно створення інтервалів відрязняється, вона зображенна рисунку А.6. Інтервали починаються з 1-ї пари. Кожен інтервал має час початку та кінця. До інтервалів застосовуються наступні правила:

* Початок не повинен бути пізніше кінця.
* Кінець не повинен бути раніше початка
* Пара що йде перед, повинна бути раніше поточну
* Початок і кінець не повинні перехрещуватися з вже існуючею парою

Для створення інтервалу натиснути кнопку , для видалення останнього натиснитку кнопку.

Для збереження існує кнопка с дискетою. Натиснувши на кнопку с перекресленням зміни відкатяться до останніх корректних.

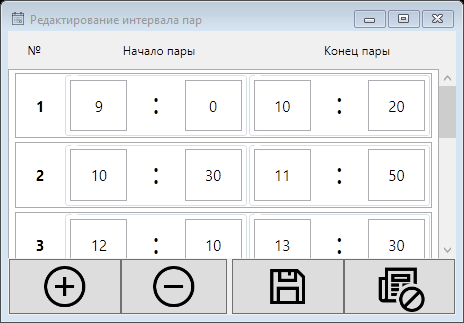


Рисунок А.6 – Редактор інтервалів

Далі слід вибрати до якої неділі слід додати (Чисельник, Знаменик або до обох) та день неділі. Якщо необхідно додати заміни до цієї пари розгорніть вкладку заміни, виберіть дату заміни та на який предмет та натисніть додати. Для видалення виберіть необхідну заміну та натисніть видалити. На рисунку А.7 зображенна панел редагування замін для пар.

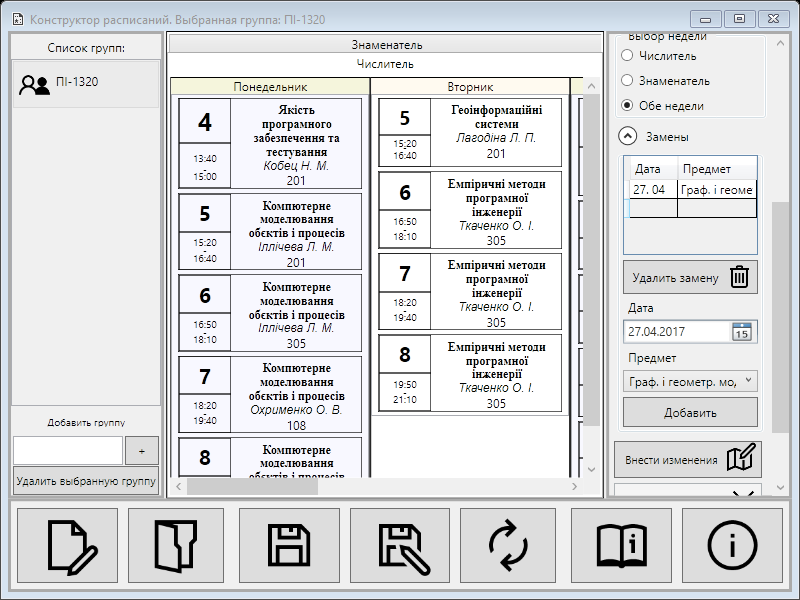


Рисунок А.7 – Редактор замін для пар

Коли всі данні заповнені натискання «Внести изменения» створить нову лекцію.

Для створення комметаря для групи виберіть кнопку «Комментарии» на правій панелі, у відкритому вікні (рисунок А.8) введіть ваш комментар та натисніть зберегти.

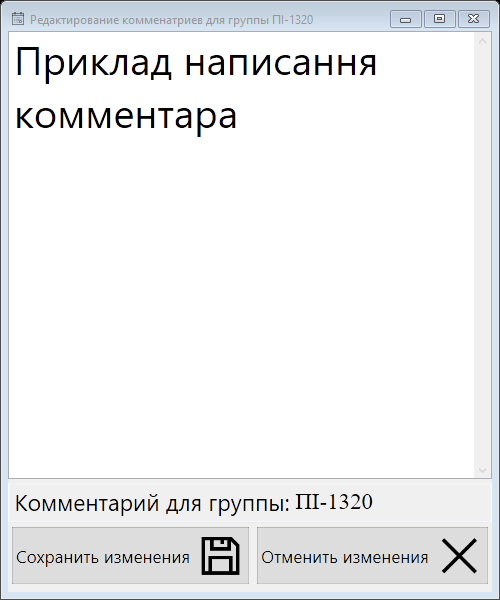


Рисунок А.8 – Редактор комментарів

Для збереження розкладу існує три варіанта: зберегти, зберегти як, синхронізувати. На функціональні панелі ці кнопки зображені як на рисунку А.9.

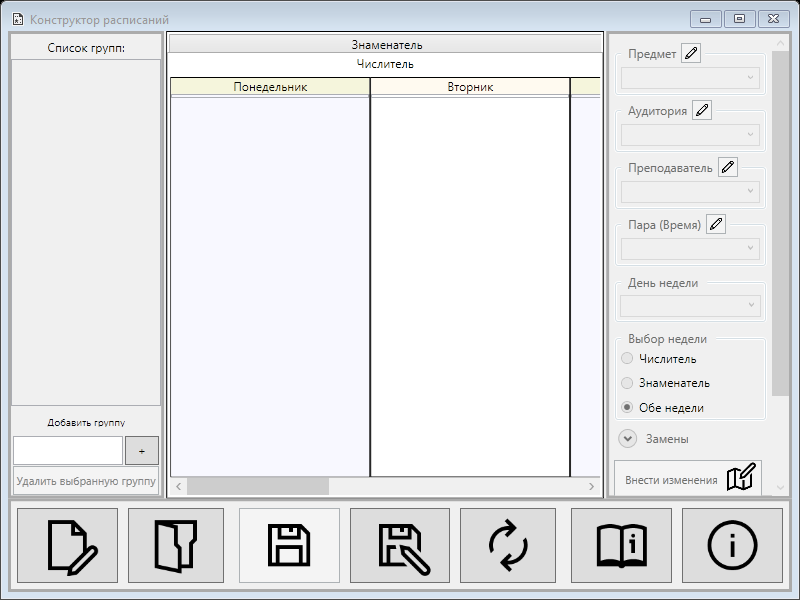


Рисунок А.9 – Панель збереження

Перший варіант зберігає до файлу з якого був відкрит розклад або збереженний останій раз. Варіант «зберегти як» запропонує користувачу вибрати до якої теки зберегти розклад під яким ім’ям.

Для синхронизації даних із сервером, відкрийте панель синхронизації (кнопка з малюнком стрілок), панель зображена на рисунку А.10. Введіть IP-адресу, та порт що надало вам керівництво вашого навчального закладу.

Спочатку натисніть на праву кнопку (перевірити), це порівняє ваш розклад з встановленим на сервері, якщо вони відрізняються у вас буде дві інші кнопки : «Завантажити на сервер» або «Завантажити з серверу».

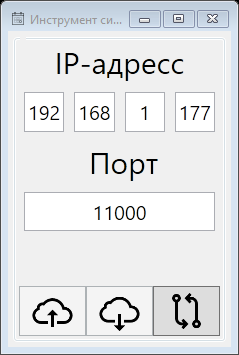


Рисунок А.10 – Панель синхронизації

Для роботи з переглядачем необхідно завантажити конкретну групу з серверу. Далі «Переглядач» буде самостійно перевіряти оновлення до цієї вибраної групи.

Щоб налаштувати, у лівій панелі переглядача (рисунок А.11) заповніть данні для з’єднання з сервером, що надав вам ваш навчальний заклад, до цих данних відносяться IP-адресса та порт.

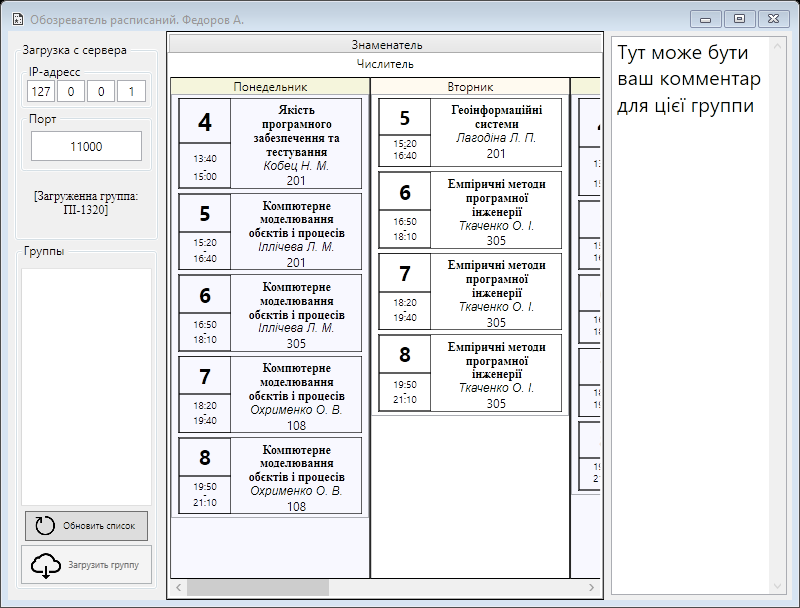


Рисунок А.11 – Панель підключення у переглядача.

Коли данні введені, натисніть кнопку «Обновить список», у полі нижче повинні зявитися доступні групи. На рисунку А.12 зображен приклад завантаженої групи.

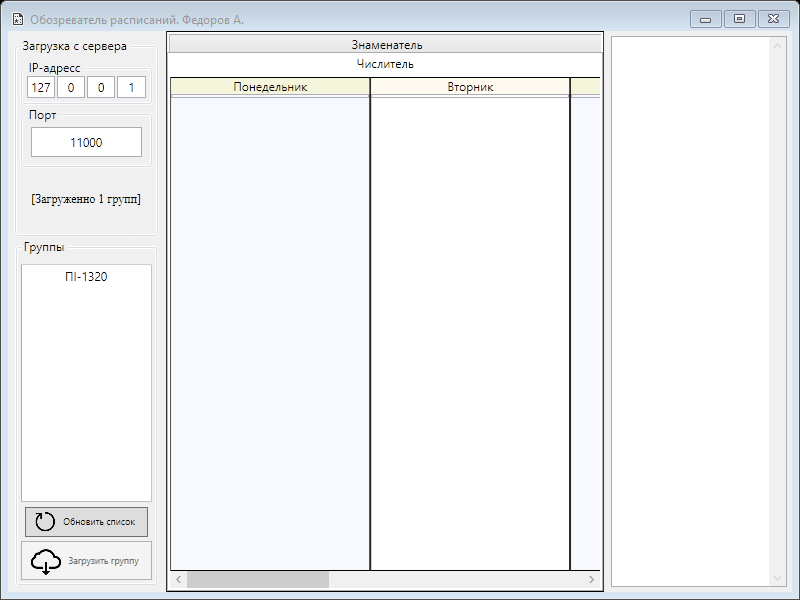


Рисунок А.12 – Список груп доступних на сервері

Виберіть бажану групу, натисніть «Загрузить групу». Через мить на робочій панелі зявиться ваш розклад, а на правій панелі комментар щодо вашої групи. Кожен деякий інтервал додаток буде перевіряти наявність нової версії розкладу використовуючи остані вдалі данні доступу до серверу для вашої групи. Якщо додаток онове ваш розклад, ви отримаете повідомлення (рисунок А.13).

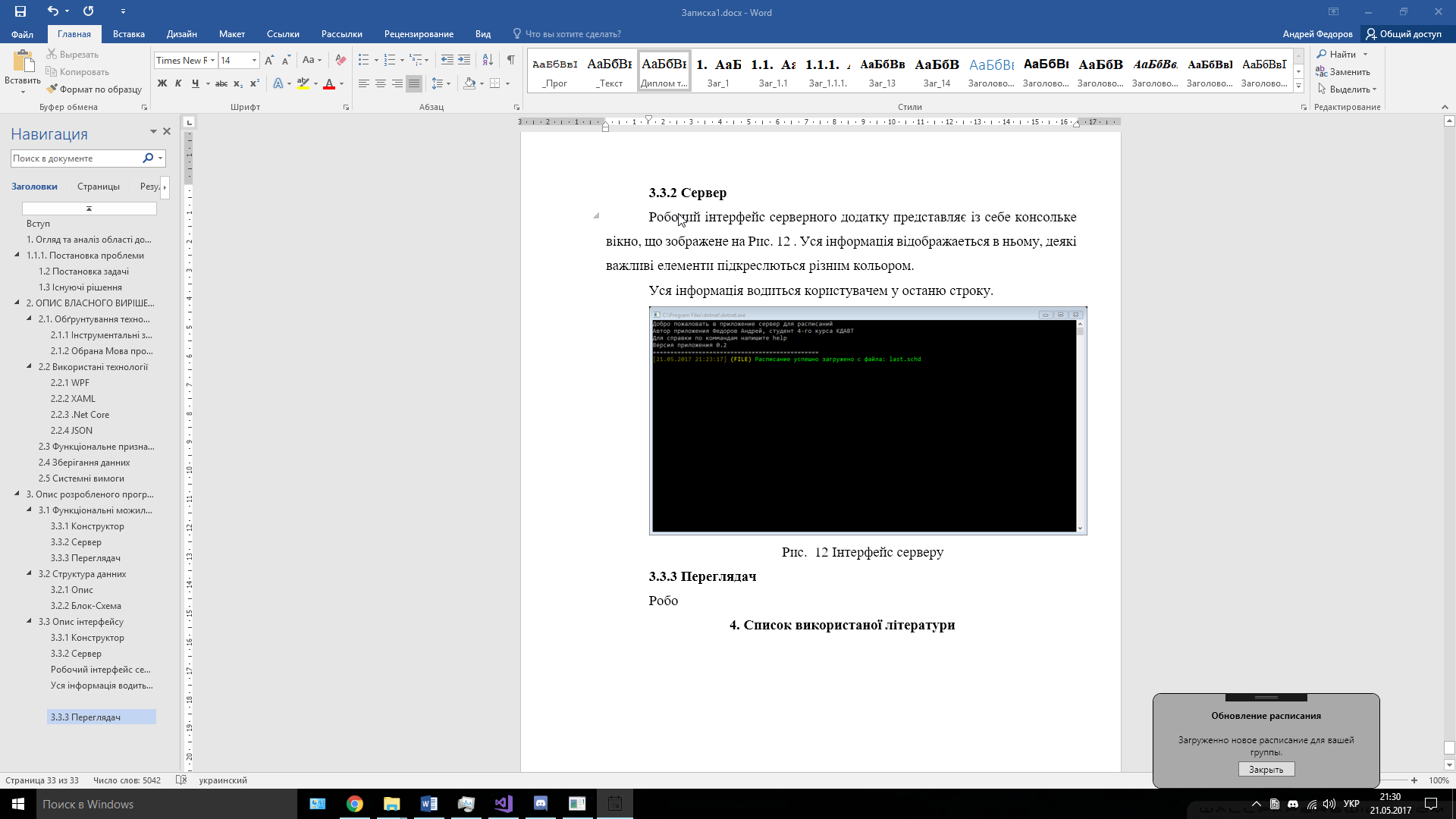


Рисунок А.13 – Повідомлення о змінах розклада

Для запуску сервера виконайте .exe фалйу, для інших ОС зверніться до справки Microsoft .NETCore щодо запуску додатків. Для Linuxсистем відкрийте термінал у теці додатку та виконайте наступну комманду у терміналі:

*dotnetServer.dll*

Після запуску додатку перед вами будеконсольне вікно, що зображена на рисунку А.14.

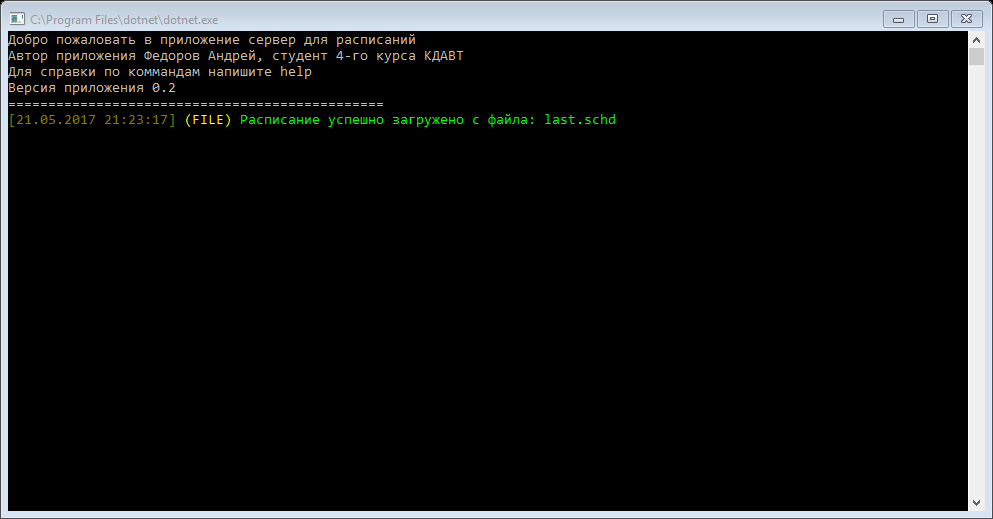


Рисунок А.14 – Запуск інтерфейс

Для старту серверу використовуйте команду startport:###### , де # ціле числове значення порта.

Після вдалого старту сервера ви отримаете наступний вигляд, як зображений на рисунку А.15.

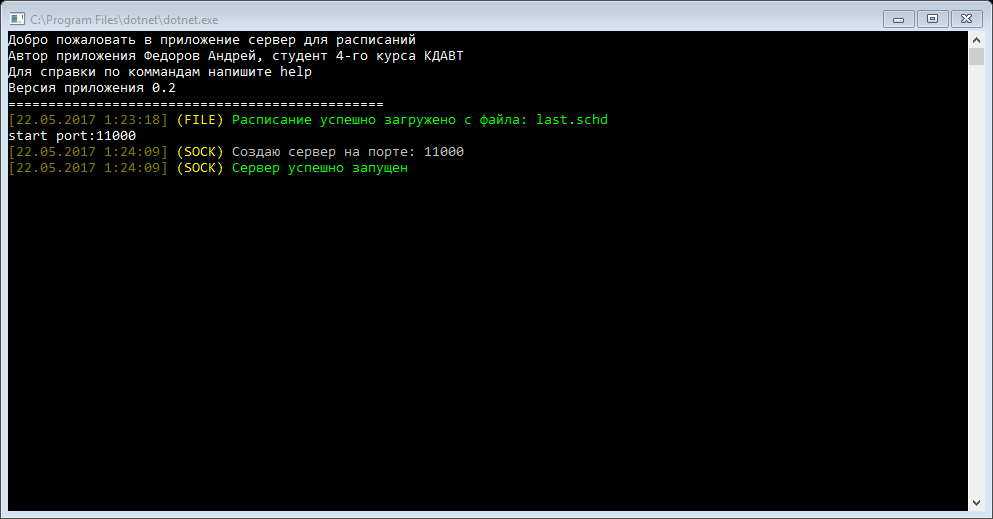


Рисунок А.15 – Успішний запуск серверу

Білий список – список IPадрес, яким дозволяеться перезаписувати встановлений на сервері розклад.

Чорний список – список IPадрес, яким буде відмовленно о наданні послуг.

Для додавання адреси до списку виконайте наступну комманду:

*(whitelist/blacklist) add:###.###.###.###*

Виберіть одну с комманд, що знаходяться в дужках залежно від вибарного списку. Замінити *###.###.###.###*наIPv4-адресу.

Для виделння адреси зі списку виконайте наступну комманду:

*(whitelist/blacklist) remove:###.###.###.###*

Виберіть одну с комманд, що знаходяться в дужках залежно від вибарного списку. Замінити *###.###.###.###*на IPv4-адресу.

Для виводу допомогу використовуйте команду help.

ДОДАТОК Б. Код програми

usingSystem;

using System.Collections.Generic;

using System.Text;

namespace Builder

{

staticclassCipher

{

publicstaticstring encryption(string informmation)

{

if (informmation.Length % 2 != 0)

{

informmation += " ";

}

int number = 0;

string cod = "";

for (int i = 0; i < informmation.Length; i += 2)

{

cod += (char)(informmation[i] + 1 + number);

number++;

if (number > 3)

{

number = 0;

}

}

number = 0;

for (int i = 1; i < informmation.Length; i += 2)

{

cod += (char)(informmation[i] + 2 + number);

number++;

if (number > 3)

{

number = 0;

}

}

return cod;

}

publicstaticstring transcript(string informmation)

{

string cod = "";

int number = 0;

for (int i = 0; i < informmation.Length / 2; i++)

{

cod += (char)(informmation[i] - 1 - number);

cod += (char)(informmation[i + informmation.Length / 2] - 2 - number);

number++;

if (number > 3)

{

number = 0;

}

}

if (cod[cod.Length - 1] == ' ')

{

string cod2 = "";

for (int i = 0; i < cod.Length - 1; i++)

{

cod2 += cod[i];

}

cod = cod2;

}

return cod;

}

}

}

using Newtonsoft.Json;

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Linq;

using System.Security.Cryptography;

using System.Text;

using System.Threading.Tasks;

namespace Builder

{

publicstaticpartialclassData

{

publicclassHash

{

privatestaticMD5 md5Hash = MD5.Create();

publicstaticstring GetMd5Hash(string input)

{

// Convert the input string to a byte array and compute the hash.

byte[] data = md5Hash.ComputeHash(Encoding.UTF8.GetBytes(input));

// Create a new Stringbuilder to collect the bytes

// and create a string.

StringBuilder sBuilder = newStringBuilder();

// Loop through each byte of the hashed data

// and format each one as a hexadecimal string.

for (int i = 0; i < data.Length; i++)

{

sBuilder.Append(data[i].ToString("x2"));

}

// Return the hexadecimal string.

return sBuilder.ToString();

}

// Verify a hash against a string.

publicstaticbool VerifyMd5Hash(string input, string hash)

{

// Hash the input.

string hashOfInput = GetMd5Hash(input);

// Create a StringComparer an compare the hashes.

StringComparer comparer = StringComparer.OrdinalIgnoreCase;

if (0 == comparer.Compare(hashOfInput, hash))

{

returntrue;

}

else

{

returnfalse;

}

}

}

}

publicpartialclassSchedule

{

public Schedule() { }

publicIntervalCollection intervals;

publicObservableCollection<Lector> lectorList { get; set; }

publicObservableCollection<string> predmetList { get; set; }

publicObservableCollection<Group> groupList { get; set; }

publicObservableCollection<int> classList { get; set; }

publicstring pcName;

publicstring json() { returnCipher.encryption(JsonConvert.SerializeObject(this)); }

publicstaticSchedule fillMe(string \_Encodedjson)

{

if (string.IsNullOrWhiteSpace(\_Encodedjson)) thrownewException("Файлнепрочитан, илиегосодержимоепустое");

string \_json = Cipher.transcript(\_Encodedjson);

Schedule newone = JsonConvert.DeserializeObject<Schedule>(\_json);

return newone;

}

publicstring hash()

{

string firstJson = JsonConvert.SerializeObject(this);

returnData.Hash.GetMd5Hash(firstJson);

// так надо ибо некоторые типы интерпритируються по разному после десериализации

}

}

}

using Newtonsoft.Json;

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Linq;

using System.Security.Cryptography;

using System.Text;

using System.Threading.Tasks;

namespace Builder

{

publicstaticpartialclassData

{

publicclassHash

{

privatestaticMD5 md5Hash = MD5.Create();

publicstaticstring GetMd5Hash(string input)

{

// Convert the input string to a byte array and compute the hash.

byte[] data = md5Hash.ComputeHash(Encoding.UTF8.GetBytes(input));

// Create a new Stringbuilder to collect the bytes

// and create a string.

StringBuilder sBuilder = newStringBuilder();

// Loop through each byte of the hashed data

// and format each one as a hexadecimal string.

for (int i = 0; i < data.Length; i++)

{

sBuilder.Append(data[i].ToString("x2"));

}

// Return the hexadecimal string.

return sBuilder.ToString();

}

// Verify a hash against a string.

publicstaticbool VerifyMd5Hash(string input, string hash)

{

// Hash the input.

string hashOfInput = GetMd5Hash(input);

// Create a StringComparer an compare the hashes.

StringComparer comparer = StringComparer.OrdinalIgnoreCase;

if (0 == comparer.Compare(hashOfInput, hash))

{

returntrue;

}

else

{

returnfalse;

}

}

}

}

publicpartialclassSchedule

{

public Schedule() { }

publicIntervalCollection intervals;

publicObservableCollection<Lector> lectorList { get; set; }

publicObservableCollection<string> predmetList { get; set; }

publicObservableCollection<Group> groupList { get; set; }

publicObservableCollection<int> classList { get; set; }

publicstring pcName;

publicstring json() { returnCipher.encryption(JsonConvert.SerializeObject(this)); }

publicstaticSchedule fillMe(string \_Encodedjson)

{

if (string.IsNullOrWhiteSpace(\_Encodedjson)) thrownewException("Файлнепрочитан, илиегосодержимоепустое");

string \_json = Cipher.transcript(\_Encodedjson);

Schedule newone = JsonConvert.DeserializeObject<Schedule>(\_json);

return newone;

}

publicstring hash()

{

string firstJson = JsonConvert.SerializeObject(this);

returnData.Hash.GetMd5Hash(firstJson);

// так надо ибо некоторые типы интерпритируються по разному после десериализации

}

}

}

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Builder

{

publicpartialclassIntervalCollection

{

public IntervalCollection() { }

publicint last { get { return timeList.Count; } }

publicObservableCollection<Interval> timeList = newObservableCollection<Interval>();

}

publicpartialclassInterval

{

publicint index;

publicTimeSpan start;

publicTimeSpan end;

public Interval() { }

}

}

using System;

using System.Collections.Generic;

using System.IO;

using System.Text;

namespace Server

{

publicclassCommandLine

{

conststring LOGTYPE = "INPT";

publicstring command;

publicArgument[] arguments;

publicCommandLine get()

{

string input = Console.ReadLine();

string[] raw = input.ToLower().Split(' ');

File.AppendAllText(Log.filename, "[" + DateTime.Now + "] >>>>" + input + Environment.NewLine);

arguments = newArgument[10];

command = raw[0];

if (raw.Length != 1)

{

for (int i = 1; i < raw.Length; i++)

{

string[] split = raw[i].Split(':');

try

{

arguments[i - 1] = newArgument(split[0], split[1]);

}

catch (IndexOutOfRangeException)

{

Log.write(LOGTYPE, "Для " + command + " произведеннекоректныйвводпараметра", ConsoleColor.Yellow);

returnnull;

}

}

}

string checkSTR = check();

if (checkSTR == "parametr")

{

Log.write(LOGTYPE, "Команда " + command + " невыполнена. Ошибкавводааргументов", ConsoleColor.Yellow);

returnnull;

}

elseif (checkSTR == "nocommand")

{

Log.write(LOGTYPE, "Комманда " + command + " несуществует", ConsoleColor.Yellow);

returnnull;

}

returnthis;

}

string check()

{

switch (command)

{

case"start":

if (getParametr("port") != null) return"good"; else { return"parametr"; };

case"stop": return"good";

case"help": return"good";

case"blacklist":

if (getParametr("add") != null || getParametr("remove") != null) return"good"; else { return"parametr"; };

case"whitelist":

if (getParametr("add") != null || getParametr("remove") != null) return"good"; else { return"parametr"; };

}

return"nocommand";

}

publicArgument getParametr(string lookingfor)

{

foreach (Argument a in arguments)

{

if (a?.argument == lookingfor)

{

return a;

}

}

returnnull;

}

}

publicclassArgument

{

publicstring argument;

publicstring parametr;

public Argument(string aug, string param)

{

argument = aug;

parametr = param;

}

}

}

using System;

using System.Collections.Generic;

using System.Net;

using System.Text;

namespace Server

{

publicstaticclassFunction

{

publicstaticArgument getParametr(CommandLine input, string lookingfor)

{

foreach (Argument a in input.arguments)

{

if (a?.argument == lookingfor)

{

return a;

}

}

returnnull;

}

publicstaticvoid execute(CommandLine input)

{

string command;

try

{

command = input.command;

}

catch (NullReferenceException)

{

return;

}

switch (command)

{

case"start":

startFunc(input);

break;

case"stop":

stopFunc();

break;

case"help":

helpFunc();

break;

case"blacklist":

blacklist(input);

break;

case"whitelist":

whitelist(input);

break;

}

}

privatestaticvoid startFunc(CommandLine input)

{

int port = 0;

try

{

port = Convert.ToInt32(getParametr(input, "port").parametr);

}

catch (FormatException) { Log.write("INPT", "Введенневерныйпортдлякомманды start", ConsoleColor.Red); return; }

if (Server.SocketServer.status == "Отключен")

SocketServer.Initialize(port);

else { Log.write("SOCK", "Серверужезапущен. Для остановки используйте команду stop", ConsoleColor.Red); }

}

privatestaticvoid stopFunc()

{

if (Server.SocketServer.status == "Включен")

SocketServer.DeActivate();

else { Log.write("SOCK", "Сервернезапущен. Для запуска используйте команду start port:номерпорта", ConsoleColor.Red); }

}

#region help

publicstaticList<helpinfo> allhelp = newList<helpinfo>();

publicstructhelpinfo

{

publicstring command;

publicstring desc;

publicList<argumentinfo> arguments;

}

publicstructargumentinfo

{

publicstring argument;

publicstring parametr;

}

privatestaticvoid helpFunc()

{

Console.WriteLine("=======================");

Console.WriteLine("Коммандывводятьсявследующемформате: {комманда} {аргумент}:{параметраргумента}. Пример: start port:11000");

Console.WriteLine("=======================");

void fillHelp()

{

helpinfo toAdd;

argumentinfo toAdd2;

toAdd.command = "start";

toAdd.desc = "Запускает сервер на выбранном порте";

toAdd.arguments = newList<argumentinfo>();

toAdd2.argument = "port";

toAdd2.parametr = "Обязательныйпараметр. Указывает порт на котором будет запущен сервер";

toAdd.arguments.Add(toAdd2);

allhelp.Add(toAdd);

//////////////

toAdd.command = "stop";

toAdd.desc = "Останавливаетсервер";

toAdd.arguments = newList<argumentinfo>();

allhelp.Add(toAdd);

//////////////

toAdd.command = "blacklist";

toAdd.desc = "IP-адреса занесенные в черный список не обрабатываються";

toAdd.arguments = newList<argumentinfo>();

toAdd2.argument = "add";

toAdd2.parametr = "Добавляет IP-адресвчерныйсписок. Прим: blacklist add:127.0.0.1";

toAdd.arguments.Add(toAdd2);

toAdd2.argument = "remove";

toAdd2.parametr = "Удаляет IP-адрес из черного списка. Прим: blacklist remove:127.0.0.1";

toAdd.arguments.Add(toAdd2);

allhelp.Add(toAdd);

//////////////

toAdd.command = "whitelist";

toAdd.desc = "Расписания устанавливаються на сервер только с IP-адресов занесеные в белый список";

toAdd.arguments = newList<argumentinfo>();

toAdd2.argument = "add";

toAdd2.parametr = "Добавляет IP-адресвбелыйсписок. Прим: whitelist add:127.0.0.1";

toAdd.arguments.Add(toAdd2);

toAdd2.argument = "remove";

toAdd2.parametr = "Удаляет IP-адрес из белого списка. Прим: whitelist remove:127.0.0.1";

toAdd.arguments.Add(toAdd2);

allhelp.Add(toAdd);

}

if (allhelp.Count == 0) fillHelp();

foreach (helpinfo h in allhelp)

{

Console.WriteLine("Команда: " + h.command);

Console.WriteLine(h.desc);

if (h.arguments.Count > 0)

{

Console.WriteLine("Аргументы:");

foreach (argumentinfo a in h.arguments)

{

Console.WriteLine(a.argument + " - " + a.parametr);

}

}

Console.WriteLine("=======================");

}

}

#endregion

privatestaticvoid blacklist(CommandLine input)

{

IPAddress ip;

try

{

ip = IPAddress.Parse(input.arguments[0].parametr);

}

catch (FormatException) { Log.write("INPT", "Введенневерный IP длякомманды blacklist", ConsoleColor.Red); return; }

if (input.arguments[0].argument == "add")

{

Builder.Global.listAdd(Builder.Global.blacklist, ip);

Builder.Global.SaveList(Builder.Global.blacklist, "blacklist.json");

Log.write("FILE", ip, "Добавлен в черный список");

}

if (input.arguments[0].argument == "remove")

{

if (Builder.Global.listRemove(Builder.Global.blacklist, ip))

{

Builder.Global.SaveList(Builder.Global.blacklist, "blacklist.json");

Log.write("FILE", ip, "Убран из черного списка");

}

else { Log.write("FILE", ip, "Не найден в черном списке"); }

}

}

privatestaticvoid whitelist(CommandLine input)

{

IPAddress ip;

try

{

ip = IPAddress.Parse(input.arguments[0].parametr);

}

catch (FormatException) { Log.write("INPT", "Введенневерный IP длякомманды whitelist", ConsoleColor.Red); return; }

if (input.arguments[0].argument == "add")

{

Builder.Global.listAdd(Builder.Global.whitelist, ip);

Builder.Global.SaveList(Builder.Global.whitelist, "whitelist.json");

Log.write("FILE", ip, "Добавлен в белый список");

}

if (input.arguments[0].argument == "remove")

{

if (Builder.Global.listRemove(Builder.Global.whitelist, ip))

{

Builder.Global.SaveList(Builder.Global.whitelist, "whitelist.json");

Log.write("FILE", ip, "Убран из белого списка");

}

else { Log.write("FILE", ip, "Не найден в белом списке списке"); }

}

}

}

}

using Newtonsoft.Json;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Net.Sockets;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using Share;

namespace Server

{

publicstaticclassSocketServer

{

conststring LOGTYPE = "SOCK";

publicstaticIPHostEntry ipHost;

publicstaticIPAddress ipAddr;

publicstaticIPEndPoint ipEndPoint;

publicstaticstring status = "Отключен";

publicstaticSocket sListener;

publicstaticbool enabled = false;

publicstaticstring data;

publicstaticvoid Initialize(int port)

{

Log.write(LOGTYPE, "Создаю сервер на порте: " + port);

if (sListener != null)

{

try

{

sListener.Shutdown(SocketShutdown.Both);

}

catch (Exception) { }

}

ipHost = Dns.GetHostEntryAsync("localhost").Result;

ipAddr = IPAddress.Any;

ipEndPoint = newIPEndPoint(ipAddr, port);

Activate();

}

privatestaticvoid Connected(Socket \_handler)

{

IPAddress address = ((IPEndPoint)\_handler.RemoteEndPoint).Address;

if (Builder.Global.isInList(Builder.Global.blacklist, address))

{

Log.write(LOGTYPE, address, "Входящее подключение отклонено. Адреснаходитсявчерномсписке", ConsoleColor.Yellow);

SendResponse(\_handler, "blacklist");

return;

}

Log.write(LOGTYPE, address, "Входящееподключение", ConsoleColor.Magenta);

data = null;

byte[] bytes = newbyte[Int32.MaxValue / 100];

int bytesRec = \_handler.Receive(bytes);

data += Encoding.UTF8.GetString(bytes, 0, bytesRec);

Command recived;

try

{

recived = JsonConvert.DeserializeObject<Command>(data);

}

catch (Exception)

{

Log.write(LOGTYPE, "Неудаетьсяразобратьзапрос", ConsoleColor.Red);

return;

}

recived.qualify(address);

SendResponse(\_handler, recived.toAnswer);

}

privatestaticvoid SendResponse(Socket \_handler, string toSend)

{

byte[] msg = Encoding.UTF8.GetBytes(toSend);

\_handler.Send(msg);

}

publicstaticvoid Activate()

{

sListener = newSocket(ipAddr.AddressFamily, SocketType.Stream, ProtocolType.Tcp);

sListener.Bind(ipEndPoint);

enabled = true;

status = "Включен";

Thread potok = newThread(Listen);

potok.Start();

Log.write(LOGTYPE, "Серверуспешнозапущен", ConsoleColor.Green);

}

publicstaticvoid DeActivate()

{

Log.write(LOGTYPE, "Останавливаюсервер", ConsoleColor.Blue);

enabled = false;

status = "Отключен";

sListener.Dispose();

}

publicstaticvoid Listen()

{

while (enabled)

{

Socket handler;

sListener.Listen(10);

try

{

handler = sListener.Accept();

}

catch (SocketException) { break; }

Task.Run(() => Connected(handler));

// text();

}

}

privatestaticvoid text(string write)

{

Console.WriteLine("[NET]ПОЛУЧЕНОТВЕТ. СОДЕРЖАНИЕ:");

Log.write(LOGTYPE, "Клиент ответил: " + write);

}

}

}

using Newtonsoft.Json;

using Server;

using System;

using System.Collections.Generic;

using System.IO;

using System.Net;

using System.Text;

namespace Builder

{

publicstaticclassGlobal

{

publicstaticSchedule MainSchedule;

publicstaticList<IPAddress> blacklist = newList<IPAddress>();

publicstaticList<IPAddress> whitelist = newList<IPAddress>();

publicstaticGroup getGroup(Schedule input, string groupName)

{

foreach (Group g in MainSchedule.groupList)

{

if (g.name == groupName) return g;

}

returnnull;

}

publicstaticvoid SaveSchedule(Schedule input)

{

try

{

Global.MainSchedule = input;

File.WriteAllText("last.schd", MainSchedule.json());

Log.write("FILE", "Расписаниесохраненновфайл", ConsoleColor.Green);

}

catch (Exception)

{

Log.write("FILE", "Неудалосьзаписатьновоерасписаниевфайлlast.schd", ConsoleColor.Red);

}

}

publicstaticvoid LoadSchedule(string filename)

{

try

{

string read = File.ReadAllText(filename);

Global.MainSchedule = JsonConvert.DeserializeObject<Schedule>(Cipher.transcript(read));

Log.write("FILE", "Расписаниеуспешнозагруженосфайла: " + filename, ConsoleColor.Green);

}

catch (Exception)

{

Log.write("FILE", "Неудалосьзагрузитьфайл: " + filename, ConsoleColor.Red);

}

}

publicstaticbool isInList(List<IPAddress> list, IPAddress ip)

{

foreach (IPAddress addr in list)

{

if (addr.ToString() == ip.ToString()) returntrue;

}

returnfalse;

}

publicstaticvoid listAdd(List<IPAddress> list, IPAddress ip)

{

if (isInList(list, ip)) return;

list.Add(ip);

}

publicstaticbool listRemove(List<IPAddress> list, IPAddress ip)

{

foreach (IPAddress addr in list)

{

if (addr.ToString() == ip.ToString()) { list.Remove(addr); returntrue; }

}

returnfalse;

}

publicstaticList<IPAddress> LoadList(string filename)

{

List<IPAddress> result = newList<IPAddress>();

try

{

string read = File.ReadAllText(filename);

List<string> readO = JsonConvert.DeserializeObject<List<string>>(read);

foreach (string s in readO)

{

result.Add(IPAddress.Parse(s));

}

return result;

}

catch (Exception)

{

Log.write("FILE", "Непрочитатьфайл: " + filename, ConsoleColor.Red);

returnnewList<IPAddress>();

}

}

publicstaticvoid SaveList(List<IPAddress> list, string filename)

{

List<string> towrite = newList<string>();

foreach (IPAddress addr in list)

{

towrite.Add(addr.ToString());

}

try

{

string json = JsonConvert.SerializeObject(towrite);

File.WriteAllText(filename, json);

}

catch (Exception)

{

Log.write("FILE", "Неудалосьзаписатьвфайл " + filename, ConsoleColor.Red);

}

}

}

}

using Microsoft.Win32;

using Newtonsoft.Json;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Net;

using System.Text;

using System.Threading.Tasks;

namespace Builder

{

publicstaticclassFileInteraction

{

publicstaticstring filePath;

publicstaticstring fileName;

privatestaticSaveFileDialog saveFileDialog = newSaveFileDialog();

privatestaticOpenFileDialog openFileDialog = newOpenFileDialog();

publicstaticbool saveToFile(string toSave)

{

try

{

saveFileDialog.Filter = "Файлрасписания | \*.schd";

saveFileDialog.DefaultExt = "schd";

if (saveFileDialog.ShowDialog() == true)

{

fileName = System.IO.Path.GetFileName(saveFileDialog.FileName);

filePath = saveFileDialog.FileName;

File.WriteAllText(filePath, toSave);

Global.setSaveButton();

returntrue;

}

returnfalse;

}

catch (Exception)

{

filePath = "";

fileName = "";

Global.setSaveButton();

returnfalse;

}

}

publicstaticbool saveToFile(string toSave, string path)

{

try

{

fileName = System.IO.Path.GetFileName(path);

filePath = saveFileDialog.FileName;

File.WriteAllText(path, toSave);

Global.setSaveButton();

returntrue;

}

catch (Exception)

{

filePath = "";

fileName = "";

Global.setSaveButton();

returnfalse;

}

}

publicstaticbool saveToSavedPath(string toSave)

{

try

{

File.WriteAllText(filePath, toSave);

Global.setSaveButton();

returntrue;

}

catch (Exception)

{

returnfalse;

}

}

publicstaticstring openFile()

{

try

{

openFileDialog.Filter = "Файлрасписания | \*.schd";

openFileDialog.DefaultExt = "schd";

if (openFileDialog.ShowDialog() == true)

{

filePath = openFileDialog.FileName;

fileName = System.IO.Path.GetFileName(openFileDialog.FileName);

Global.setSaveButton();

returnFile.ReadAllText(filePath);

}

Global.setSaveButton();

return"";

}

catch (Exception)

{

filePath = "";

fileName = "";

Global.setSaveButton();

return"";

}

}

publicstaticstring openFile(string path)

{

try

{

filePath = path;

fileName = System.IO.Path.GetFileName(path);

Global.setSaveButton();

returnFile.ReadAllText(filePath);

}

catch (Exception)

{

filePath = "";

fileName = "";

Global.setSaveButton();

return"";

}

}

publicstructNetSettings

{

publicstring ip;

publicint port;

}

publicstaticvoid saveNetSettings()

{

NetSettings toSave = newNetSettings()

{

ip = NetFunctions.ip,

port = NetFunctions.portNumber

};

string json = JsonConvert.SerializeObject(toSave);

try

{

File.WriteAllText("netSettings.json", json);

}

catch (Exception) { }

}

publicstaticvoid loadNetSettings()

{

string json = File.ReadAllText("netSettings.json");

NetSettings loaded = JsonConvert.DeserializeObject<NetSettings>(json);

NetFunctions.ip = loaded.ip;

NetFunctions.portNumber = loaded.port;

string[] ip = loaded.ip.Split('.');

Global.syncForm.ip1.Text = ip[0];

Global.syncGroupForm.ip1.Text = ip[0];

Global.syncForm.ip2.Text = ip[1];

Global.syncGroupForm.ip2.Text = ip[1];

Global.syncForm.ip3.Text = ip[2];

Global.syncGroupForm.ip3.Text = ip[2];

Global.syncForm.ip4.Text = ip[3];

Global.syncGroupForm.ip4.Text = ip[3];

Global.syncForm.port.Text = loaded.port.ToString();

Global.syncGroupForm.port.Text = loaded.port.ToString();

Global.syncForm.Check.IsEnabled = true;

Global.syncGroupForm.RefreshGroups.IsEnabled = true;

}

}

}

using Newtonsoft.Json;

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Linq;

using System.Reflection;

using System.Text;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Documents;

namespace Builder

{

publicpartialclassWeek : Entry

{

public Week(string type)

{

name = type;

Monday = newDay("Monday", name);

Tuesday = newDay("Tuesday", name);

Wednesday = newDay("Wednesday", name);

Thursday = newDay("Thursday", name);

Friday = newDay("Friday", name);

Saturday = newDay("Saturday", name);

}

publicDay pick(string toFind)

{

switch (toFind)

{

case"Понедельник":

return Monday;

case"Вторник":

return Tuesday;

case"Среда":

return Wednesday;

case"Четверг":

return Thursday;

case"Пятница":

return Friday;

case"Суббота":

return Saturday;

}

returnnull;

}

publicvoid redraw()

{

Monday.reDraw();

Wednesday.reDraw();

Tuesday.reDraw();

Thursday.reDraw();

Friday.reDraw();

Saturday.reDraw();

}

}

publicpartialclassGroup : Entry

{

public Group(string \_name)

{

chuslutel = newWeek("ch");

znamenatel = newWeek("zm");

name = \_name;

commentary = "";

}

[JsonIgnore]

bool allowNewTemp = true;

publicvoid massReDraw()

{

TextRange toSet = newTextRange(Global.main.commentary.Document.ContentStart, Global.main.commentary.Document.ContentEnd)

{

Text = Global.selectedGroup.commentary

};

if (allowNewTemp)

{

allowNewTemp = false;

Schedule prepareToWire = newSchedule();

prepareToWire.fillMe();

string json = prepareToWire.json();

FileInteraction.saveToFile(json, "last.schd");

allowNewTemp = true;

}

chuslutel.redraw();

znamenatel.redraw();

}

}

publicpartialclassLector : Entry

{

public Lector(string \_lastName, string \_name, string \_middleName)

{

name = \_name; middleName = \_middleName; lastName = \_lastName;

}

publicoverridestring ToString()

{

string result;

result = lastName;

try

{

result += " " + name[0] + ". " + middleName[0] + ".";

}

catch (Exception)

{

result = lastName;

}

return result;

}

}

publicpartialclassLectionSwap

{

public LectionSwap(DateTime \_period, string \_para)

{

period = \_period;

para = \_para;

}

[JsonIgnore]

publicstring periodString

{

get

{

string result;

string correctDate(int input)

{

if (input < 10)

{

return"0" + input.ToString();

}

return input.ToString();

}

result = string.Format("{0}. {1}", correctDate(period.Day), correctDate(period.Month));

return result;

}

}

publicoverridestring ToString()

{

return periodString + " [" + para + "]";

}

}

publicpartialclassDay : Entry

{

public Day(string \_name, string weekParent)

{

week = weekParent;

name = \_name;

lectionList = newObservableCollection<Lection>();

ListBox list;

lookname = week + \_name;

list = (ListBox)Global.main.FindName(lookname);

list.ItemsSource = lectionList;

}

publicvoid add(Lection \_input)

{

Lection duplicate;

duplicate = gotDuplicate(\_input.lectionInterval);

if (duplicate != null) // есливднеужеестьпаравэтоминтервале

{

duplicate.auditory = \_input.auditory;

duplicate.lectionInterval = \_input.lectionInterval;

duplicate.lector = \_input.lector;

duplicate.name = \_input.name;

duplicate.swapList = \_input.swapList;

duplicate.setParents(week, name);

}

else { \_input.setParents(week, name); lectionList.Add(\_input); }

IOrderedEnumerable<Lection> look = lectionList.OrderBy(s => s.lectionInterval);

lectionList = Collection.ToCollectionFromNum<Lection>(look);

reDraw();

}

Lection gotDuplicate(int look)

{

foreach (Lection c in lectionList)

{

if (c.lectionInterval == look) return c;

}

returnnull;

}

publicvoid reDraw()

{

try

{

ObservableCollection<LectionControl> toShow = newObservableCollection<LectionControl>();

foreach (Lection c in lectionList)

{

try

{

toShow.Add(newLectionControl(c));

}

catch (Exception)

{

lectionList.Remove(c);

}

}

ListBox list = (ListBox)Global.main.FindName(lookname);

list.ItemsSource = toShow;

list.Items.Refresh();

}

catch (InvalidOperationException)

{

reDraw();

}

}

}

publicpartialclassLection : Entry

{

public Lection(string \_name, int \_lectionNumber, Lector \_lector, int \_auditory)

{

name = \_name;

auditory = \_auditory;

lectionInterval = \_lectionNumber;

lector = \_lector;

swapList = newObservableCollection<LectionSwap>();

}

publicvoid setParents(string weekname, string dayname)

{

day = dayname;

week = weekname;

}

publicLection Clone()

{

Lection toReturn;

toReturn = newLection(name, lectionInterval, lector, auditory)

{

swapList = swapList

};

return toReturn;

}

}

}

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Builder

{

publicpartialclassIntervalCollection

{

publicbool checkCorrect(Interval \_input)

{

TimeSpan empty = newTimeSpan();

int index = \_input.index - 1;

if (\_input.start == newTimeSpan() | \_input.end == newTimeSpan()) returnfalse;

if (\_input.start > \_input.end)

{

returnfalse;

}

if (last > 0)

{

for (int i = 0; i < index; i++) // проверка были ли раньше пары у которых время раньше

{

Interval c = timeList[i];

if (c != null)

{

if (\_input.start <= c.start)

{ // если у ранних начальная дата больше

if (c.end != empty & c.start != empty)

returnfalse;

}

if (\_input.end <= c.end)

{ // если у ранних конечная дата больше

if (c.end != empty & c.start != empty)

returnfalse;

}

if (\_input.start <= c.end)

{

if (c.end != empty & c.start != empty)

//если старт находиться внутри промежутка

returnfalse;

}

}

}

}

for (int i = index + 1; i < last - 1; i++)

{

Interval c = timeList[i];

if (c != null)

{

if (\_input.start > c.start)

{ // если у поздних начальная дата меньше

if (c.start != empty)

returnfalse;

}

if (\_input.end >= c.end)

{ // если у поздних конечная дата меньше

if (c.end != empty)

returnfalse;

}

if (\_input.end >= c.start)

{ // его конец не может залазить в его старт

if (c.end != empty & c.start != empty)

returnfalse;

}

}

}

returntrue;

}

publicbool setTime(Interval \_input)

{

if (checkCorrect(\_input))

{

try { timeList[\_input.index - 1] = \_input; }

catch (Exception) { timeList.Add(\_input); }

returntrue;

}

elsereturnfalse;

}

publicIntervalCollection Clone()

{

IntervalCollection result = newIntervalCollection()

{

timeList = Collection.ToCollection<Interval>(Collection.ToList<Interval>(timeList))

};

return result;

}

}

publicpartialclassInterval

{

public Interval(int \_index, TimeSpan \_start, TimeSpan \_end)

{

index = \_index;

start = \_start;

end = \_end;

}

public Interval(int \_index)

{

index = \_index;

}

publicoverridestring ToString()

{

string startH, startM;

string endH, endM;

if (start.Hours < 10) { startH = "0" + start.Hours; } else { startH = start.Hours.ToString(); };

if (start.Minutes < 10) { startM = "0" + start.Minutes; } else { startM = start.Minutes.ToString(); };

if (end.Hours < 10) { endH = "0" + end.Hours; } else { endH = end.Hours.ToString(); };

if (end.Minutes < 10) { endM = "0" + end.Minutes; } else { endM = end.Minutes.ToString(); };

return index + " (" + startH + ":" + startM + " - " + endH + ":" + endM + ")";

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

using Share;

using Newtonsoft.Json;

namespace Builder

{

///<summary>

///Логикавзаимодействиядля ServerSync.xaml

///</summary>

publicpartialclassServerSync : Window

{

TextBox[] ips;

string ip = "";

int portNumber;

public ServerSync()

{

InitializeComponent();

}

privatevoid Window\_Closing(object sender, System.ComponentModel.CancelEventArgs e)

{

e.Cancel = true;

Visibility = Visibility.Hidden;

}

privatevoid NumberValidationTextBox(object sender, TextCompositionEventArgs e)

{

Regex regex = newRegex("[^0-9]+");

e.Handled = regex.IsMatch(e.Text);

}

privatevoid port\_TextChanged(object sender, TextChangedEventArgs e)

{

int test;

try

{

test = Convert.ToInt32(port.Text);

}

catch (FormatException)

{

port.Text = "0";

return;

}

if (test <= 0) { port.Text = "0"; return; }

if (test > 65535) { port.Text = "0"; return; }

portNumber = test;

checkBoth();

;

}

void checkBoth()

{

if (portNumber <= 0) { Check.IsEnabled = false; port.Text = "0"; return; }

if (portNumber > 65535) { Check.IsEnabled = false; port.Text = "0"; return; }

if (string.IsNullOrWhiteSpace(ip)) { Check.IsEnabled = false; return; }

Check.IsEnabled = true;

NetFunctions.ip = ip;

NetFunctions.portNumber = portNumber;

}

privatevoid ip\_TextChanged(object sender, TextChangedEventArgs e)

{

ip = "";

TextBox target = (TextBox)sender;

int value;

try

{

value = Convert.ToInt32(target.Text);

}

catch (FormatException) { return; }

if (value < 0 || value > 255) { target.Text = "0"; return; }

try

{

foreach (TextBox t in ips)

{

int octat = Convert.ToInt32(t.Text);

ip += octat;

ip += ".";

}

ip = ip.Remove(ip.Length - 1);

checkBoth();

}

catch (Exception) { }

}

privatevoid Window\_Loaded(object sender, RoutedEventArgs e)

{

ips = newTextBox[4] { ip1, ip2, ip3, ip4 };

}

privatevoid Check\_Click(object sender, RoutedEventArgs e)

{

Schedule toSend = newSchedule();

toSend.fillMe();

bool answer;

try

{

answer = NetFunctions.compareSchedule(toSend);

}

catch (Exception)

{

Status.Content = "[Неудалосьсоединиться]";

return;

}

Send.IsEnabled = answer;

Get.IsEnabled = answer;

if (answer) Status.Content = "[Версии расписаний отличаються]";

else { Status.Content = "[Версии расписаний идентичны]"; }

}

privatevoid Send\_Click(object sender, RoutedEventArgs e)

{

Schedule toSend = newSchedule();

toSend.fillMe();

bool answer;

try

{

answer = NetFunctions.sendSchedule(toSend);

}

catch (Exception)

{

Status.Content = "[Неудалосьсоединиться]";

return;

}

if (answer) Status.Content = "[Расписаниеуспешнозагружено]";

else { Status.Content = "[Расписаниенеустановлено]"; }

}

privatevoid Get\_Click(object sender, RoutedEventArgs e)

{

Schedule recive = NetFunctions.getSchedule();

recive?.applyMe();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

using Share;

using Newtonsoft.Json;

using System.Collections.ObjectModel;

namespace Builder

{

///<summary>

///Логикавзаимодействиядля ServerSync.xaml

///</summary>

publicpartialclassServerSyncGroup : UserControl

{

TextBox[] ips;

string ip = "";

int portNumber;

ObservableCollection<string> groupList;

public ServerSyncGroup()

{

InitializeComponent(); ips = newTextBox[4] { ip1, ip2, ip3, ip4 };

}

privatevoid NumberValidationTextBox(object sender, TextCompositionEventArgs e)

{

Regex regex = newRegex("[^0-9]+");

e.Handled = regex.IsMatch(e.Text);

}

privatevoid port\_TextChanged(object sender, TextChangedEventArgs e)

{

int test;

try

{

test = Convert.ToInt32(port.Text);

}

catch(FormatException)

{

port.Text = "0";

return;

}

if (test <= 0) {port.Text = "0"; return; }

if (test > 65535) { port.Text = "0"; return; }

portNumber = test;

checkBoth();

;

}

void checkBoth()

{

if (portNumber <= 0) { port.Text = "0"; RefreshGroups.IsEnabled = false; GetGroup.IsEnabled = false; return; }

if (portNumber > 65535) { port.Text = "0"; RefreshGroups.IsEnabled = false; GetGroup.IsEnabled = false; return; }

if (string.IsNullOrWhiteSpace(ip)) { RefreshGroups.IsEnabled = false; GetGroup.IsEnabled = false; return; }

RefreshGroups.IsEnabled = true;

groupCombo\_SelectionChanged(null, null);

NetFunctions.ip = ip;

NetFunctions.portNumber = portNumber;

}

privatevoid ip\_TextChanged(object sender, TextChangedEventArgs e)

{

ip = "";

TextBox target = (TextBox)sender;

int value;

try

{

value = Convert.ToInt32(target.Text);

}

catch(FormatException) { return; }

if (value < 0 || value > 255) { target.Text="0"; return; }

try

{

foreach (TextBox t in ips)

{

int octat = Convert.ToInt32(t.Text);

ip += octat;

ip += ".";

}

ip=ip.Remove(ip.Length-1);

checkBoth();

}

catch (Exception) { }

}

privatevoid Window\_Loaded(object sender, RoutedEventArgs e)

{

}

publicvoid installOneGroup(Group input)

{

Global.selectedGroup = input;

Global.groupList = newObservableCollection<Group>() { input };

Status.Text = "[Загруженнагрупа: "+input.name+"]";

}

publicvoid GetGroup\_Click(object sender, RoutedEventArgs e)

{

try

{

if (Global.selectedGroup != null)

{

if (((String)groupCombo.SelectedItem) == Global.selectedGroup.name)

{

if (NetFunctions.compareGroup(Global.selectedGroup))

{

installOneGroup(NetFunctions.GroupDownload(groupCombo.SelectedItem asstring));

}

else { Status.Text = "[Вгрупе " + Global.selectedGroup.name + " нетизмений]"; }

}

else { installOneGroup(NetFunctions.GroupDownload(groupCombo.SelectedItem asstring)); }

}

else

{

installOneGroup(NetFunctions.GroupDownload(groupCombo.SelectedItem asstring));

}

}

catch (Exception) { Status.Text = "[Неудалосьполучитьгрупу]"; }

Global.selectedGroup.massReDraw();

}

privateasyncvoid RefreshGroups\_Click(object sender, RoutedEventArgs e)

{

ObservableCollection<string> Ingrouplist = null;

Status.Text = "[Идетобновлениесписка]";

Task<string> task =Task.Run(() => Load());

Status.Text = await task;

if(Ingrouplist==null) { groupCombo.Items.Clear(); return; }

groupCombo.ItemsSource = Ingrouplist;

groupCombo.IsEnabled = true;

string Load()

{

try

{

List<string> recived = NetFunctions.GroupListDownload();

groupList = Collection.ToCollection<string>(recived);

if (groupList.Count > 0) { Ingrouplist = groupList; }

return"[Загруженно " + groupList.Count + " груп]";

}

catch (Exception) { return"[Не удалось обновить список груп]"; }

}

}

privatevoid groupCombo\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

string selection = groupCombo.SelectedItem asstring;

if (string.IsNullOrEmpty(selection)) { GetGroup.IsEnabled = false; return; }

GetGroup.IsEnabled = true;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace Builder

{

///<summary>

///Логикавзаимодействиядля Para.xaml

///</summary>

publicpartialclassIntervalControl : UserControl

{

public IntervalControl()

{

InitializeComponent();

}

publicstaticreadonlyDependencyProperty TimeProperty =

DependencyProperty.Register("time", typeof(Interval), typeof(IntervalControl), newFrameworkPropertyMetadata(newPropertyChangedCallback(TimeChanged)));

publicstaticreadonlyDependencyProperty checkMethodProperty =

DependencyProperty.Register("checkMethod", typeof(voidMethod), typeof(IntervalControl), newFrameworkPropertyMetadata(newPropertyChangedCallback(TimeChanged)));

publicInterval time

{

get => (Interval)GetValue(TimeProperty); set => SetValue(TimeProperty, value);

}

publicdelegatevoidvoidMethod(IntervalControl sender);

publicvoidMethod checkMethod

{

get => (voidMethod)GetValue(checkMethodProperty); set => SetValue(checkMethodProperty, value);

}

privatestaticvoid TimeChanged(DependencyObject sender,

DependencyPropertyChangedEventArgs e)

{

IntervalControl window = (IntervalControl)sender;

window.pairLabel.Content = window.time.index;

window.startHText.Text = window.time.start.Hours.ToString();

window.startMText.Text = window.time.start.Minutes.ToString();

window.endHText.Text = window.time.end.Hours.ToString();

window.endMText.Text = window.time.end.Minutes.ToString();

window.checkMethod?.Invoke(window);

}

privatevoid NumberValidationTextBox(object sender, TextCompositionEventArgs e)

{

Regex regex = newRegex("[^0-9]+");

e.Handled = regex.IsMatch(e.Text);

}

privatevoid TextChanged(object sender, TextChangedEventArgs e)

{

string senderText = ((TextBox)sender).Text;

if (!string.IsNullOrEmpty(startHText.Text) && !string.IsNullOrEmpty(startMText.Text) && !string.IsNullOrEmpty(endHText.Text) && !string.IsNullOrEmpty(endMText.Text))

{

if (((TextBox)sender).Name.Last() == 'H')

{

if (Convert.ToInt32(senderText) > 23)

{

((TextBox)sender).Clear();

return;

}

}

else

{

if (Convert.ToInt32(senderText) > 59)

{

((TextBox)sender).Clear();

return;

}

}

TimeSpan start, end;

start = newTimeSpan(Convert.ToInt32(startHText.Text), Convert.ToInt32(startMText.Text), 0);

end = newTimeSpan(Convert.ToInt32(endHText.Text), Convert.ToInt32(endMText.Text), 0);

Interval toChange = newInterval(time.index, start, end);

time = toChange;

}

}

}

}

using System;

using System.Media;

using System.Windows;

using System.Windows.Threading;

publicpartialclassNotificationWindow : Window

{

public NotificationWindow()

{

InitializeComponent();

Dispatcher.BeginInvoke(DispatcherPriority.ApplicationIdle, newAction(() =>

{

System.Drawing.Rectangle workingArea = System.Windows.Forms.Screen.PrimaryScreen.WorkingArea;

System.Windows.Media.Matrix transform = PresentationSource.FromVisual(this).CompositionTarget.TransformFromDevice;

Point corner = transform.Transform(newPoint(workingArea.Right, workingArea.Bottom));

Left = corner.X - ActualWidth - 100;

Top = corner.Y - ActualHeight;

}));

SoundPlayer player = newSoundPlayer("Resources/notif.wav");

player.Load();

player.Play();

}

privatevoid Storyboard\_Completed(object sender, EventArgs e)

{

Close();

}

privatevoid Button\_Click(object sender, RoutedEventArgs e)

{

Close();

}

}using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Net.Sockets;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

namespace Builder

{

publicstaticclassNet

{

publicstaticIPHostEntry ipHost;

publicstaticIPAddress ipAddr;

publicstaticIPEndPoint ipEndPoint;

publicstaticSocket socket;

publicstaticvoid Initialize(string ipv4, int port)

{

FileInteraction.saveNetSettings();

ipHost = Dns.GetHostEntry("localhost");

ipAddr = IPAddress.Parse(ipv4); //Куда мы будем отправлять

ipEndPoint = newIPEndPoint(ipAddr, port); // Обєкт-цель

}

publicstaticbool connect()

{

socket = newSocket(ipAddr.AddressFamily, SocketType.Stream, ProtocolType.Tcp); // Сокет

socket.Connect(ipEndPoint);

System.Diagnostics.Debug.Write("================\nСокетсоединяетсяс " + socket.RemoteEndPoint.ToString());

return socket.Connected;

}

publicstaticstring Send(string mess)

{

byte[] responseBytes = newbyte[Int32.MaxValue / 100]; // Буфердляполученияданных

int responseBytesCount; // Количество байтов которые получим

if (socket == null || socket.Connected == false) connect();// Соединяем сокет с удаленной точкой

byte[] msg = Encoding.UTF8.GetBytes(mess); // сообщение для отправки в байтомов виде

socket.Send(msg); // Отправляем данные через сокет

System.Diagnostics.Debug.Write("\nОтправил");

responseBytesCount = socket.Receive(responseBytes); // Получаемответотсервераикол-вобайтов

Stop();

string result = Encoding.UTF8.GetString(responseBytes, 0, responseBytesCount);

if (result == "blacklist") { MessageBox.Show("Ваш IP-адресснаходитьсявчерномспискенасервере"); Global.RefreshTimer.IsEnabled = false; }

if (result == "whitelist") MessageBox.Show("Для внесение изменений в расписаний необходимо находиться в белом списке на сервере");

return result; // ответ в виде стринга

}

publicstaticvoid Stop()

{

socket.Shutdown(SocketShutdown.Both);

socket.Close();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Share;

using Newtonsoft.Json;

using System.Collections.ObjectModel;

namespace Builder

{

publicstaticclassNetFunctions

{

publicstaticstring ip;

publicstaticint portNumber;

#region Schedule

publicstaticbool compareSchedule(Schedule input)

{

string hash = input.hash();

Command execute = newCommand()

{

type = "ScheduleHashVerify"

};

execute.arguments.Add(hash);

Net.Initialize(ip, portNumber);

string answer = Net.Send(JsonConvert.SerializeObject(execute));

if (answer == "different") returntrue;

else { returnfalse; }

}

publicstaticbool sendSchedule(Schedule input)

{

string json = JsonConvert.SerializeObject(input);

Command execute = newCommand()

{

type = "ScheduleUpload"

};

execute.arguments.Add(Cipher.encryption(json));

execute.arguments.Add(input.hash());

Net.Initialize(ip, portNumber);

string answer = Net.Send(JsonConvert.SerializeObject(execute));

if (answer == "accepted") returntrue;

else { returnfalse; }

}

publicstaticSchedule getSchedule()

{

Command execute = newCommand()

{

type = "ScheduleDownload"

};

Net.Initialize(ip, portNumber);

string answer = Net.Send(JsonConvert.SerializeObject(execute));

if (answer == "bad") returnnull;

Command response;

response = JsonConvert.DeserializeObject<Command>(answer);

Schedule get = JsonConvert.DeserializeObject<Schedule>(Cipher.transcript(response.arguments[0]));

if (get.hash() != response.arguments[1]) { returnnull; } //error

return get;

}

#endregion schedule

#region group

publicstaticbool compareGroup(Group input)

{

string hash = input.hash();

Command execute = newCommand()

{

type = "GroupHashVerify"

};

execute.arguments.Add(input.name);

execute.arguments.Add(hash);

Net.Initialize(ip, portNumber);

string answer = Net.Send(JsonConvert.SerializeObject(execute));

if (answer == "different") returntrue;

else { returnfalse; }

}

publicstaticList<String> GroupListDownload()

{

Command execute = newCommand()

{

type = "GroupListDownload"

};

Net.Initialize(ip, portNumber);

string json = Net.Send(JsonConvert.SerializeObject(execute));

try

{

List<string> result = JsonConvert.DeserializeObject<List<string>>(json);

return result;

}

catch (Exception) { returnnull; }

}

publicstaticGroup GroupDownload(string groupName)

{

Command execute = newCommand()

{

type = "GroupDownload"

};

execute.arguments.Add(groupName);

Net.Initialize(ip, portNumber);

try

{

string json = Net.Send(JsonConvert.SerializeObject(execute));

Command response = JsonConvert.DeserializeObject<Command>(json);

string decoded = Cipher.transcript(response.arguments[0]);

Group result = JsonConvert.DeserializeObject<Group>(decoded);

if (result.hash() == response.arguments[1])

{

Global.intervals.timeList = JsonConvert.DeserializeObject<ObservableCollection<Interval>>(response.arguments[2]);

return result;

}

returnnull