МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДЕПАРТАМЕНТ ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ
ВИКОНАВЧОГО ОРГАНУ КИЇВСЬКОЇ МІСЬКОЇ РАДИ
(КИЇВСЬКОЇ МІСЬКОЇ ДЕРЖАВНОЇ АДМІНІСТРАЦІЇ)

КИЇВСЬКЕ ТЕРИТОРІАЛЬНЕ ВІДДІЛЕННЯ МАЛОЇ АКАДЕМІЇ НАУК УКРАЇНИ
(КИЇВСЬКА МАЛА АКАДЕМІЯ НАУК)

відділення: «Комп'ютерних наук»

секція: «Комп'ютерні системи та мережі»

базова дисципліна: математика

ВИКОРИСТАННЯ "ХМАРНИХ" ТЕХНОЛОГІЙ ТА "ІНТЕРНЕТУ РЕЧЕЙ" У КЕРУВАННІ НАВЧАЛЬНИМ ЗАКЛАДОМ

РОБОТУ ВИКОНАЛА:
слухач Липницька Поліна Денисівна
30 липня 2002 р.н., учениця 9 класу
КПНЛ №145, Печерський район
провулок Нестеровський 6, Київ
(095)-395-80-72
Федорів Любомир Атанасійович, методист,
заслужений вчитель України, КПНЛ №145,
(044)-287-11-49

КИЇВ–2016

ВИКОРИСТАННЯ “ХМАРНИХ” ТЕХНОЛОГІЙ ТА “ІНТЕРНЕТУ РЕЧЕЙ” У КЕРУВАННІ НАВЧАЛЬНИМ ЗАКЛАДОМ

ЗМІСТ

ВСТУП

Ця наукова робота знаходиться на стику декількох предметних галузей. Її прикладним застосуванням є автоматизація адміністрування навчальним закладом, а саме, такої функції, як управління відвідуваністю в школі. Питання пропускного режиму та безпеки у школі, безумовно, актуальне, особливо в наш час, про що йдеться у наказі Міністерства науки і освіти України №2 від 06.01.2015г. Проте, безпека та режим для школи – це, безумовно, не все, що цікавить адміністрацію.

Важливо не тільки контролювати відвідуваність учнів, але і робити це системно, накопичувати та аналізувати статистику, робити висновки, мати зворотній зв'язок, впливати на ситуацію. Створення пропускного режиму у школі недоцільно будувати за аналогією з промисловою установою, тому що специфічні потреби освітніх установ вимагають інших нових рішень.

Використання тут комп'ютерних систем та сучасних технологій інтернету надає великі можливості. Це і покращення комунікацій з учнями та їх батьками. І облік відвідування школярами окремих занять, гуртків, лекцій. А отже, обчислення популярності таких занять та рейтингу викладачів. Можливим стає і аналіз сезонної статистики пропусків з метою розпізнавати ситуації масових захворювань та об'єктивного прийняття рішень про карантин. При доданні певної технічної підтримки можливим є контроль психоемоційного стану учнів та багато іншого.

У зв'язку з цим, поставивши практичну мету - автоматизувати контроль відвідування в школі, ця наукова робота прийшла до вивчення передових комп'ютерних технологій. Метою дослідження стало використання комп'ютерних систем та мереж для поліпшення адміністрування шкільного навчального процесу. Теоретичною та практичною цілями були обрані вивчення можливостей технологій, які стали відомим трендом початку XXI століття, - "хмарних обчислень" та "інтернету речей", а також їх впровадження. В результаті робота була реалізована, як створення власної системи контролю, аналізу та управління відвідуваністю в школі із застосуванням цих сучасних технологій.

Це дослідження є новаторським. Безумовно, є різні підходи до контролю відвідуваності в школі - від паперових технологій, до застосування в школах (особливо комерційних) комплексів контролю доступу, скопійованих з бізнес-об'єктів. Останні не тільки досить дорогі, але що більш важливо - неефективні, тому що не є профільними для закладів освіти. Спеціальні системи для шкіл майже не розробляють, тому що школа не є прибутковим ринком збуту.

Проте, існує такий динамічний сегмент "розумних" будинків та офісів, де кількість нових технічних рішень зростає з темпами до 24% щорічно [1]. У цьому сегменті постійно виникають передові ідеї, багато з яких можуть бути перенесені на освітні установи. У нашому випадку - це дві ключові технології - "хмарні обчислення" та "інтернет речей". Розвиток таких технології відбувся завдяки:

- збільшенню обчислювальних і комунікаційних можливостей простих (побутових) пристроїв, таких як, наприклад, мобільні телефони;
- розширення простору мереж Wi-Fi, 3G, появі технологій NFC, що означає збільшення ступенів свободи при реалізації комунікаційних рішень;
- великої кількості “хмарних” платформ, де не вкладаючи грошей в "залізо" можна створити необхідні сервіси для централізованого керування "речами".

На жаль теорія та практика використання “хмарних” технологій та "розумних" речей у адмініструванні школи майже не розглядається. Тому у цієї роботі зроблено одне з перших намагань це зробити та були досягнуті позитивні результати.

Впровадження цих технологій дозволило створити шкільну систему адміністрування відвідуваності багатофункціональною та гнучкою. А також незалежною від будівельних робіт та витрат на техніку. Систему, навіть, не потрібно монтувати - немає дротів та роз'ємів. Комунікаційні функції беруть на себе бездротові мережі. Для зберігання бази даних про статистику відвідуваності, для її обробки та видачі інформації за запитами система, що запропонована, не потребує фізичних пристроїв, так як використовує "хмарну" інфраструктуру.

Для функцій обміну інформацією та взаємодії з людьми - класним керівником та адміністрацією, батьками учнів та учнями - були використанні технології зі сфери

"інтернету речей". Безпосередньо, приладами, що їх реалізують стали телефони Android (або будь-який смартфон). При цьому, 99% апаратів не потрібно було ні купувати, ні допрацьовувати, так як систему утилізувала телефони учнів, батьків і співробітників школи, на які було встановлено лише додаткове програмне забезпечення.

Таким чином, розпочавши з практичної мети - спростити і підвищити ефективність функцій адміністрації школи та класних керівників робота перейшла до вивчення технологій "хмарних обчислень" та "інтернету речей". Обрана правильна комбінація цих технологій дозволила на їх основі розробити нестандартне рішення для шкільної установи. Завершилася робота впровадженням і тестуванням в Київському Природничо-Науковому Ліцеї №145.

1. ТЕОРЕТИЧНА ЧАСТИНА

1.1. АНАЛІЗ РОЗВИТКУ «ІНТЕРНЕТУ РЕЧЕЙ»

«Інтернет Речей» - це широка мережа не обчислювальних приладів - звичайних речей (не комп’ютерів), підключених до інтернету, що збирають дані та обмінюються ними та можуть керуватися через інтернет. Вже сьогодні підключених до інтернету приладів стало більше, ніж людей. За прогнозами компанії Сіменс [1], до 2020 р. до інтернету буде підключено більш ніж 26 млрд «речей», тобто в багато разів більше чисельності людства.

Існує три складові частини «інтернету речей»:

1) прилади, що містять сенсори, контролери, датчики, реле. Це - фізичні прилади («речі»), котрі раніше не були призначені для підключення до мережі (телефони, побутова автоматика та електроніка), але зараз отримали можливість реєструватися у мережі, створюючи частину великої віртуальної спільноти «речей»;

2) мережі інтернету і локальні мережі. З урахуванням сучасних вимог це - насамперед бездротові мережі;

3) центри обробки інформації , як правило, в «хмарі» (Cloud computing). Ці центри виконують збір, зберігання, обробку та аналіз даних, що надходять від речей. А також виробляють і відправляють команди приладам для розумної взаємодії між собою «речей» відповідно до заданих алгоритмів.

«Інтернет речей» став активним стимулом розвитку мікроелектроніки. На думку компанії Cisco «інтернет речей» може бути каталізатором інновацій та інвестицій і поліпшення якості життя в усьому світі. Об'єм ринку роботизованих речей і «інтернет речей» до 2022 р. досягне $21 млрд і сукупного середньорічного темпу зросту в 30% [2].

«Інтернет Речей» на сьогоднішній день працює і в сферах управління, наприклад, з наступними:

- автомобілями (прогноз заторів, управління машиною без водія);
- офісами (розумний офіс включає в себе «розумне» освітлення);
- управління за допомогою смартфона або планшета телевізорами;
- об'єктами охорони (система безпеки) та медицини (медичні прилади).

Інтернет речей розвивається і в Україні, особливо у сфері банки, великі охоронні агентства, постачальники рішень моніторингу транспорту, рішень для точок продажів, перевізники, служби таксі, служби доставки, постачальники електроенергії, водоканали [3].

Потенційно обширною сферою застосування інтернету речей є освіта. На жаль, ця сфера знаходиться ледь не на останньому місці. Вивчення інтернету-речей звичайно додається до освітніх програм і спонсується комп'ютерними фірмами, зокрема Microsift [4]. Однак, використання «інтернету речей» для підвищення якості процесу навчання і адміністрування школи має незначні приклади.

Автор статті «Інтернет Речей в класну кімнату» [5] Макс Мейерс, пише, зокрема, про великі можливості використання «Інтернету речей» для контролю відвідуваності, мониторингу фізичного і емоційного стану учнів, відстеження психологічних реакцій, концентрації уваги и т.п. Реалізувати це автор пропонує з використанням спеціальних гаджетів, що активно з'являються на ринку,

вмонтованих в браслети. Проте, серед публікацій за даним напрямком ,практично, відсутні джерела, що аналізують можливість створення «інтернету речей» в школі, використовуючи вже існуючі загальнодоступні технічні засоби, такі як смартфони і планшети (найбільш поширені з них — системи Android).

Чому рівень проникнення «інтернету речей» в школах настільки низький? Основним бар'єром розвитку інтернету речей, причому, не тільки в освіті, можна вважати психологічний. Так як інвестиційних бар'єрів практично немає. Інтернет речей є, мабуть, однією з найбільш доступних, дешевих, легко масштабованих та гнучких у використанні технологій сучасності.

1.2. ВИВЧЕННЯ МОЖЛИВОСТЕЙ «ХМАРНИХ ТЕХНОЛОГІЙ»

Поява «інтернету речей» стала можливим завдяки розвитку «хмарних» технологій та обчислень. «Хмарні» технології — це технології зберігання і обробки даних, в яких комп'ютерні ресурси надаються користувачу через онлайн-сервіс.

Розвиток «хмарних» технологій достатньо стрімке явище. У 2015 році обсяг інвестицій в сфері «хмарних» технологій перевищив $ 40 млрд. Деякі експерти прогнозують, що до 2020 року цей показник досягне $ 240 млрд [6].

Зростання «хмарних» технологій обумовлено, з одного боку, розвитком технічної бази - наявністю досить потужних процесорів і розвиненого програмного забезпечення. З іншого боку, розвитку «хмарної» індустрії сприяє високий попит. У США від власного «заліза» відмовилися і перейшли у «хмари» понад 80% компаній [7]. Лідерами всесвітнього ринку ,безумовно, є Amazon (Amazon web services, AWS), Microsoft (Azure), Google Cloud Platform.

Попит на «хмарні» технології формується завдяки їх перевагам:доступності, мобільності, низької вартості, гнучкості, надійності та технологічності. Світові ціни на «хмарні» послуги постійно знижуються, а кількість та якість зростає. Основними варіантами використання «хмарної» інфраструктури є: приватна «хмара», публічна «хмара», суспільна та гібридна.

Обсяг світового ринку публічних хмарних послуг в 2016 р. досягне $ 204 млрд, що на 16,5% більше, ніж в 2015 р. . З такою динамікою «хмарних» технологій і зростанням «інтернету речей» цілком реалістичною може стати гасло «Internet of Everything» (як продовження«Internet of Things»).

Зростання ринку і різноманіття пропозицій на ньому забезпечує достатній вибір технічних рішень, заснованих на наступних моделях «хмарних» обчислень:

- SAAS — сервіс, як послуга, надає бізнес-додатки в форматі інтернет-сервісів;
- PAAS — платформа, як послуга, дозволяє використовувати «хмарну» інфраструктуру для розміщення програмного забезпечення;
- IAAS — інфраструктура, як послуга, що дає можливість використовувати хмарну інфаструктуру для самостійного управління ресурсами.

У компанії Oracle вважають, що до 2025 року 80% всіх IT-бюджетів компаній і держави підуть в «хмари». Поки що найбільше зростання спостерігається у моделі IAAS, клієнтами яких є IT директора бізнес компаній, що розвивають аутсорсинг. В майбутньому випереджаючим буде розвиток SAAS систем ,багато в чому, за рахунок сфери суспільних послуг і освіти зокрема.

Проте, зараз ситуація із застосуванням «хмар» в освітній сфері далека від ідеальної. Існують, безумовно, передові SAAS продукти, як приклад сервіс компанії Google - «Google class». Проте, використання його в Україні, на жаль, не стало поширеним. Використання «хмарних» технологій для адміністрування освітніх установ з метою підвищення їх ефективності та економії витрат, взагалі, обговорюється тільки як проект.

1.3. ОБГРУНТУВАННЯ ОБРАНОЇ МОДЕЛІ

З метою реалізації цілей, поставлених в цій роботі, були розглянуті та проаналізовані різні варіації «хмарних» моделей та зроблено вибір, що був обумовлений наступним:

- немає такого продукту, який вже реалізований в моделі SAAS та пропонував би готове рішення для навчального закладу для контролю відвідуваності, яке, крім того, інтегрувало б пристрої зі світу «інтернету речей»;
- не було сенсу займатися низькорівневими функціями адміністрування віртуальних серверів з метою самостійної побудови системи, на базі якої розгортуються сервера, а значить, не було сенсу обирати модель IAAS;
- функції, що виконуються у «хмарах», не є унікальними (незважаючи на новизну створеної системи), тому що вони базуються на стандартному керуванні базою даних та HTTP сервером та підтримуються в багатьох PAAS;
- є достатньо технологій для створення HTTP серверів, але був обраний фреймворк Nodejs з використанням мови програмування Javascript (так, як ця технологія і її інструменти мають відносно невисокий поріг проникнення);
- існує певний вибір платформ PAAS із підтримкою Nodejs для створення систем контролю «інтернету речей», багато з яких є безкоштовними;
- Weave - аналіз та можливо й використання найновішого PAAS сервісу від Google, було залишено для наступних етапів дослідження.

З урахуванням вищенаведеного, остаточним рішенням був вибір платформи OpenShift [8], яка припускає всі рівні розробки - від початкових програмістів до досвідчених корпоративних розробників. Платформа створена в публічному «хмарному» сервісі і заснованому RedHat (творцями однойменної версії Linux).

1.4. ДОСЛІДЖЕННЯ І ВИБІР ПЛАТФОРМИ ANDROID

Одними з лідерів ефективних і недорогих рішень, що включають в себе програмні і апаратні засоби, які використовуються в сфері «інтернету речей», є Android, Arduino та Raspberi Pi. Google, наприклад, нещодавно було створено і представлено оновлення до своєї платформи, яке спрощує підключення Android пристроїв до «інтернету речей» і «хмарних» сервісів [9].

Серед цих лідерів є відмінності. Якщо Raspberi Pi та Arduino відмінно підходять для тієї області застосування, де використовується робототехніка, управління рухомими об'єктами, контроль датчиків, включення-виключення реле. То для забезпечення людино-машинної взаємодії найбільш підходять пристрої Android.

Основні переваги операційної системи Android для використання у даному дослідженні вказані на малюнку 1.

У чому полягають настільки переконливі переваги Android ? Сама операційна система Android, розвинена роками за рахунок зусиль компанії Google, стала надзвичайно зручною, як для користувачів, так і для розробників. Створення програмних продуктів для Android здійснюється з використанням максимально зручного середовища розробки - Android Studio. Самі пристрої Android є повноцінною технічною базою для створення безліч пристроїв «розумного» будинку або офісу. Android має багато датчиків, включаючи гіроскопи, GPS, фотокамеру, яка може використовуватися для сканування штрих-кодів і бар-кодів.

Android пристрої з легкістю підключаються до будь-яких бездротових мереж. Android одним з перших просуває інтеграцію в свої пристрої системи Near Field Communication (NFC). Ця система отримала максимальне поширення в якості технології для здійснення безконтактних платежів. Однак NFC з великою ефективністю може використовуватися в будь-якій варіації «інтернету речей», зокрема, там де потрібне зчитування інформації з об'єкту, що знаходиться поблизу (до десятків сантиметрів) від пристрою Android.

Таким чином виникла ідея багатоцільового використання можливостей Android в системі управління відвідуваністю, як:

- первинного пристрою, що реєструє проходження контролю учнів (вхід і вихід з приміщення навчального закладу) шляхом сканування штрих кодів;
- та ж функція, але із застосуванням NFC технології (NFC мітки в кишенях або на браслетах учнів це - передавачі, а Android пристрій на пункті пропуску - приймач сигналу). Таке технічне рішення заплановано для реалізації на наступних етапах проекту;

- моніторингу інформації про присутність учнів класним керівником або адміністратором на екрані свого Android пристрою з використанням для цього спеціально розробленого програмного забезпечення – віджету;
- отримання інформації про присутність або відсутність учня в навчальному закладі батьками на Android пристроях, через інтеграцію в соціальних мережах (групах «Facebook» або «Вконтакті» класу);
- та ж функція, але із застосуванням SMS або Viber.

Важливим аспектом, що вимагає окремої уваги є безпека. Під безпекою, насамперед, розуміється ступінь захищеності інформації, що зберігається і передається пристроями, включеними до складу «інтернету речей». Неможливість для стороннього отримати або змінити цю інформацію.

Безумовно, використання «інтернету речей» саме по собі створює значні ризики кібер-атак [10] і визначає засоби мінімізації цих ризиків. З огляду на дані аспекти, пристрої Android були проаналізовані і визнані придатними з урахуванням того, що: пристрої мають парольний захист, використовують розмежування повноважень на рівні операційної системи (аналог UNIX), дозволяють організувати передачу інформації по закритих каналах з використанням VPN і багато іншого.

1.5. ПЕРСПЕКТИВИ ЗАСТОСУВАННЯ МОДЕЛІ В СФЕРІ ОСВІТИ

Сфера застосування «інтернету речей» і «хмарних» технологій в освіті взагалі та в адмініструванні навчального процесу зокрема невичерпна. У наведеній нижче таблиці зроблена спроба узагальнити найбільш актуальні способи застосування проаналізованих в попередніх розділах технологій.

Не дивлячись на велику сферу застосування сучасних технологій в освіті, в даній роботі упор був зроблений на аспектах дисципліни, безпеки і захисту здоров'я учнів. Для вивчення була обрана система адміністрування відвідуваності навчального закладу, а інші можливості і сфери застосування технологій (таблиця 1) були залишені для наступних етапів.

Таблиця 1 - Способи застосування «інтернету речей» і «хмарних» технологій в адмініструванні навчального процесу

Область застосування	**Технологічні особливості**	**Корисні функції**
Безпека	Реєстрація входу-виходу учня до школи NFC датчиком (сканером штрих коду)	• Немає сторонніх в школі; • Немає перекличок, економія часу; • Адміністрація проінформована он-лайн; • Запит, чи перебував учень в школі в конкретний час (батькам, правоохоронцям) з бази даних
	Контроль за переміщенням територією школи - NFC датчики в дверях класів	• Інформація про концентрацію учнів в окремих частинах школи, необхідна для оперативних рішень (пожежа); • Легко знайти конкретного учня в будівлі школи.
Дисципліна	Реєстрація входу-виходу учня до/зі школи	• Об'єктивність і доказовість при покаранні прогульників; • Аналіз зв'язку відвідувань і успішності, оцінювання учнів.
Здоров'я	Реєстрація входу-виходу учня до/зі школи	• Аналіз статистики присутності в період епідемії для прийняття рішень по карантину.

	Датчики з термометрами і пульсомірами в браслетах учнів	• Виявлення хворих, своєчасна допомога і ізоляція; • запобігання епідемії.
Якість навчання	NFC датчики в дверях класів	• Статистика відвідування занять; • Рейтинг викладачів (більше відноситься до ВНЗів і занять, що вільно відвідуються).
	Датчики з біологічними параметрами в браслетах учнів	• Реєстрація психологічного стану - нудно, розсіяна увага на уроках; • Миттєве інформування вчителя про стан сприйнятливості класу
	NFC мітки використовуються для голосування	• Учні можуть під час уроку мовчки торкатися мітками до кнопок «не зрозуміло», щоб сигналізувати вчителю
Платні сервіси	NFC мітки для оплати	• Оплата в їдальні за допомогою браслета; • Оплата платних занять, гуртків
Безкоштовні сервіси	NFC мітки для реєстрації	• Реєстрація при отриманні книг в бібліотеці; • Реєстрація при отриманні спортивного інвентарю

1.6. ЗАВДАННЯ ДЛЯ ПРАКТИЧНОГО ДОСЛІДЖЕННЯ

Таким чином в теоретичній частині був зроблений аналіз предмету дослідження - «хмарних технологій» і «інтернету речей», вивчена сфера застосування - адміністрування навчального процесу в школах, описані особливості і різновиди технологій, що використовуються, обрані конкретні інструменти досягнення поставленої мети - створення власної комп'ютерної системи управління відвідуваністю з використанням комп'ютерних систем і мереж.

Виходячи з цього були визначені наступні завдання для вирішення їх в практичній частині даного дослідження:

- аналіз вимог до програмного забезпечення, опис його функцій;
- розробка функціональної моделі програмного забезпечення;
- вибір архітектури програмного забезпечення та інструментів розробки;
- розробка алгоритмів програм, їх аналіз;
- установка, настройка і використання системи розробки програмного забезпечення - Android Studio;
- написання програмного коду (мови Java і JavaScript);
- дизайн інтерфейсної частини (екранів користувача, інтернет сторінок);
- остаточна верстка програм;
- реєстрація, настройка і використання сервісів OpenShift;
- розгортання серверного програмного забезпечення на «хмарі» OpenShift;
- розгортання клієнтського програмного забезпечення на смартфонах Android, його запуск і тестування;
- об'єднання всіх складових частин (смартфонів та сервера в єдину систему); тестування, усунення помилок;
- навчання користувачів;
- практичне використання, накопичення і аналіз досвіду.

2. ПРАКТИЧНА ЧАСТИНА

2.1.ОПИС ФУНКЦІЙ І СКЛАДОВИХ ЧАСТИН ПРОГРАМНОГО КОМПЛЕКСУ

Призначенням програмного комплексу є збереження інформації про проходження учнем пропускного пункту (поста охорони на вході в навчальному закладі) в базі даних на «хмарному» сервері. В подальшому дана інформація використовується для безлічі цілей: від простого моніторингу чисельності учнів у школі або окремому класі в конкретний момент часу або отримання простих звітів про тих, хто запізнився, до найскладніших функцій статистичної обробки інформації, що зберігається, передачі її адміністрації школи і батькам.

Для реалізації цієї мети програмний комплекс повинен складатися з наступних складових частин (малюнок 2):

- пристрою («речі»), що виконує читання інформації, що ідентифікує учня - на початковому етапі - це читання телефоном Android закріпленим на пункті пропуску в школу штрих (бар-) кодів. Штрих коди видавались учням у вигляді ламінованих карток (можливо у вигляді стікерів для наклейки на посвідчення учня, мобільні телефони і тп);
- пристрою («речі») для відображення поточної інформації про присутніх в будівлі школи учнів (кількість і фамільний склад) - телефоном Android класного керівника;
- пристрою («речі») для передачі батькам інформації, що учень запізнився або не прийшов для зв'язку - так само телефоном Android, який одержує повідомлення в соціальній мережі або у вигляді СМС;
- «хмарного» сервера, що виконує функції збереження в базі даних, взаємодії з усіма пристроями, виконання складних завдань, таких як статистична обробка інформації, видача звітів за запитами або регулярних (щоденних) звітів.

Цикл функцій системи представлений на малюнку 3.

Реєстрація учня починається з проходження пункту контролю, де учні самостійно або за допомогою співробітників охорони сканують штрих код. Залежно від того, чи з'явився учень вчасно або із запізненням він потрапляє на сервері в різні списки. В подальшому список тих, хто запізнився будуть представлений у вигляді звіту адміністрації. Учні, які не з'явилися в школу, виявляються в особливому списку. Батькам таких учнів система може відправити повідомлення в соціальну мережу або у вигляді СМС (вайбер-смс), а батьки отримують можливість зареєструватися на сервері і надати пояснення у вигляді повідомлення класному керівнику. Така система дозволяє попереджати втрату батьками і адміністрацією контролю над учнем в навчальний час. В кінці навчального дня учні реєструються на виході, щоб система зберегла в базі даних той період часу, протягом якого учень знаходився в школі.

Звичайно ідеальним було б використання не візуального сканування міток учнів, а читання їх NFC міток, проте таке технічне рішення було відкладено на перспективу. Вивчення NFC інтерфейсу в пристрої Android виходило за рамки цього етапу дослідження. Робота, в цілому, була розділена на кілька послідовних етапів (малюнок 4), виходячи зі складності і об’ємноті роботи, і відповідно, етапи 2 і 3 є планом для продовження даної роботи.

2.2. ВІДІБРАНІ ПРОЕКТНІ РІШЕННЯ. ОПИС КЛІЄНТСЬКОЇ І СЕРВЕРНОЇ ЧАСТИНИ

В основу програмного комплексу було покладено архітектура клієнт-серверного додатка. Структура і функції клієнтської і серверної частин представлені на малюнках 5 і 6.

Клієнтська частина являє собою мобільний додаток для пристрою Android. Розробка даної програми виконана з використанням мови програмування Java і платформи Android Studio. Лістинг програми представлений в додатку А.

В ході розробки даної програми був використаний принцип програмування «обробки подій», що є характерним для розробки під мобільні пристрої [11]. Подією для пристрою Android є отримання інформації, сканованою зі штрих коду учня, і у відповідь на дану подію додаток починає виконувати цикл дій описаних нижче.

Потрібно відзначити, що штрих код учня укладає в собі закодоване графічне представлення цілого числа в діапазоні від 1 до 400. Це число - номер учня в списку. Даний список відповідностей - прізвище, ім'я та номер - зберігається на сервері у вигляді файлу і використовується сервером для розшифровки. Тобто сервер після отримання від клієнта - Android пристрою - інформації у вигляді числа декодує цю інформацію і зберігає в базі даних вже символьні значення прізвища і імені учня.

Android пристрій отримує “подію” - проходження учня пункту контролю, яка викликає запис в відповідні змінні програми кода учня. Після цього пристрій Android формує пакет інформації для відправки на сервер. Це відбувається наступним чином. До коду учня додається дата і час його реєстрації, отриманого викликом системної функції. Потім код додається в чергу для відправки.

Таким чином, наступним архітектурним принципом, що впроваджений у клієнтській частині програми, є принцип “черги” (буфера), в який записуються повідомлення перед відправкою на сервер. Такий підхід був використаний для забезпечення надійності роботи [12]. Черга є пов'язаний список змінних. Список забезпечує додавання, видалення і читання змінних з початку і кінця (принцип FIFO). Так як пристрій Android пов'язано з Інтернетом бездротовим комунікаційним каналом Wi-Fi то можливі тимчасові втрати зв'язку або зниження якості зв'язку. Щоб інформація про реєстрацію учня в такі моменти не загубилася, вона залишається в черзі, звідки буде видалена лише якщо від сервера у відповідь прийде інформація про успішно прийняту порцію даних.

Серверна частина реалізована у вигляді програмного забезпечення, написаного на мові програмування JavaScript, а середовищем в якій дана програма запускається є Nodejs. Без Nodejs, що інтерпретує текст програми з мови JavaScript в команди операційної системи, серверна частина не працювала б взагалі.

Архітектурним рішенням серверної частини є використання REST-сервера для обробки запитів, що надходять від клієнта [13]. Зручну бібліотеку для реалізації сервера REST-сервера. містить в собі Nodejs, така бібліотека називається Express.js.

REST-сервер є як би перекладачем, який з одного боку, отримує через інтернет від клієнта інформацію у вигляді протоколу HTTP. З іншого боку, перетворює ці запити в програмний код і мову зрозумілу для бази даних, що зберігає основну інформацію. Основними видами запитів, які підтримуються даним REST-сервером, є команди GET і POST.

Запит POST є HTTP запитом, що містить вказівку сервера зберегти певну інформацію в базі даних. Такою інформацією є передана клієнтом - Android пристроєм - закодована інформація про проходження учнем пункту контролю. Запит POST в нашій системі має тільки один вид.

GET запитів навпаки безліч. GET - це команда клієнтом серверу надати певним чином оброблену (обрану, відсортовану) інформацію з бази даних. Так як уявлень інформації може бути безліч, то і запитів можливо аналогічну кількість: окремий GET запит на отримання кількості учнів в школі, окремий GET для отримання списку присутніх учнів, а також GET запити для видачі всіляких звітів.

На даному етапі дослідження мобільні пристрої класного керівника і батьків ще не підключені до сервера. Тому єдиною можливістю отримання інформації є звичайний браузер персонального комп'ютера. Але це - тимчасова ситуація і програма буде в наступному доопрацьована.

Крім перерахованого, існує і окремий модуль серверного програмного забезпечення, завданням якого є підготовка і відправка на електронну пошту класного керівника і завуча щоденних звітів про тих, хто запізнився і присутніх в школі учнів. Така програма написана як скрипт також на мови програмування JavaScript. Програма використовує вбудовану бібліотеку для відправки електронної пошти Email.js. А адреси отримувачів бере з конфігураційного файлу. Запуск цієї програми відбувається за розкладом. Розклад створено в самій операційній системі Linux, на якій розгорнута програма частина сервера. Це - відомий модуль Cron, що

дозволяє налаштувати запуск за розкладом будь-якої програми або системної функції (подібно будильнику). У нашому випадку система налаштована на генерацію і надсилання звітів в 13-00.

Лістинг серверної частини програмного забезпечення представлений в додатку Б.

2.3.РОЗГОРТАННЯ І ТЕСТУВАННЯ ПРОГРАМНОГО КОМПЛЕКСУ

Клієнтська частина системи була названа - мобільний додаток «Form Master Helper» («Допомога класному керівнику») версія 1.0. Для розгортання клієнтської частини програмного забезпечення був використаний телефон Samsung з програмним забезпеченням Android версії 2.3.3. Навіть така стара версія операційної системи телефону (найсучаснішою є версія 7) виявилася сумісною з розробленою програмою і забезпечила її стійку роботу. Для спрощення реєстрація мобільного додатка «Form Master Helper» в GooglePlay не використовувалася. Проте, в перспективі, після повного тестування і доведення системи до повноцінного функціоналу це стане виправданим і необхідним.

Зважаючи на відмову від використання GooglePlay, розгортання додатків відбувалося наступним чином. Спочатку програмний код був скомпільований і упакований (Android Studio робить це автоматично запуском команди «build») в пакетний файл з розширенням apk. Даний файл, будучи записаний будь-яким засобом в постійну пам'ять пристрою Android сприймається останнім, як інсталяційний пакет. У нашому випадку файл був переданий на Android з настільного комп'ютера за допомогою електронної пошти, як вкладення в лист. Телефон Android, відкривши вміст листа і розпізнавши вкладення, як інсталяційний пакет, автоматично встановив його. Після цього програма була готова до роботи, як будь-який інший додаток Android.

Розгортання програмного забезпечення сервера на «хмарному» сервісі PAAS від OpenShift відбувалося складніше. Спочатку на сайті OpenShift був створений

обліковий запис користувача і обраний безкоштовний пакет (обмеження пакета за швидкістю та обсягом віртуального диска виявилися достатні для невеликих масштабів застосування системи в даний час).

Потім необхідно було проінструктувати сервіс від OpenShift які «картриджі» (тобто пакети стандартного серверного програмного забезпечення). OpenShift повинен спочатку бути налаштований, щоб потім на основі цього встановленого програмного забезпечення можна було б розгорнути файли з програмами, створеними в даній роботі та запустити ці програми.

Виходячи з вимог проекту було обрано такі «картриджі» (налаштування «картриджів» зображені на малюнку 7):

- Node.js - фреймворк для розгортання серверної програми;
- Crone - утиліта для запуску програми відправки звітів на пошту за розкладом;
- Redis, як база даних, в якій зберігається інформація.

Після завершення цієї настройки, OpenShift надав зареєстрованому користувачу спеціальну утиліту (її ім'я - rhc) для того, щоб з її допомогою (запустивши її на настільному комп'ютері в командному рядку з відповідними параметрами) можна було б передати (вивантаживши - «upload») сервісу OpenShift файли з кодами програми. Спочатку перші версії, а в наступному нові редакції після поліпшення функцій та після усунення помилок. OpenShift, прийнявши завантажені файли, самостійно розгортав ці коди програм, запускав їх, а також забезпечив супровід, наприклад, записував у журнал інформацію про виникаючі помилки (в log файли).

Тестування системи відбувалося у вигляді циклу дій, зображених на малюнку 8. У ході тестування новоствореного і розгорнутого програмного забезпечення виникали збої та інші ситуації, що призводять до помилок. Зокрема, було виявлено та усунуто такі помилки:

- виявилося необхідним заздалегідь включити в телефоні Samsung опцію - «режим розробника», без якої програма не запускалася. Ця опція дозволяє розгортати на Android аматорські програми, які не зареєстровані на GooglePlay;

- так як OpenShift працює в часовому поясі сходу США, то звіти за програмою спочатку приходили з запізненням о 7 годині, а реєстрація учнів, які прийшли вчасно і тих, хто запізнився, взагалі відбувалася неадекватно. Ситуація була виправлена за допомогою введення в програму налаштувань, які корегують часовий пояс;
- безліч збоїв в роботі програми викликали помилки по причині невисокої якості сигналу в мережі Wi-Fi. Такі помилки потребували збільшення розмірів «черги» (описане в п.2.2) до розмірів, що враховує форс-мажор, коли список всіх учнів залишався б не переданим на сервер. Такий запас забезпечує 100% надійність програми.

Були виявлені також інші помилки, недоробки і недоліки дизайну програм, які перераховані на малюнку 7. Всі помилки були виправлені і програми доопрацьовані до стану готового до використання на практиці.

2.4. ВПРОВАДЖЕННЯ І ДОСВІД ПРАКТИЧНОГО ЗАСТОСУВАННЯ

Після тестування і усунення помилок, були підготовлені остаточні версії програм і був проведений фізичний монтаж реєстраційного приладу - телефону - на стійці охорони на вході в школу. При фізичному монтажі в розрахунок бралися економічність системи і надійність її використання. Ввечері, після закінчення навчального дня, пройшли перші успішні випробування, які показали працездатність і надійність змонтованої системи.

Протягом наступного дня учням 9-Б класу КПНЛ №145 були роздані картки реєстрації з ламінованого картону, на яких були роздруковані штрих коди. Учням було зроблено інструктаж і продемонстровано використання програми. Учням в цей день була надана можливість самостійно випробувати систему в будь-який час, так як до кінця дня генерація звітів була вимкнена. З наступного дня учні приходячи в школу здійснювали реєстрацію в системі в штатному режимі. І звіти почали надходити одержувачам.

Досвід застосування системи показав наступне:

- хоча сканування реєстраційних карток зі штрих-кодами здійснюється вбудованою фотокамерою телефону Android, не зовсім призначеної для такої мети, воно відбувається досить швидко і безпомилково. Швидкість сканування в середньому становить 1 секунду. Середня швидкість проходження учнем пункту реєстрації - 3 секунди;
- сказане вище не скасовує необхідності переходу до NFC на наступних етапах і ось чому: близько 30% учнів на жаль забувають зареєструватися в системі в поспіху, в останні хвилини до дзвінка або ігнорують реєстрацію в випадкку запізнення (сподіваючись уникнути попадання в список тих, хто запізнився). Тобто системою найактивніше користуються дисципліновані учні;
- впровадження використання модуля NFC може дозволити автоматично зареєструвати учня з NFC міткою в кишені чи сумці, що проходить повз. Для цього може знадобитися модернізована системи - установка підсилювачів сигналу на телефони Android. Але навіть без модернізації системи використання NFC дозволить прискорити реєстрацію в 2-3 рази, так, що це допоможе подолати психологічний бар'єр (небажання витрачати навіть секунди на реєстрацію при запізненні в школу);
- можливо буде також вмонтувати NFC мітки в значки школяра (ліцеїста) або браслети, що дозволить скоротити випадки навмисної і непоміченою чи уявної забудькуватості учнів;
- якщо такі значки або браслети стануть також обов'язковими для користування сервісами школи - їдальнею, бібліотекою, орендою спортивного інвентарю або обов'язковими для пропуску в кабінети з особливим режимом роботи: комп'ютерний клас або кабінет хімії (де зберігаються матеріальні цінності або небезпечні предмети), то носіння NFC міток перестане бути «хобі», а стане обов'язком.

ВИСНОВКИ

Зрозуміло, що система, розроблена в даному дослідженні, первісна версія, яка отримала назву «Помічник класного керівника» («Form Master Helper») є лише першим кроком у логічно послідовному процесі все більшого проникнення комп'ютерних та мережевих технологій, в тому числі «хмарних» обчислень і «інтернету речей» в освітній процес і його адміністрування.

В цілому, незважаючи на ранні стадії практичного використання, система була позитивно оцінена викладачами, перш за все керівником класу, бо ця система значно полегшує їх повсякденну роботу і дисциплінує учнів. У більшості учнів система також викликала інтерес і позитивні відгуки. І це при тому, що безліч функцій запланованих для системи на майбутнє знаходяться на початковому етапі, у нерозробленому стані. Дані функції планується ввести в роботу пізніше, після отримання достатнього досвіду від застосування системи протягом хоча б одного-двох навчальних семестрів.

СПИСОК ТЕРМЕНІВ

Android — операційна система і платформа для мобільних телефонів та планшетних комп'ютерів, створена компанією Google на базі ядра Linux.

Linux — загальна назва сімейства відкритих операційних систем.

Google Play — магазин додатків, що дозволяє власникам пристроїв з Android завантажувати і купувати різні додатки.

IoT (Інтернет речей**)** — це мережа, що складається із взаємозв'язаних фізичних об'єктів (речей) або пристроїв, які мають вбудовані датчики, а також програмне забезпечення, що дозволяє здійснювати передачу і обмін даними.

PAAS, IAAS, SAAS — моделі хмарних обчислювань (с.11-12).

Хмарні обчислення (Cloud Computing) — це модель забезпечення повсюдного та зручного доступу на вимогу через мережу до спільного пулу обчислювальних ресурсів, що підлягають налаштуванню.

NFC (Near Field Communication) — «зв'язок на невеликих відстанях» — технологія бездротового високочастотного зв'язку малого радіусу дії «в один дотик».

Nodejs — платформа з відкритим кодом для виконання високопродуктивних мережевих стосунків, написаних мовою JavaScript.

OpenShift — сервісна платформа для розробників програмного забезпечення.

HTTP — протокол передачі даних, що використовується в комп'ютерних мережах.

REST (Representational State Transfer, «передача репрезентативного стану») — підхід до архітектури мережевих протоколів, які забезпечують доступ до інформаційних ресурсів.

URL (Уніфіко́ваний лока́тор ресу́рсів) — адреса певного ресурсу в Інтернеті.

VPN (Віртуальна приватна мережа) — це логічна мережа, створена поверх інших мереж, на базі загальнодоступних або віртуальних каналів інших мереж (Інтернет).

СПИСОК ІНТЕРНЕТ ДЖЕРЕЛ

1. http://www.siemens.com/innovation/en/home/pictures-of-the-future/digitalization-and-software/internet-of-things-facts-and-forecasts.html
2. http://bda-expert.com/2016/10/robototehnika-stanovitsya-prioritetom/
3. http://www.epravda.com.ua/rus/publications/2015/08/13/554178/
4. http://hi-tech.ua/internet-veshhey-v-ukrainskih-shkolah/
5. https://www.edsurge.com/news/2015-03-28-connecting-the-classroom-with-the-internet-of-things
6. https://kontur.ru/articles/225
7. http://www.capital.ua/ru/specproject/34299-mirovoe-oblako-vplot-do-kontsa-2013-g-derzhalos-na-trekh-kitakh-amazon-amazon-web-services-aws-microsoft-azure-google-cloud-platform-uzhe-v-blizhayshee-vremya-na-rynke-mogut-izmenitsya-lidery
8. https://developers.openshift.com
9. http://news.finance.ua/ru/news/-/390894/google-predstavil-obnovlennyj-android-dlya-interneta-veshhej
10. https://servernews.ru/927477

ДОДАТКИ

А. Лістинг клієнтскої частини програмного забезпечення

(URL - https://github.com/Wolperting/FormMasterHelper_AndroidPpart)

```
package com.wolper.formmasterhelper;

import android.app.Activity;
import android.content.Intent;
import android.content.SharedPreferences;
import android.graphics.Color;
import android.os.AsyncTask;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.TextView;
import android.widget.Toast;
import com.google.zxing.integration.android.IntentIntegrator;
import com.google.zxing.integration.android.IntentResult;
import java.util.Date;
import org.apache.commons.validator.routines.UrlValidator;



public class FirstScr extends AppCompatActivity {

    private volatile boolean commingin=true;
    private TextView textView_invite;
    private final int SERVER_SETUP_REQUEST=25;
    private final int SERVER_SECURE_REQUEST=26;
    MainStorageSingleton mainStorageSingleton;
    HttpRequestTask httpTask;
    SharedPreferences sPref;


    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.first_menu, menu);
        return true;
    }


    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            //this menu item starts a new activity for server setting up
            case R.id.server_settings:
                Intent myIntent1 = new Intent(FirstScr.this, SecondScr.class);
                startActivityForResult(myIntent1, SERVER_SETUP_REQUEST);
                return true;
            case R.id.server_security:
                Intent myIntent2 = new Intent(FirstScr.this, ThirdScr.class);
                startActivityForResult(myIntent2, SERVER_SECURE_REQUEST);
                return true;
        }
```

```
    return super.onOptionsItemSelected(item);
}

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_first_scr);

    //CreateStorage
    mainStorageSingleton=MainStorageSingleton.getInstance();


    //Restore view content after phone changing or waking up
    textView_invite = (TextView) findViewById(R.id.text_scanned);
    if (savedInstanceState!=null) {
        commingin=savedInstanceState.getBoolean("inout", true);
    }

    //Set animation
    showQueueLength();
    Animation anim = AnimationUtils.loadAnimation(this, R.anim.anim_alpha);
    textView_invite.setAnimation(anim);


    //Create or restore AsyncAnction infinite cycle
    httpTask = (HttpRequestTask) getLastCustomNonConfigurationInstance();
    if (httpTask==null) {
        httpTask = new HttpRequestTask();
        httpTask.execute();
        restoreSettings();
    }
    httpTask.link(this);
}



//Setting button reaction
public void onClicButtons(View target) {
    switch (target.getId()) {
        case R.id.button_scan_out:
            setComming(false);
            break;
        case R.id.button_scan_in:
            setComming(true);
            break;
    }
    doScan();
}


//Save link to external AsyncTask
@Override
public Object onRetainCustomNonConfigurationInstance() {
    if (httpTask!=null) httpTask.unlink();
    return httpTask;
}


//Save configuration
@Override
public void onPause() {
    super.onPause();
```

```
        saveSettings();
    }


    //Save view content when phone stops or changes position
    @Override
    protected void onSaveInstanceState(Bundle bundle) {
        bundle.putBoolean("inout", commingin);
    }


    //Stop the infinite cycle on finishing the app
    @Override
    protected void onDestroy(){
        if (isFinishing())
            if (httpTask!=null) httpTask.cancel(false);
        super.onDestroy();
    }



    //Getting result from called activities
    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent intent) {

        if (resultCode != Activity.RESULT_OK)
            return;

        //Getting Server Setup result
        if (requestCode==SERVER_SETUP_REQUEST) {
            String s_srv=intent.getStringExtra("server");
            UrlValidator urlValidator = new UrlValidator(new String []
{"http","https"});
            if (!urlValidator.isValid(s_srv))
                    Toast.makeText(FirstScr.this, "Неверный формат адреса сервера",
Toast.LENGTH_SHORT).show();
                else {
                    mainStorageSingleton.server=s_srv;
                    Toast.makeText(FirstScr.this, "Сервер - "+s_srv,
Toast.LENGTH_SHORT).show();
                    httpTask.cleanError();
                }
        }

        //Getting Password Setup result
        if (requestCode==SERVER_SECURE_REQUEST) {
            String s_psw=intent.getStringExtra("password");
            mainStorageSingleton.password=s_psw;
            Toast.makeText(FirstScr.this, "Пароль - "+s_psw,
Toast.LENGTH_LONG).show();
        }


        //getting Scan Result
        IntentResult scanningResult =
IntentIntegrator.parseActivityResult(requestCode, resultCode, intent);
        if (scanningResult != null) {
            String scanContent = scanningResult.getContents();

            //showing Scan Result on the screen and sending to server
            sendScanResult(commingin, scanContent);
            showQueueLength();
```

```
        }
    }


    //Changing in-out mode
    private void setComming(boolean b) {
        commingin=b;
    }


    //Show Queue Length
    private void showQueueLength() {

textView_invite.setText(customAnimation(mainStorageSingleton.getQueue().size()));
    }


    //Scanninig barcode
    private void doScan(){
        IntentIntegrator scanIntegrator = new IntentIntegrator(FirstScr.this);
        scanIntegrator.initiateScan();
    }


    //Sending scan results to server
    private void sendScanResult(boolean commingin, String scanContent) {
        PlayLoad playLoad = new PlayLoad();
        playLoad.setCode(scanContent);
        playLoad.setInout(commingin?"in": "out");
        playLoad.setDate(new Date().getTime());
        mainStorageSingleton.getQueue().offer(playLoad);
        httpTask.cleanError();
    }



    //Save and restore settings persistently
    private void saveSettings() {
        sPref = getPreferences(MODE_PRIVATE);
        SharedPreferences.Editor ed = sPref.edit();
        ed.putString("fm_settings1", mainStorageSingleton.server);
        ed.putString("fm_settings2", mainStorageSingleton.password);
        ed.commit();
    }


    //Restore after waking up
    private void restoreSettings() {
        sPref = getPreferences(MODE_PRIVATE);
        mainStorageSingleton.server = sPref.getString("fm_settings1", "");
        mainStorageSingleton.password = sPref.getString("fm_settings2", "");
    }


    //Animation for status string
    private String customAnimation(int x){
        String s="";
        if (x==0) {s="Очередь пуста"; textView_invite.setTextColor(Color.GREEN);}
        if (x>0) {s=">"+x+"чел"; textView_invite.setTextColor(Color.RED);}
        if (x>2) s=">>>"+x+"чел";
        if (x>4) s=">>>>"+x+"чел";
        if (x>6) s=">>>>>"+x+"чел";
```

```
        return s;
    }


    //Nested class
    //Class for sending scan result (POST) to REST service asyncronously
    private class HttpRequestTask extends AsyncTask<Void, String, Void> {

        private FirstScr activity;
        private volatile boolean hasError;

        public void link(FirstScr activity){
            this.activity=activity;
        }

        public void unlink(){
            this.activity=null;
        }

        @Override
        protected Void doInBackground(Void... params) {
            while (true) {

                    //check if interrupted and finish if work is done
                    if (isCancelled())
                        if (mainStorageSingleton.getQueue().size()==0) return null;

                    PlayLoad playload=mainStorageSingleton.getQueue().poll();
                    if (playload==null) continue;
                    SendRest sendRest =
SendRest.initWithServerName(mainStorageSingleton.server).

setPassword(mainStorageSingleton.password).prepareForSend(playload);

                    //on any error we will return playload to the queue and signal
for error (only once)
                    if (!sendRest.senmMe()) {
                        mainStorageSingleton.getQueue().offer(playload);
                        publishProgress(sendRest.getError());
                    }
                    else hasError=false;
            }
        }

        @Override
        protected void onProgressUpdate (String ...v) {
            if (!hasError) Toast.makeText(FirstScr.this, "Ошибка сервера-"+v[0],
Toast.LENGTH_LONG).show();
            hasError=true;
        }

        public void cleanError(){
            hasError=false;
        }

    }

}
```

Б. Лістинг серверної частини програмного забезпечення

```
//modules
var express = require('express');
var app = express();
var bodyParser = require("body-parser");
var fs = require("fs");
var basicAuth = require('basic-auth-connect');
var fmh_dao = require('./fmh_dao.js');


//constants
var dirNAME='RECORDS/';
var nameFIN='_in.txt';
var nameFOUT='_out.txt';
var nameFLATE='_late.txt';
//settings for been late and reporting
var lateHr=8;
var lateMn=40;
var reportHr=16;
//security options
var password='passw';
var user='FMH';

//openShiftConst
var server_port = process.env.OPENSHIFT_NODEJS_PORT || 8082
var server_ip_address = process.env.OPENSHIFT_NODEJS_IP || '127.0.0.1'
//time zone corrector
var timeCorrHr=7;
var timeCorr=(3600000*timeCorrHr);

//statistics
//people counter
var count=0;
//array of checked in
var checkedIn=[];

//settings for express
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));
app.use(basicAuth(user, password));
app.engine('html', require('ejs').renderFile);
app.set('view engine', 'html');
app.set('views', __dirname+"/views");


//creating REST api
app.post('/reg', function (req, res) {
      var sdate=req.body.date;
    //validate input
      if (isNaN(sdate)) res.send('wrong input');
    //correcr time difference
      sdate+=timeCorr;
    //      console.log('reseived! '+req.body.code + ' '+req.body.inout+ ' '+(new
Date(sdate)).toString());
    //process and writing result to the file
        writeFile(req.body.code, req.body.inout, (new Date(sdate)).getHours(), (new
Date(sdate)).getMinutes());
```

```
        res.send('ok');
});


app.get('/count', function(req, res){
        res.send(''+count);
});


app.get('/status', function(req, res){
      fmh_dao.getAllFromRedis(function(arr) {
            res.render('index.ejs', {title:"Students list for "+getNowDate(),
students: arr});
      });
});


//starting server
var server = app.listen(server_port, server_ip_address, function () {
 console.log("Listening on localhost  and port %s...", server.address().port);
});




//writing to file
function writeFile(code, inout, hr, min){
    //check for error or missing data
    if (!code || !inout) return;
    if (hr>24 || min >60) return;
    if (hr<0 || min <0) return;
    if ((strcmp(inout,'in')!=0)&&(strcmp(inout,'out')!=0)) return;

    //creating text line of formated text
    function concatMe(code, inout, hr, min){
      var smin=min<10?'0':"";
      smin+=min;
      //find student name by code
      var scode=fmh_dao.findByname(code);
      return scode+" "+inout+" "+hr+":"+smin+"\n";
    }

    //creating a title for a table in the file
    function writeTitle(inout, hr, min){
      var sinout;
      //initiate new day counter
      initCount(inout);
      if (strcmp(inout,'in')==0)
          if (isLate(hr, min)) sinout='been LATE';
            else sinout='ckecked IN';
      else sinout='checked OUT';
      return 'The list of '+sinout+ ' students of '+getNowDate()+
              "\n\n ----------------------------------------------------\n";
    }


    //return if we already checked in
    if (ifCheckedIn(code, inout)) return;


    //creating the file name
```

```
    var nameF;
    if (strcmp(inout,'out')==0) nameF=dirNAME+getNowDate()+nameFOUT;
        else if (isLate(hr, min)) nameF=dirNAME+getNowDate()+nameFLATE;
      else nameF=dirNAME+getNowDate()+nameFIN;


    //checking for existence of the file
    fs.exists(nameF, function (exists) {
      if (exists) {
            fs.appendFile(nameF, concatMe(code, inout, hr, min), function (err, fd)
{if (err) console.log(err);});
      } else {
          fs.writeFile(nameF, writeTitle(inout, hr, min)+concatMe(code, inout, hr,
min), function (err, fd) {if (err) console.log(err);});
      }
      //do count students
      countStudents(inout);
      addtoChecked(code, inout);
      console.log('written to file '+nameF);
    });
}


//taking the current date for forming the file name
function getNowDate(){
    var date = new Date(new Date().getTime()+timeCorr);
    return date.getDate()+"_"+date.getMonth()+"_"+date.getFullYear();
}


//checking if one is late
function isLate(hr, min){
    return ((hr>lateHr) || ((hr==lateHr) && (min>lateMn)));
}


//helper for compering strings
function strcmp(a, b) {
    if (a.toString() < b.toString()) return -1;
    if (a.toString() > b.toString()) return 1;
    return 0;
}


//for counting students
function countStudents(inout){
    if (strcmp(inout,'in')==0) count++;
      else count--;
    if (count<0) count=0;
}

//set counter to null
function initCount(inout){
    if(strcmp(inout, 'in')!=0) return;
    count=0;
    checkedIn=[];
    //using redis
    fmh_dao.deleteAllFromRedis();
}

//add/remove to/form checked in array
function addtoChecked(code, inout){
```

```
    if (strcmp(inout, 'in')==0) {
      checkedIn.push(code);
      //using redis
      fmh_dao.storeInRedis(code);
      return;
    }
    var ind = checkedIn.indexOf(code);
    if (ind<0) return;
    checkedIn.splice(ind, 1);
    //using redis
    fmh_dao.deleteFromRedis(code);
}

//check if is already cheched in/out
function ifCheckedIn(code, inout){
    if (strcmp(inout, 'in')===0) return (checkedIn.indexOf(code)>=0);
    return (checkedIn.indexOf(code)<0);
}
```