Увод

Cloud Computing[1] навлиза неусетно в ежедневието на хора и все повече се превръща в неизменна част от живота ни. Този процес на интегриране започва с появата на Youtube, Facebook, Google Docs, Concur Travel & Expense и други SaaS[2] решения. Повечето потребители на тези усулуги не правят разлика между SaaS[2] и статичен Website[3]. За разлика от тях, разработчиците на облачни приложения ежедневно се сблъскват с предизвикателства и трудности, свързани с процесите на имплементация, интегриране, Application Lifecycle Management[4] и други.

Първото изпитание на което са подложени софтуерните инжeнери е инсталирането на всички необходими инструменти преди да започне процеса на разработване. Примери за това са специфичните за проекта Runtime[5], IDE[6], Build System[7], Framework[8], Server[9], Source Control System[10], SDK[11] и други. Следва цикълът на разработка, писане на тестове и продуктивен код. За да се провери дали текущата имплементацията работи коректно се използва тестова система (най-често локален Server[9]). След като всички тестове се изпълняват успешно и продуктивния код е готов, следва да се изпробва, дали програмата може да работи не само в тестовата среда, но и в условията на реалната система. Преди да се изпробва поведението на програмата в реалната среда, трябва да се мине през процес на Build[12] на приложението и всички необходими артефакти. Следващата стъпка е Deploy[13] в реалната система.

Тези стъпки накратко описват как протича един пълен цикъл от процеса на разработване, тестване, Build[12] и Deploy[13] на дори и най-простото облачно приложение. Най-често само един цикъл не стига за да се завърши програмното задание. Така процесът на работа от еднократен се превръща в итеративен. В повечето случаи за завършването на един пълен цикъл е необходимо значително време, което в комбинация с итеративния подход за решаване на поставената задача, води до драстично понижен Turnaround Time[14] и многократно увеличeн Overhead[15].

Текущата дипломна работа има за цел да представи ползите от използването на съвременни технологии като Cloud Computing[1] и WebIDE[16], в съчетание с добре познати концепции като In-System Development[17] и RAD[18]. Симбиозата на тези технологии и концпеции води до значително улесняването на софтуерните инженери при разработването на облачни приложения, максимален Turnaround Time[14] и минимален Overhead[15].

Мотивация

Ползите от избора на WebIDE[16] пред стандартно Desktop IDE[19] са повече от една. Най-очевидната полза е, че изискванията към компютрите, използвани за разработка, намаляват драстично (скорост на изчисленията, оперативна памет, твърд диск и други). Единственото, от което имат нужда разработчиците, е добра интернет връзка и съвременен Browser[20]. Само това подобрение води до следните резултати - намалени оперативни разходи, разходи за амортизация и морално изхабяване. Друг не толкова очевиден плюс за софтуерните фирми, е липсата на време за първоначална инсталация на всички необходими инструменти и настройки на работната среда. Това води до минимален Overhead[15] преди започване на разработване.

Съчетанието между WebIDE[16] и In-System Development[17] води до значително предимство спрямо стандартния модел на работа. Липсата на времена за Build[12] и Deploy[13] ускорява допълнително работния процес. Програмирането директно върху реална/работна среда елиминира грешките, които се получават при транспортиране на готовия продукт от тестовата система към реалната среда. Всичко това е възможно благодарение на принципите взети от програмирането на микроконтролери и логически схеми работеща върху платка, така наречения In-System Programming[21] модел.

Комбинацията на горните технологии и концепции с RAD[18] техники и принципи като използването на Template[22], Wizard[23] и други, значително намаляват Turnaround Time[14] и ускоряват процеса на имплементация. RAD[18] техниките до голяма степен елиминират писането на Boilerplate Code[24] и спомагат за концентрация на усилията върху съществената бизнес логика.

Цел

Целта на дипломната работа е проектирането и реализирането на модули за RAD[18] инструменти за In-System Development[17] на облачни приложения. Реализираните модули трябва да могат да се използват в процеса на разработката на облачно приложение.

За постигането на поставената цел, в дипломната работа са дефинирани следните задачи:

* Да се направи обзор и сравнителен анализ на съществуващи WebIDE[16].
* Проектиране и интегриране на модули за избраното WebIDE[16].
* Създаване на облачно приложение с избраното WebIDE[16].

Обзор и сравнителен анализ на съществуващи решения

Обявената от Eclipse[25] инициатива за създаване на индустриален стандарт за разрабоване в облака, показва че сме достигнали до нов етап в използването на инструменти за създаване на софтуер. В момента се намираме в началото на преход от използване на Desktop IDE[19] и On-Premises[26] софтуер към WebIDE[16] и On-Demand[27] услуги (<https://www.eclipse.org/org/press-release/20141027_cloud_initiative.php>).

В настоящата дипломна работа ще бъдат разгледани следните проекти, част от които участват в ECD[28] - инициатива за създаване на индустриален стандарт за разработване в облака:

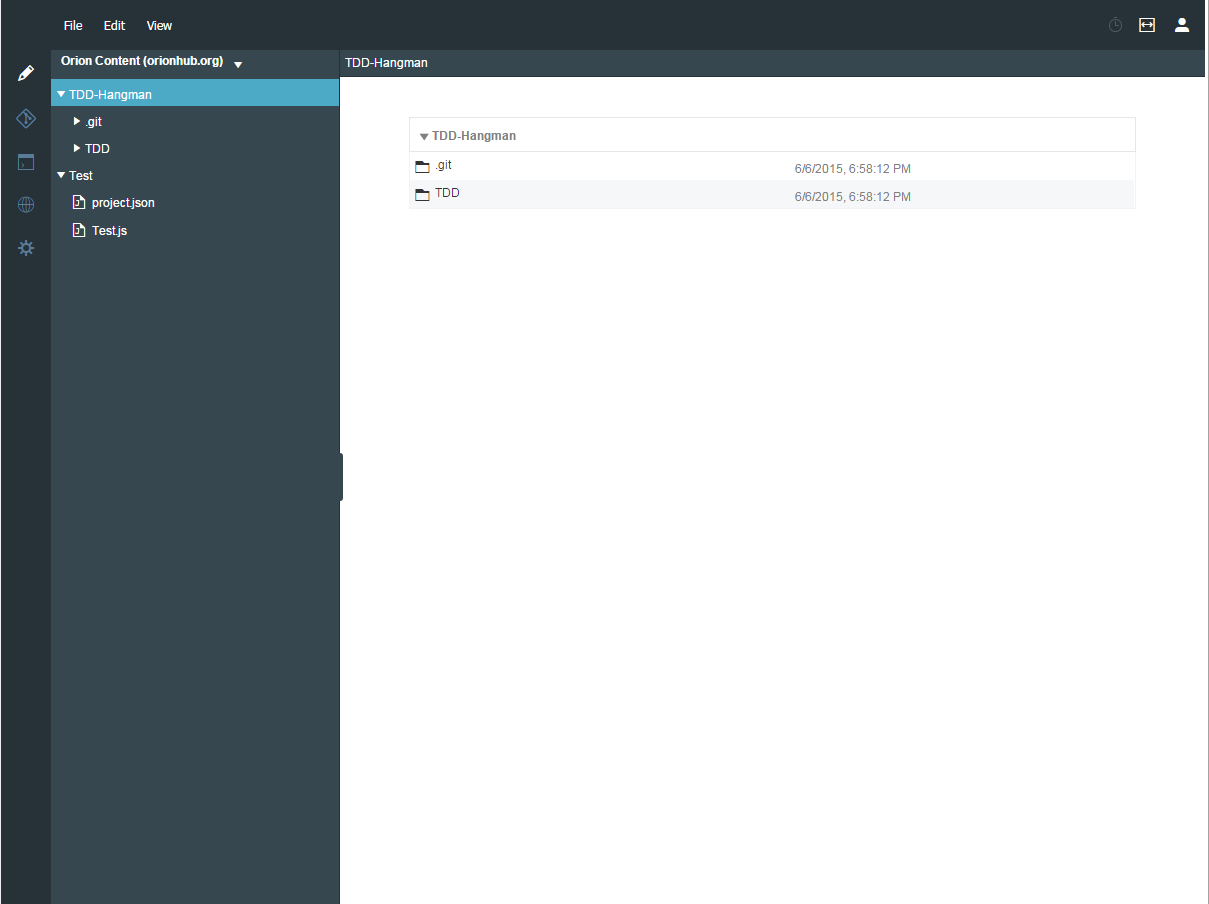
* Eclipse Orion (IBM)
* Eclipse Flux (Pivotal)
* Eclipse Che (Codenvy)
* Eclipse Dirigible (SAP)
* Cloud9 (Cloud9 IDE, Inc)

Eclipse Orion

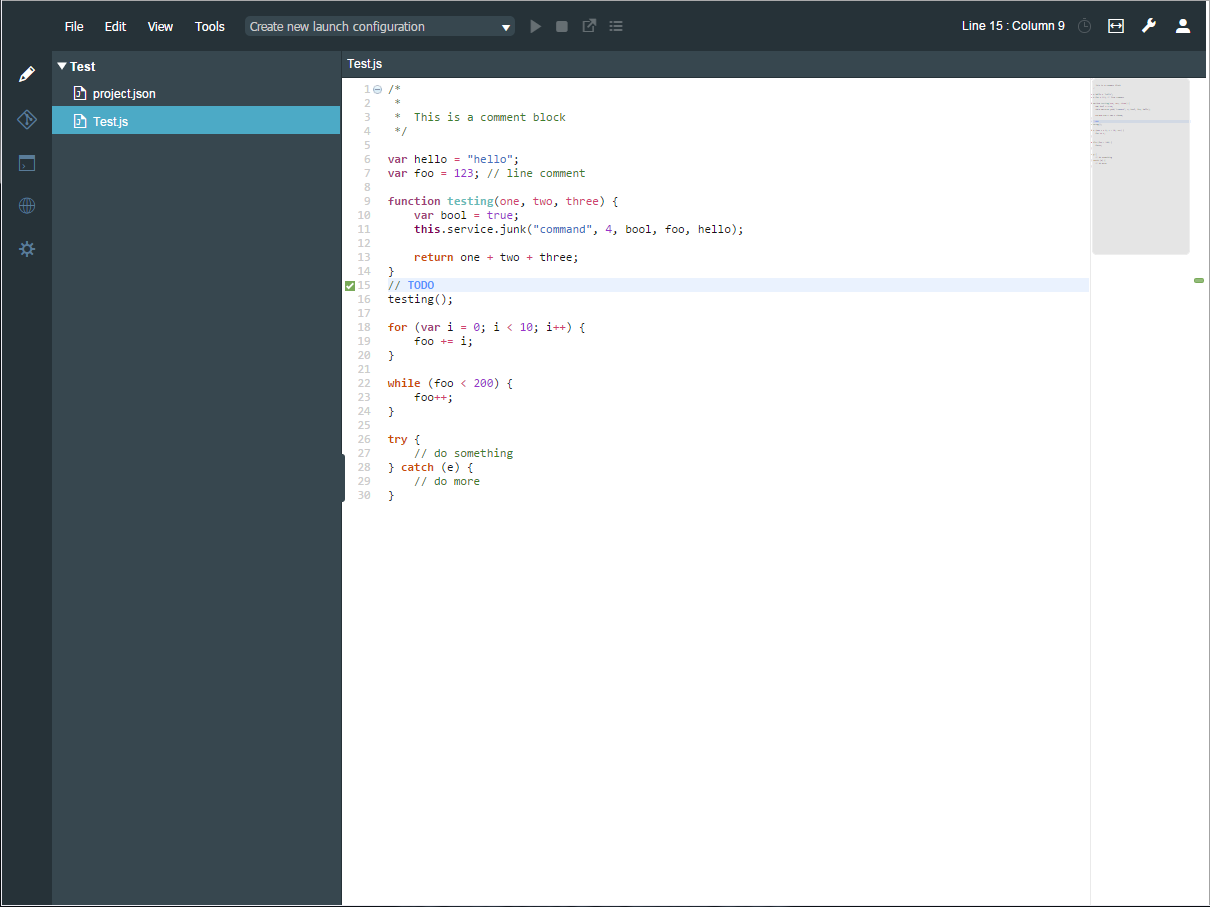
“Eclipse Orion” е Open Source[29] проект предимно разработен от “IBM”, който предлага онлайн редактор за JavaScript и други динамични езици.

Основните му предимства са, че позволява лесна конфигурация и интеграция на други облачни услуги както и лесно писане и добавяне на Plugins[30] с нови функционалности. Eclipse Orion позволява лесен Cross-Site Workflow[31] и разнообразие от Integration Points[32]. Поради добрата си модуларизация и Loose Coupling[33], части от проекта, като редактора, могат лесно да се преизползват в други проекти.

Най-големите му недостатъци са, че липсват собствени Runtimes[5], APIs[34], Templates[22] и Wizards[23]. Заради липсата на тези основни звена, Eclipse Orion е по-подходящ за обединяваща точка на различни облачни услуги, а не толкова като самостоятелна среда, в която протичат всички стъпки по разработка, настройка, конфигурация и Monitoring[35] на облачни приложения.



Фиг. 1 Работен плот на проекти в Eclipse Orion



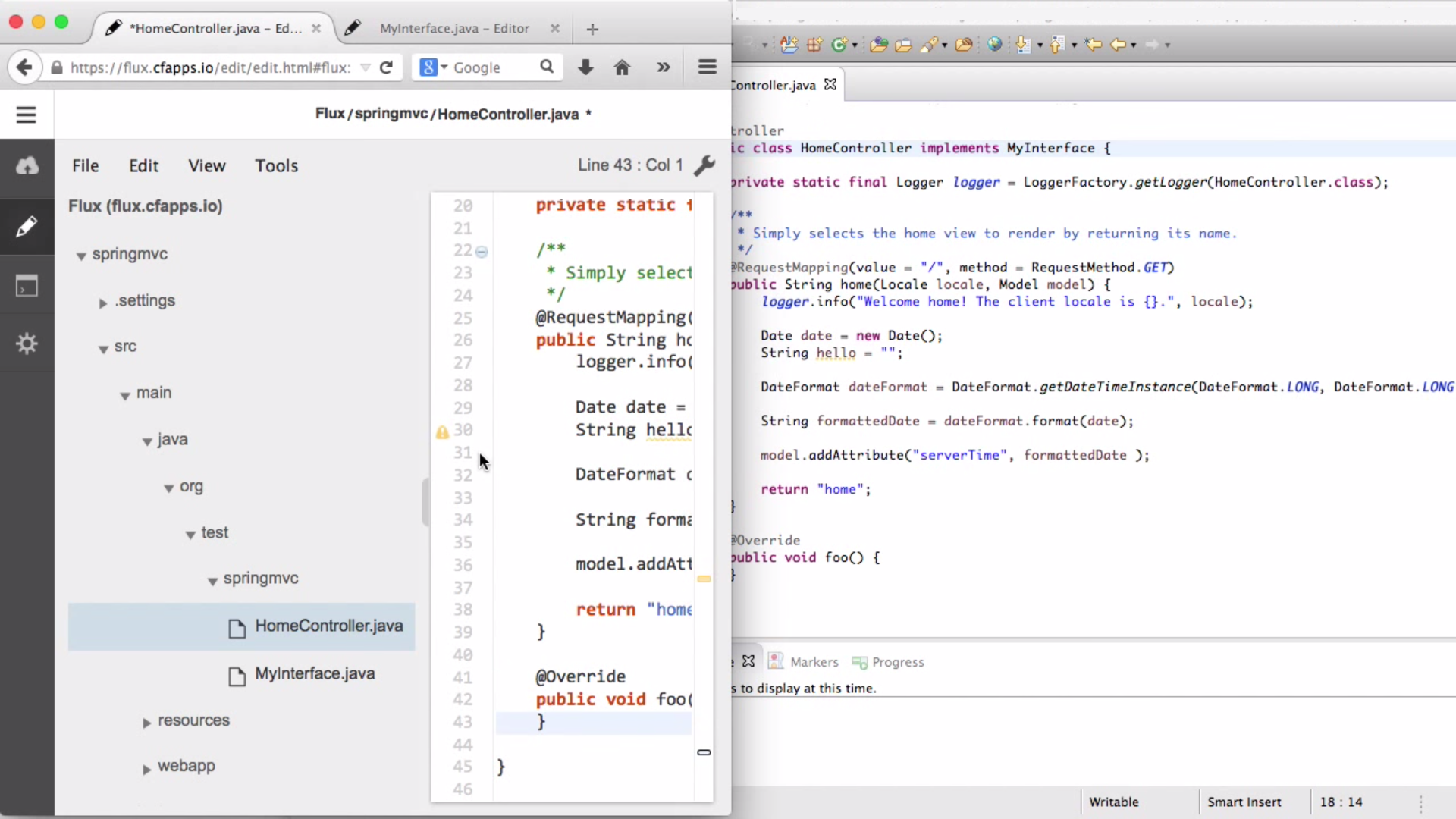
Фиг. 2 Редактор на изходен код на Eclipse Orion

Eclipse Flux

“Pivotal” е компанията, която е с основен принос за създаването на проекта “Eclipse Flux”. Мисията на Open Source[29] проекта е, да позволи плавен преход, за разработчиците, от Desktop IDE[19] към WebIDE[16].

Предимството на този проект е, че позволява лесна синхронизация на проекти от Desktop IDE[19], като по този начин софтуерните разработчици могат да изберат, кои неща да правят с Desktop IDE[19] и кои с WebIDE[16]. Тази функционалност е възможна благодарение на архитектура за дистрибутирано изпращане/получаване на съобщения, липса на спицифика относно езика на програмиране и изпращане/получаване на съобщения в дистрибутираната среда в реално време. Други плюсове на проекта са използването на Micro Services[36] и интеграцията с Eclipse Orion.

Съществени недостатъци са, липсата на собствени Runtimes[5], APIs[34], Templates[22], Wizards[23] и други. Целта на проекта е да бъде интегриран в екосистема, която предлага липсващите функционалности.



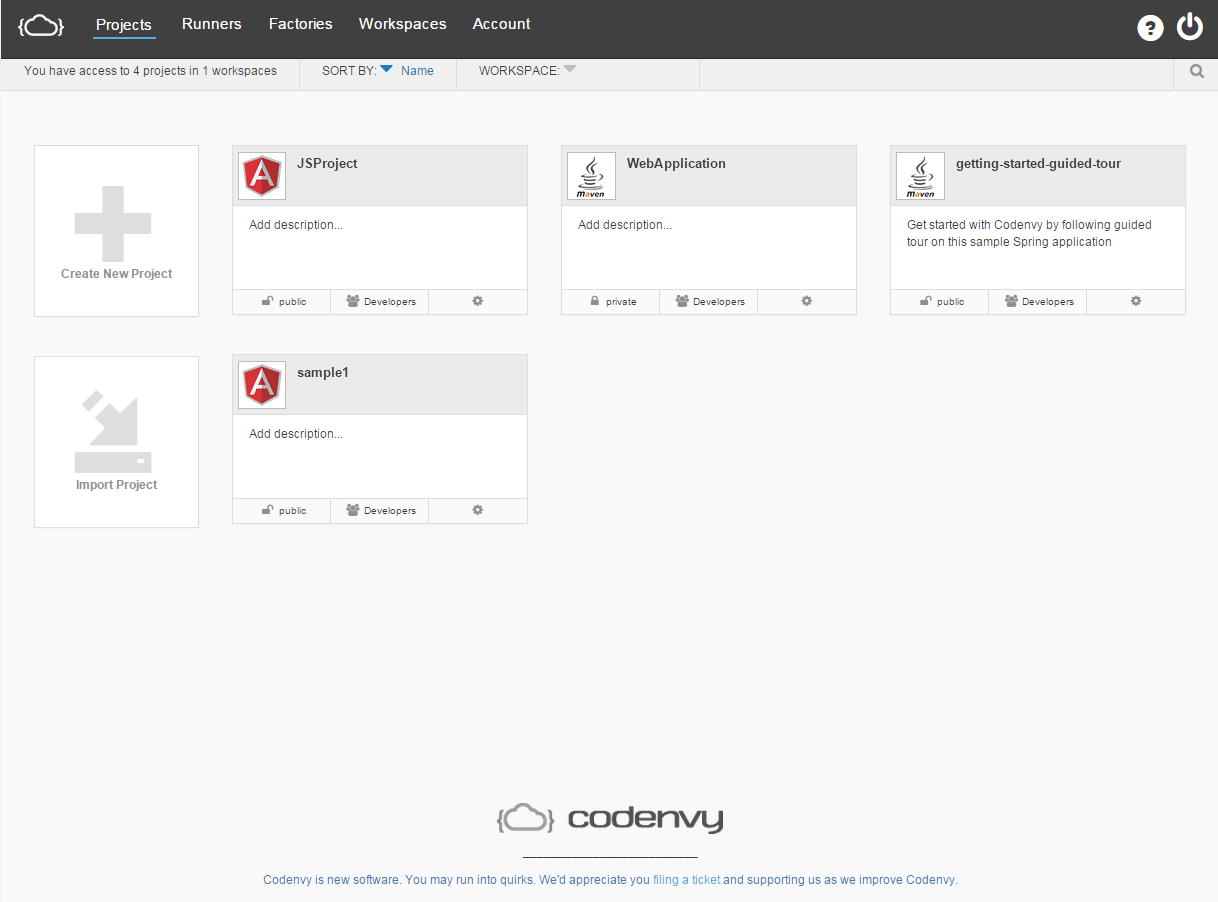
Фиг. 3 Синхронизиране на проект от Eclipse Desktop IDE в Eclipse Flux

Eclipse Che

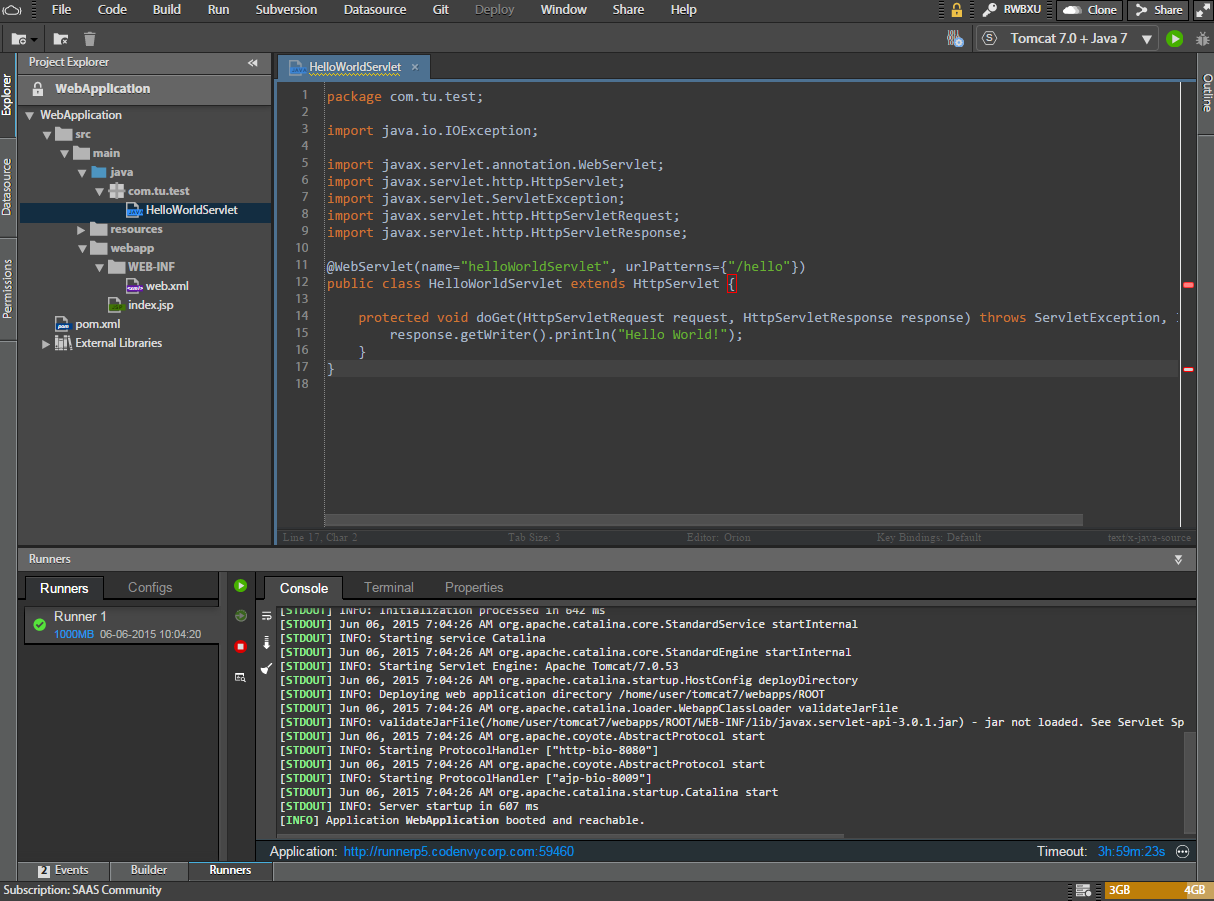
“Eclipse Che” е името на Open Source[29] проекта за “cloud IDE”(WebIDE[16]), предимно разработен от компанията “Codenvy”.

Голямото разнообразие от Runtimes[5] е освноното му предимство, някои от тях са: Java, Android, C/C++, JavaScript, PHP, Ruby и Go. Освен тях, Eclipse Che предлага и повечето известни Build Systems[7] като Maven[37], Ant[38], Grunt[39] и други. В основата на много добрата контейнерна модулоризация е поставен Docker[40]. Той позволява на Eclipse Che да предостави много на брой и напълно изолирани един от друг, Runtime[5], Build System[7] и виртуални машини. Благодарение на всичко това, Eclipse Che успява да автоматизира процесите по настройка на работната среда и необходимите инструменти, като ги пренася в облака.

Освновен недостатък на това IDE[6] е, че стандартният модел на разработка е запазен. Той е същият както при On-Premises[26] софтуера, което носи със себе си нисък Turnaround Time[14]. Използването на този модел на работа налага наличието на фази като Build[12], Deploy[13], стартиране на отделна виртуална машина, стартиране на отдалечен сървър и други. Друг голям минус е липсата на Templates[22] и Wizards[23], които биха ускорили процеса на разработване.



Фиг. 4 Начална страница на Eclipse Che



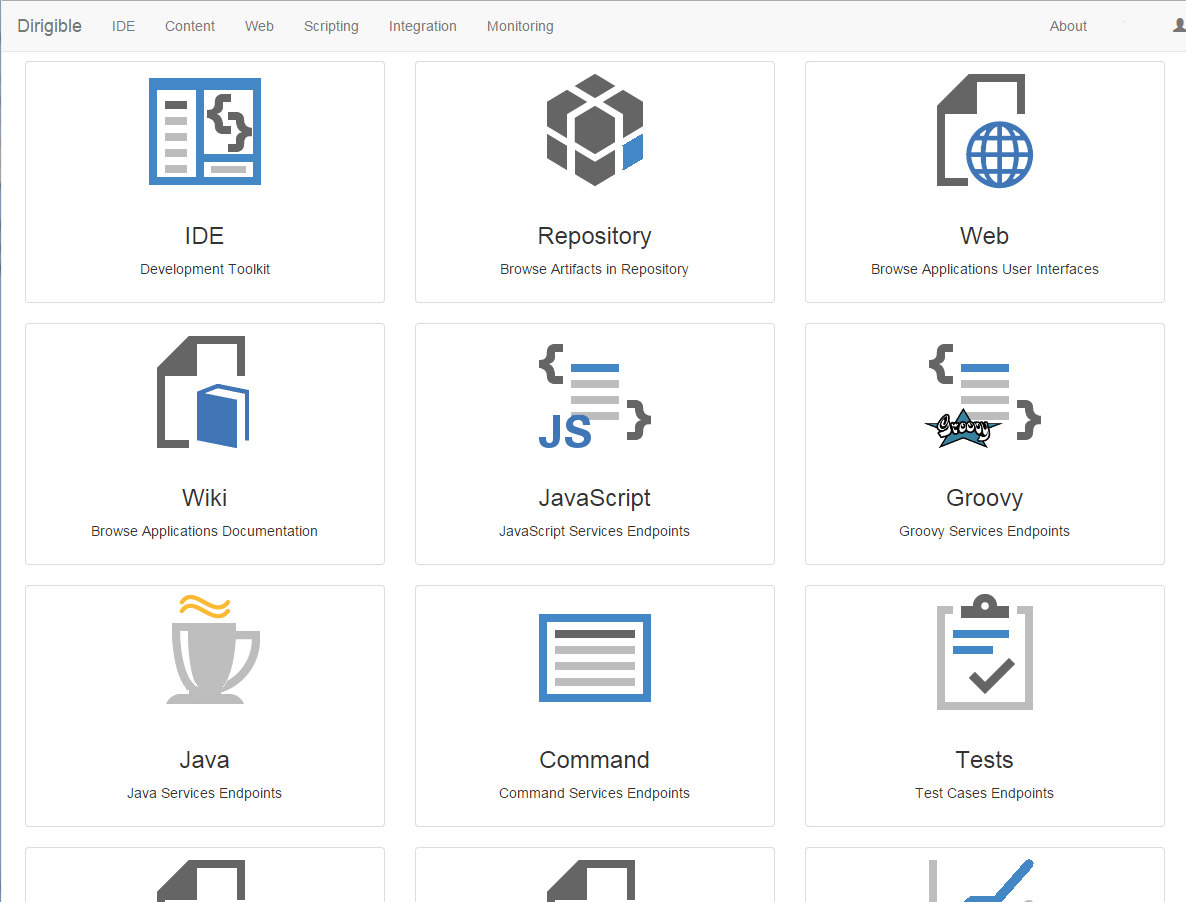
Фиг. 5 Работен плот на проект в Eclipse Che

Eclipse Dirigible

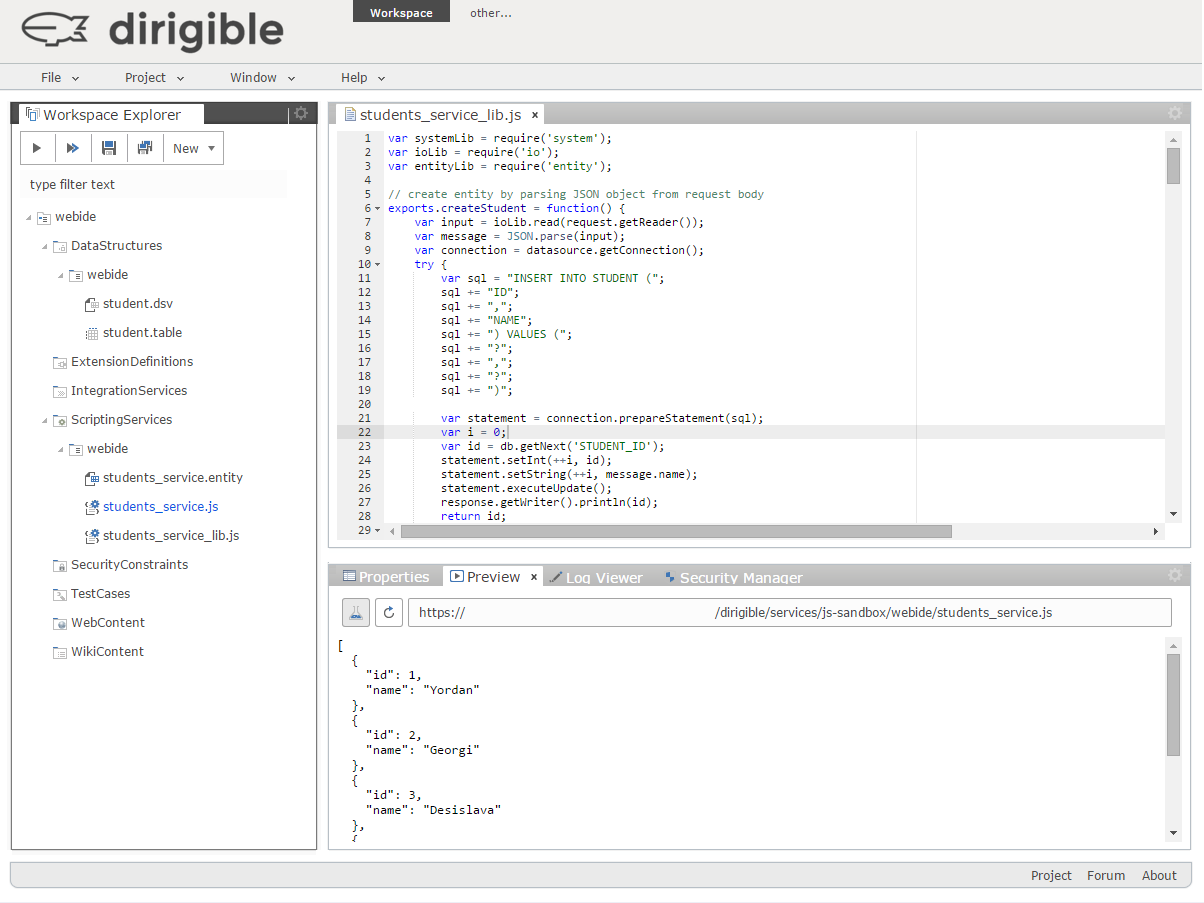
“SAP” е компанията с основен принос за създаването на Open Source[29] проекта “Eclipse Dirigible”. Проектът предлага IDEaaS[41] услуга (WebIDE[16]) за In-System Development[17] на облачни приложения.

Най-големите предимства на проекта са възможностите за In-System Development[17], RAD[18] чрез Templates[22] и Wizards[23], висок Turnaround Time[14], нисък Overhead[15], липса на време за Build[12] и Deploy[13], Runtime[5] за Java, JavaScript, Groovy, Ruby, както и лесното транспортиране на проекти от една система в друга. Друг плюс е графичния интерфейс, който до голяма степен прилича на Eclipse IDE, това значително улеснява разработчиците, които за първи път използват IDEaaS[41] услугата. Проектът предлага възможности за Offline[42] работа като инсталиранието на Plugins[30] за Eclipse IDE или работата с локален сървър. На последно място, но не и по-важност е лесното Extensibility[55] на проекта, чрез писане на Server-side[54] и Client-side[56] Plugins[30].

Няколко минуса към оценката на проекта са липсата на добър Code Completion[43] и Single Sign-On[44] опции за вписване чрез социални мрежи като Facebook, Github и други.



Фиг. 6 Начален екран на Eclipse Dirigible



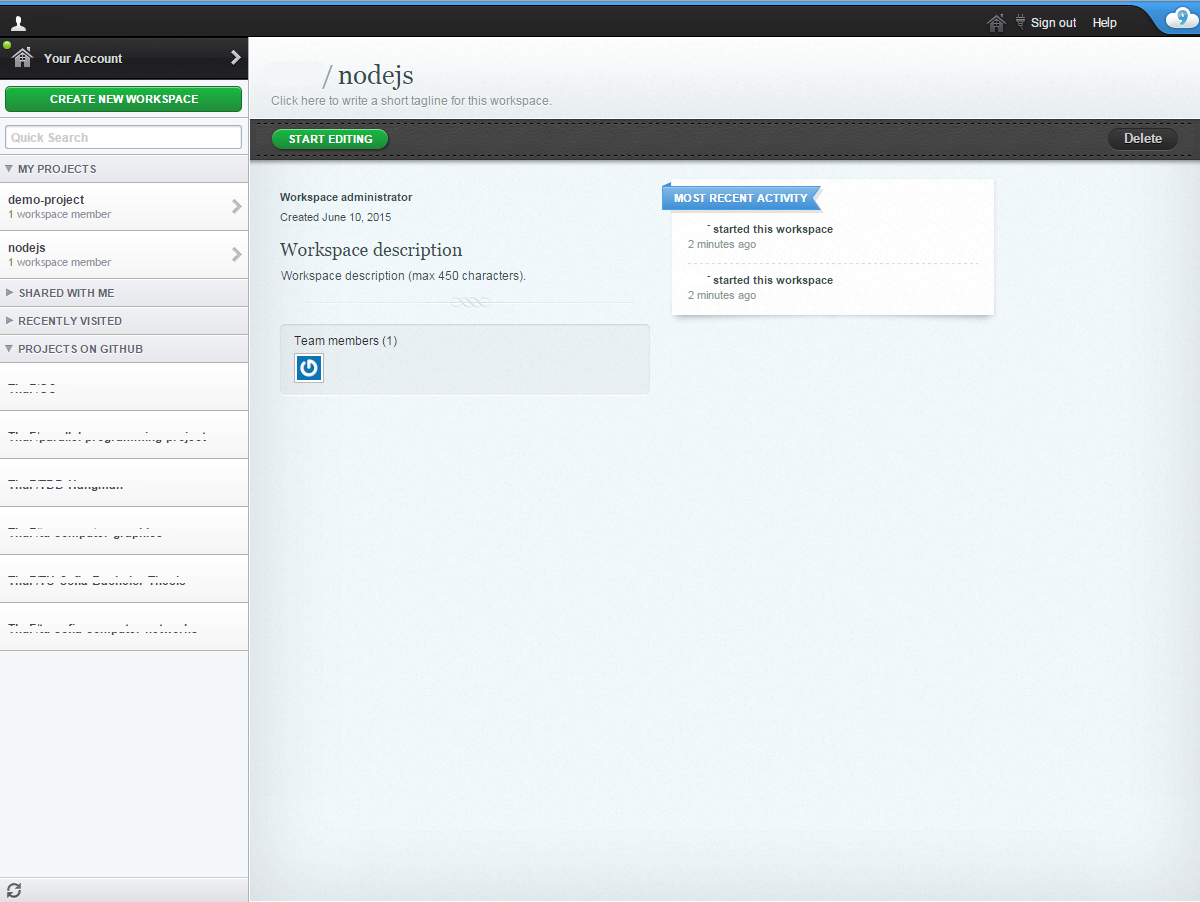
Фиг. 7 Работен плот на Eclipse Dirigible

Cloud9

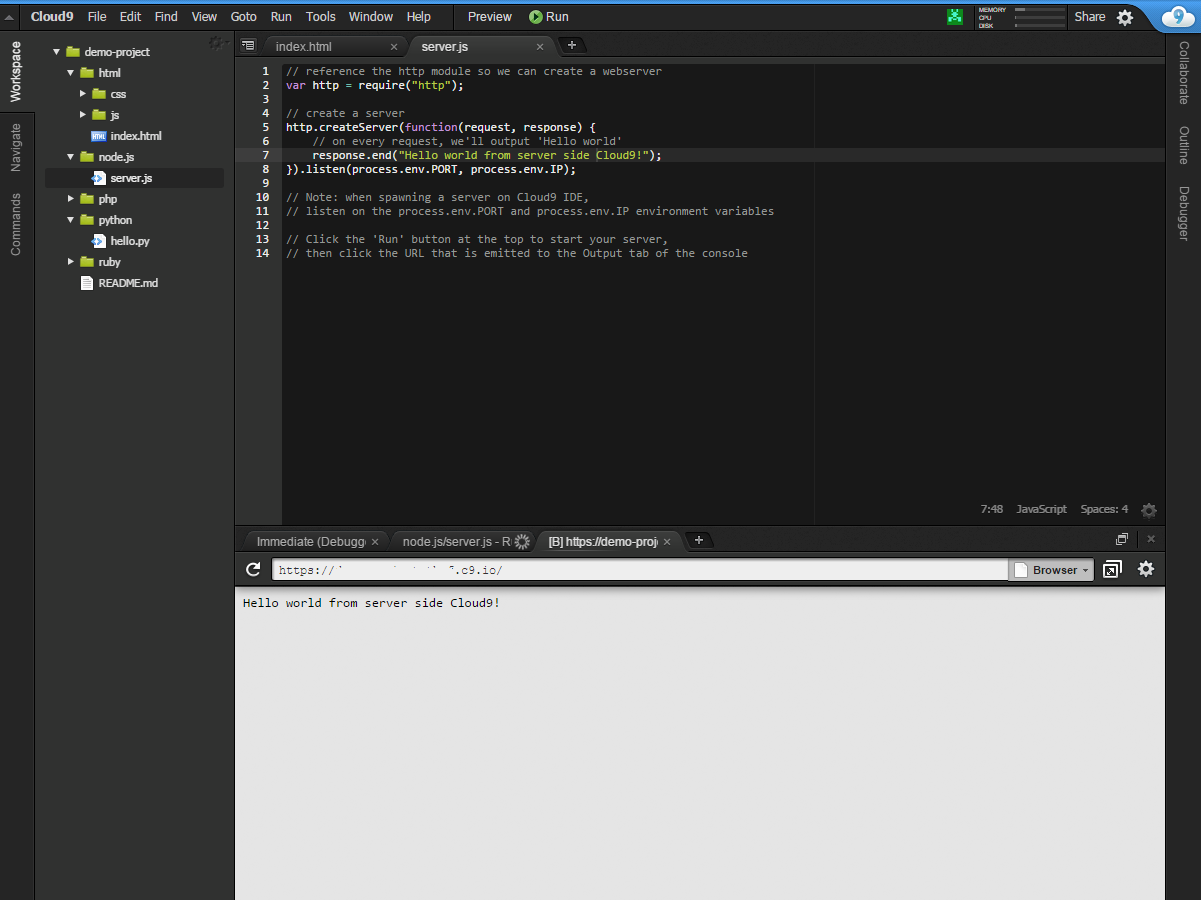
“Cloud9” е Open Source[29] WebIDE[16] проект с Non-Commercial License Agreement[45], голяма част от който е разработенa от компанията “Cloud 9 IDE, Inc”.

Някои от най-големите му плюсове са поддръжката на множество от Runtimes[5] като PHP, Ruby, Python, JavaScript, Go, възможността за едновременно редактиране на код от няколко разработчика, Templates[22] за повечето възможни артефакти на проекта и голямо разнообразие от опции за споделяне на изходния код на проекта, част от него или на готовото решение.

Съществен минус в оценката на проекта е, че не може да се използва безплатно за търговски цели (Commercial Use[46]) както и липсата на Wizards[23].



Фиг. 8 Tабло за управление на проекти в Cloud9



Фиг. 9 Работна среда за проект в Cloud9

Архитектура на среда за разработка в облака

За анализ на архитектурата на среда за разработка в облака е избран проектът Eclipse Dirigible. Архитектурата на проекта се състои от три основни компонента - IDE[6], Repository[58] и Runtime[5]. Освновните езиците за програмиране, които са използвани в проекта са: Java, JavaScript, SQL, HTML, JavaScript и CSS.

В IDE[6] частта на архитектурата се намират модулите, които отговарят за потребителския интерфейс. За Framework[8] за графичния интерфейс е избран Eclipse RAP[61]. Благодарение на Eclipse RAP[61] е възможно създаването на интерфейс, който разполага с добре познати компоненти от Eclipse Desktop IDE - папки, файлове, перспективи, Wizards[23] и други. Това позволява създаването на UI[62], който е много познат за разработчиците ползващи Eclipse Desktop IDE. Някои основни модули са:

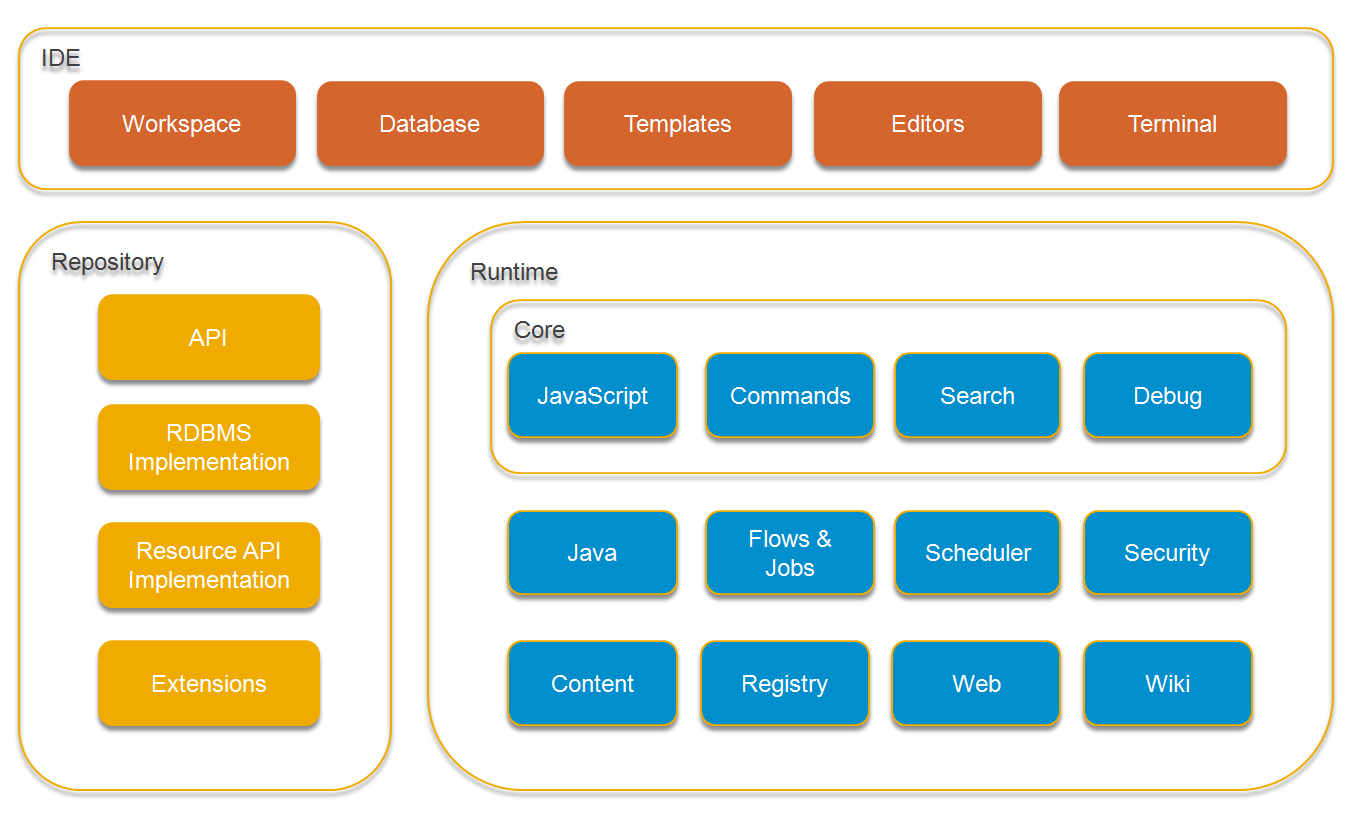
* Workspace - там се намират класовете, отговорни за графичното изобразяване на артефакти от работния плот.
* Database - в този модул се намира логиката за графична репрезентация на една от основните перспективи - “Database”. В перспективата се използват следните Views[63]: “Database View” и “SQL Console”.
* Templates - там се съдържат основни класове, Templates[22] и Wizards[23], които са отговорни за генерирането на изходен код на базата на определени правила. Този модул е базов за имплементирането на RAD[18] техниките в Eclipse Dirigible.
* Editors - модулът осигурява поддръжка на редактори за изходен код. Редакторите, които се поддържат към момента на писане на дипломната работа са: Ace[59], Eclipse Orion и WYSIWYG[60].

Repository е компонент от архитектурата, който служи за връзка между IDE[6] и Runtime[5]. Някой от основните му модули са:

* API[34] - в този модул са дефинирани интерфейсите за работа с Repository[65] обекта. Някои от тях са: IRepository, ICollection, IEntity, IResource и други.
* RDBMS[64] Implementation - там се намира имплементацията на Repository[65] API[34] за използване на база данни. В този модул се намират и SQL заявките към базата данни. Някои от класовете са: DBRepository, DBCollection, DBEntity, DBResource и други.
* Extensions - в този модул се съдържат общите за IDE[6] и Runtime[5] обекти както и поддръжката на диалекти за различни видове бази данни. Някои от класовете са: DebugSessionMetadata, DBUtils, DerbyDBSpecifier, PostgreSQLDBSpecifier, HANADBSpecifier, ConfigurationStore и други.

В Runtime[5] компонента се съдържат модулите, благодарение на които се изпълнява Server-side[54] логиката. Някои основни модули са:

* JavaScript - обработва Server-side[54] логиката написана на JavaScript.
* Commands - обработва Server-side[54] логиката за изпълняване на команди към операционната система на виртуалната машина.
* Debug - Server-side[54] логика, която позволява Debugging[50] на Server-side[54] JavaScript.

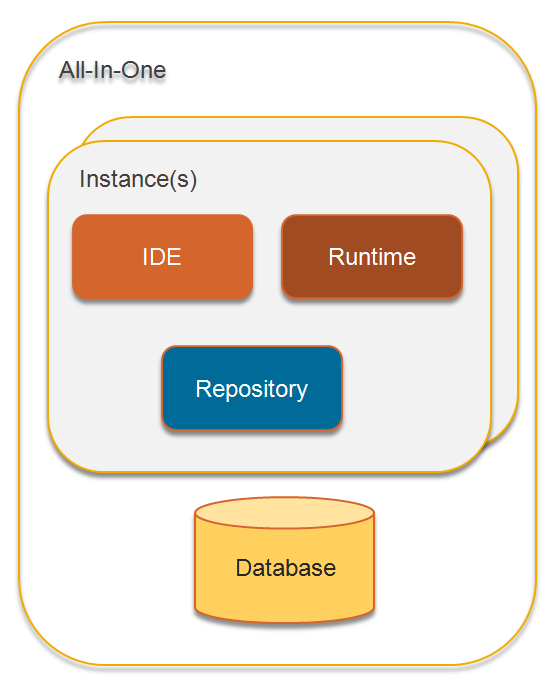
Фиг. 10 Основни компоненти на архитектурата на Eclipse Dirigible

В зависимост от конкретните нужди съществуват следните начини за Deploy[13] на Eclipse Dirigible:

* All-In-One
* Production
* RCP
* Multi-tenant

Deploy “All-In-One”

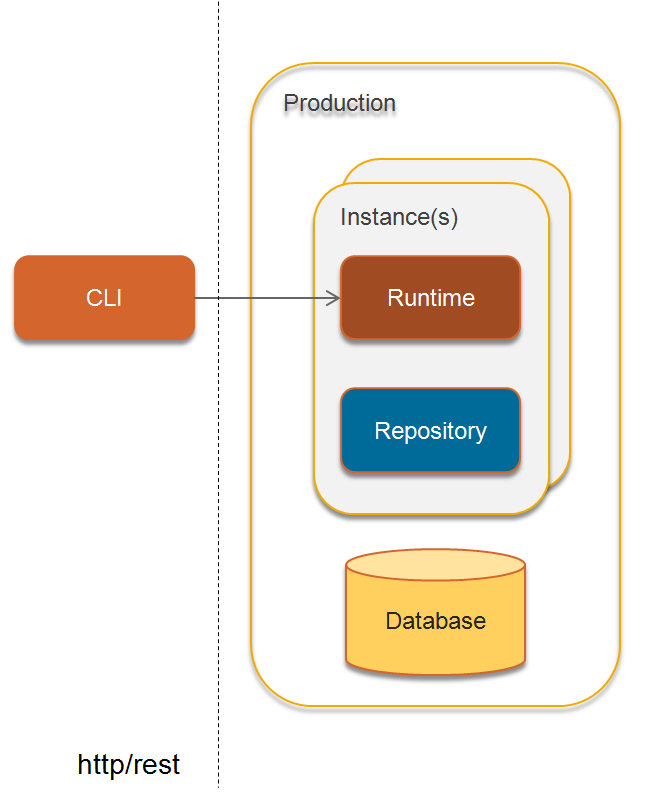
Варианта за Deploy[13] “All-In-One” предлага възможност във всяка инстанция (виртуална машина) да се съдържат трите основни компонента - IDE, Runtime и Repository. По този начин компонентите са свързани към една и съща база данни. Този модел на Deploy[13] е подходящ както за тестови Landscape[66], така и за Landscape[66] за разразботване.



Фиг. 11 Eclipse Dirigible - опция за Deploy “All-In-One”

Deploy “Production”

“Production” е опция за Deploy[13], в която разполагаме само с два от архитектурните компонента - Runtime и Repository. Този сценарий е подходящ за продуктивни Landscapes[66] - такива в които нямаме нужда от IDE, защото не се очаква в тези системи да пишем нов код. Новите версии на облачните приложения се транспортират чрез CLI[67] команди или чрез потребителския интерфейс на Registry[68].

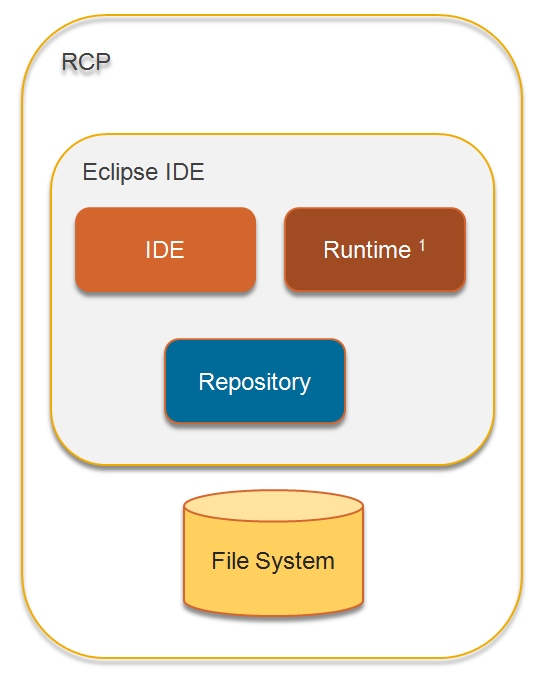


Фиг. 12 Eclipse Dirigible - опция за Deploy “Production”

Deploy “RCP”

Deploy[13] “RCP” е единият от вариантите за Local Development[57] с Eclipse Dirigible. Така основните архитектурните модули - IDE, Repository и Runtime1, стават част от Eclipse Desktop IDE. Този подход позволява преизползването на следните функционалности от Eclipse:

* JDT[69] - позволява използването на Code Completion[43] и Refactoring[49] за Java.
* Repository компонента използва стандартното Resource API на Eclipse Desktop IDE за работа с файловата система, вместо имплементацията на Resource API за работа с база данни.
* Runtime1 компонента има ограничени функции поради липсата на сървър.

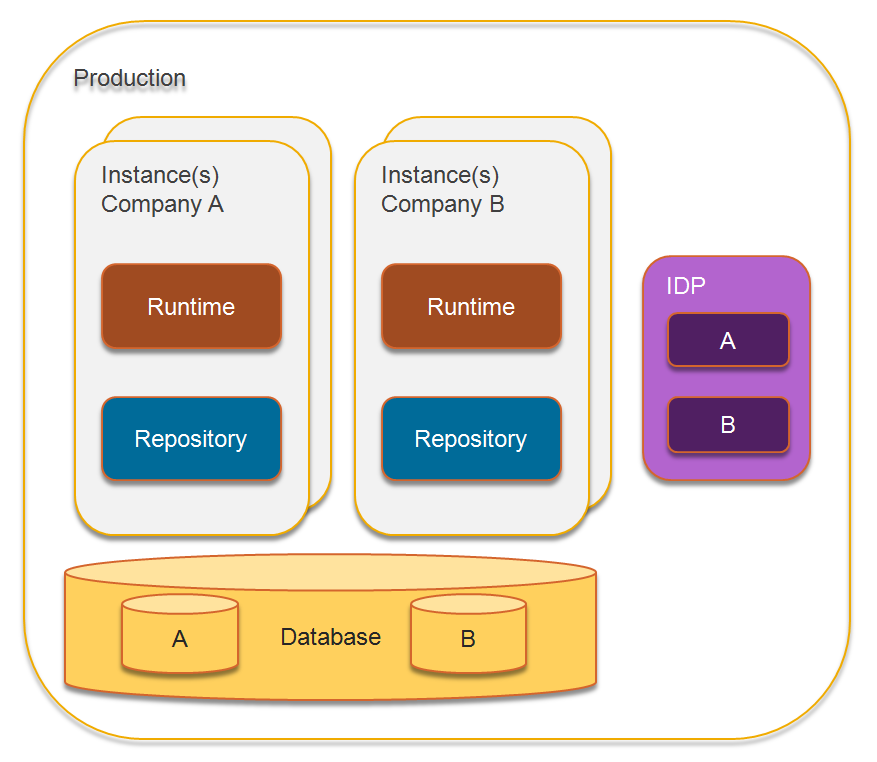


Фиг. 13 Eclipse Dirigible - опция за Deploy “RCP”

Deployment “Multi-tenant”

“Multi-tenant” е последният вариант за Deploy[13], която се допълва варианта “All-In-One” със следните неща:

* Могат да съществуват няколко инстанции едновременно.
* Всяка инстанция може да бъде конфигурирана, така че да използва собствен Identity Provider[70].
* Всяка инстанция може да бъде настроена, така че да използва отделна/изолирана схема в базата данни.



Фиг. 14 Eclipse Dirigible - опция за Deploy “Multi-tenant”

Извод

Таблица 1: Критерийно оценяване на Orion, Flux, Che, Dirigible и Cloud9

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Критерии | Orion | Flux | Che | Dirigible | Cloud9 |
| Single Sign-On[44] with social networks | ✔ | ╳ | ✔ | ╳ | ✔ |
| Code Completion[48] | ╳ | ✔ | ✔ | ╳ | ✔ |
| Refactoring[49] | ╳ | ╳ | ╳ | ╳ | ╳ |
| Debugging[50] | ╳ | ╳ | ✔ | ✔ | ✔ |
| Import/Export[51] | ✔ | ╳ | ✔ | ✔ | ✔ |
| Static Content based Code Completion[47] | ✔ | ✔ | ✔ | ✔ | ✔ |
| In-System Development Lifecycle[17] | ╳ | ╳ | ╳ | ✔ | ╳ |
| Samples[52] | ╳ | ╳ | ✔ | ✔ | ✔ |
| Templates[22] | ╳ | ╳ | ╳ | ✔ | ✔ |
| Wizards[23] | ╳ | ╳ | ╳ | ✔ | ╳ |
| Terminal[53] | ╳ | ╳ | ✔ | ✔ | ✔ |
| Server-side[54] JavaScript Runtime[5] | ╳ | ╳ | ✔ | ✔ | ✔ |
| Server-side[54] Java Runtime[5] | ╳ | ╳ | ✔ | ✔ | ✔ |
| Other Runtimes[5] | ╳ | ╳ | ✔ | ✔ | ✔ |
| Source Control System[10] | ✔ | ╳ | ✔ | ✔ | ✔ |
| Server-side[54] Extensibility[55] | ╳ | ✔ | ✔ | ✔ | ✔ |
| Client-side[56] Extensibility[55] | ✔ | ╳ | ✔ | ✔ | ✔ |
| Micro Services[36] | ╳ | ✔ | ╳ | ╳ | ╳ |
| Local Development[57] | ✔ | ✔ | ✔ | ✔ | ✔ |
| Commercial Use[46] | ✔ | ✔ | ✔ | ✔ | ╳ |
| Общо | 7 | 6 | 15 | 16 | 15 |

От направения сравнителен анализ на съществуващите решения и от таблицата за критерийно оценяване, можем да заключим, че най-подходящото решение за целите на дипломната работа е Eclipse Dirigible. Проектът събира най-много точки като отговаря на 16 от общо 20 изисквания. Някой от най-съществените предимства на Eclipse Dirigible пред останалите проекти са In-System Development[17], RAD[18] чрез Samples[54], Templates[22] и Wizards[23].